DIP*lib* function reference

dr. ir. Geert M. P. van Kempen
dr. ir. Michael van Ginkel
dr. ir. Cris L. Luengo Hendriks
dr. dr. dipl. phys. Bernd Rieger
prof. dr. ir. Lucas J. van Vliet

Quantitative Imaging Group,
Department of Applied Sciences,
Delft University of Technology

Delft
December 28, 2011

Contents

Chapter 1

Indices of functions by subject

1.1 Library Functions

1.1.1 Image Object

- ChangeDataType - Change the data type of an image
- ChangeDimensions - Changes the order of the dimensions in an image
- ChangeTo0d - Make an image zero dimensional
- HasContiguousData - Determines whether an image has all data contiguous in memory
- HasNormalStride - Determines whether an image has a normal stride
- ImageAssimilate - Inherit properties of another image
- ImageCopyProperties - Copy the properties of an image
- ImageForge - Allocate pixel data for an image
- ImageFree - Free an image
- ImageGetData - Get the data pointers of a set of images
- ImageGetDataType - Read the data type field
- ImageGetDimensionality - Read the dimensionality field
- ImageGetDimensions - Read the dimensions array
- ImageGetPlane - Read the plane number
- ImageGetStride - Read the stride array
- ImageGetType - Read the type field
- ImageNew - Allocate a structure
- ImagesCheck - Check properties of several images
- ImagesCheckTwo - Check properties of two images
- ImagesCompare - Compare properties of several images
- ImagesCompareTwo - Compare properties of two images
- ImageSetDataType - Set the data type field
- ImageSetDimensions - Set the dimensions array
- ImageSetType - Set the image type field
- ImagesSeparate - Take care of in-place operations
- ImageStrip - Restore an image to its initial ("raw") state

1.1.2 Scalar Images

- `ConvertDataType` - Converts the data type of an image
- `IsScalar` - Determines whether an image is a scalar
- `ScalarImageNew` - Allocate a scalar image

1.1.3 Strings

- `StringAppend` - Append a string to another
- `StringArrayCopy` - Copy a string array
- `StringArrayFree` - Array free function
- `StringArrayNew` - Allocate an array of strings
- `StringCat` - Concatenate two strings
- `StringCompare` - Compare two strings
- `StringCompareCaseInsensitive` - Compare two strings without minding case
- `StringCopy` - Copy a String
- `StringCrop` - Crop a string
- `StringFree` - Free a string
- `StringNew` - Allocate a string
- `StringReplace` - Replace the contents of one string with that of another
- `UnderscoreSpaces` - Replace spaces with underscores

1.1.4 Arrays

- `ArrayFree` - Array free function
- `ArrayNew` - Array allocation function
- `BooleanArrayCopy` - Copy an array
- `BooleanArrayFind` - Find value in array
- `BooleanArrayFree` - Array free function
- `BooleanArrayNew` - Array allocation function
- `BoundaryArrayFree` - Array free function
- `BoundaryArrayNew` - Array allocation function

- ComplexArrayCopy - Copy an array
- ComplexArrayFind - Find value in array
- ComplexArrayFree - Array free function
- ComplexArrayNew - Array allocation function
- ConvertArray - converts the data type of an array
- CoordinateArrayFree - Array free function
- CoordinateArrayNew - Array allocation function
- DataTypeArrayCopy - Copy an array
- DataTypeArrayFind - Find value in array
- DataTypeArrayFree - Array free function
- DataTypeArrayNew - Array allocation function
- FloatArrayCopy - Copy an array
- FloatArrayFind - Find value in array
- FloatArrayFree - Array free function
- FloatArrayNew - Array allocation function
- FrameWorkProcessArrayFree - Array free function
- FrameWorkProcessArrayNew - Array allocation function
- ImageArrayFree - Array free function
- ImageArrayNew - Array allocation function
- ImageCheckBooleanArray - Check a boolean array
- ImageCheckBoundaryArray - Check a boundary array
- ImageCheckComplexArray - Check a complex array
- ImageCheckFloatArray - Check a float array
- ImageCheckIntegerArray - Check an integer array
- IntegerArrayCopy - Copy an array
- IntegerArrayFind - Find value in array
- IntegerArrayFree - Array free function
- IntegerArrayNew - Array allocation function
- StringArrayCopy - Copy a string array

- StringArrayFree - Array free function
- StringArrayNew - Allocate an array of strings
- VoidPointerArrayCopy - Copy an array
- VoidPointerArrayFind - Find value in array
- VoidPointerArrayFree - Array free function
- VoidPointerArrayNew - Array allocation function

1.1.5 Frameworks

- MonadicFrameWork - FrameWork for monadic operations
- PixelTableFrameWork - FrameWork for PixelTable filters
- ScanFrameWork - FrameWork for scanning multiple images
- SeparableFrameWork - FrameWork for separable filters
- SingleOutputFrameWork - FrameWork for generation functions

1.1.6 Pixel Tables

- BinaryImageToPixelTable - Convert a binary image to a pixel table
- GreyValuesInPixelTable - Copy greyvalues from image in pixel table
- PixelTableAddRun - Add a new run to a pixel table
- PixelTableCreateFilter - Create a pixel table from a filter shape
- PixelTableFrameWork - FrameWork for PixelTable filters
- PixelTableGetDimensionality - Get the dimensionality of a pixel table
- PixelTableGetDimensions - Get the dimemsions of a pixel table
- PixelTableGetOffsetAndLength - Converts the pixel table's runs
- PixelTableGetOrigin - Get the origin of the pixel table
- PixelTableGetPixelCount - Get the number of pixels encoded in the pixel table
- PixelTableGetRun - Get the contents of a pixel table run
- PixelTableGetRuns - Get the number of runs in a pixel table
- PixelTableGetSize - The number of pixels in the pixel table's bounding box
- PixelTableNew - Allocate a new pixel table

- PixelTableSetRun - Initialises a pixel table run
- PixelTableShiftOrigin - Changes the origin of the pixel table
- PixelTableToBinaryImage - Convert a pixel table to a binary image

1.1.7 Data Structures

- PixelHeapFree - Destroy heap structure
- PixelHeapIsEmpty - Query heap
- PixelHeapNew - Create a new heap structure
- PixelHeapPop - Pop item onto heap
- PixelHeapPush - Push item onto heap
- PixelQueueFree - Destroy queue structure
- PixelQueueIsEmpty - Query queue
- PixelQueueNew - Create a new queue structure
- PixelQueuePop - Pop item from queue
- PixelQueuePush - Push item onto queue
- StablePixelHeapFree - Destroy heap structure
- StablePixelHeapIsEmpty - Query heap
- StablePixelHeapNew - Create a new heap structure
- StablePixelHeapPop - Pop item onto heap
- StablePixelHeapPush - Push item onto heap

1.1.8 Numerical Algorithms

- OneDimensionalSearch - Numerical algorithm

1.1.9 Sorting

- DistributionSort - Sort a block of data
- DistributionSortIndices - Sort indices to block of data
- DistributionSortIndices16 - Sort indices to a block of data
- GetRank - Value selection function

- ImageSort - Sort image data
- ImageSortIndices - Sort indices to image data
- InsertionSort - Sort a block of data
- InsertionSortIndices - Sort indices to a block of data
- InsertionSortIndices16 - Sort indices to a block of data
- QuickSort - Sort a block of data
- QuickSortAnything - Sort data of any type
- QuickSortIndices - Sort indices to a block of data
- QuickSortIndices16 - Sort indices to a block of data
- Sort - Sort a block of data
- SortAnything - Sort data of any type
- SortCompareFunction - Typedef for comparison function (sorting)
- SortIndices - Sort indices to a block of data
- SortIndices16 - Sort indices to a block of data
- SortSwapFunction - Typedef for swap and copy function (sorting)

1.1.10 Indexing

- CoordinateToIndex - Convert coordinate to pixel index
- dip__PixelGetFloat - Midlevel PixelIO function
- dip__PixelGetInteger - Midlevel PixelIO function
- dip__PixelSetFloat - Midlevel PixelIO function
- dip__PixelSetInteger - Midlevel PixelIO function
- Get - Get a pixel value
- GetComplex - Get complex pixel value
- GetFloat - Get float pixel value
- GetInteger - Get integer pixel value
- IndexToCoordinate - Convert pixel index to coordinate
- IndexToCoordinateWithSingletons - Convert pixel index to coordinate
- NeighbourIndicesListMake - Get indices to direct neighbours

- NeighbourListMake - Get list of direct neighbours
- NeighbourListMakeChamfer - Get list of neighbours based on Chamfer metric
- NeighbourListMakeImage - Get list of neighbours based on metric in image
- NeighbourListToIndices - Get indices to neighbours
- Set - the value of a pixel
- SetComplex - Set a pixel value
- SetFloat - Set a pixel value
- SetInteger - Set a pixel value

1.1.11 Memory Management

- MemoryCopy - Copy memory blocks
- MemoryFree - Free a chunk of memory
- MemoryFunctionsSet - Sets memory allocation functions
- MemoryNew - Allocate and track memory
- MemoryReallocate - Reallocate a chunk of memory
- ResourcesFree - Free resources
- ResourcesMerge - Add one resource list to another
- ResourcesNew - Allocate a resource tracking structure
- ResourceSubscribe - Track a resource
- ResourceUnsubscribe - Stop tracking a resource

1.1.12 Support Functions

- DataTypeAllowed - Check whether a data type is allowed
- DataTypeGetInfo - Get information about a data type
- error.h - Contains error messages
- ErrorFree - Free a DIPlib call tree
- Exit - Clean up before exiting
- FillBoundaryArray - Fill the border of array according to the boundary condition
- GetLibraryInformation - Support function

- GetUniqueNumber - Obtain an unique value
- GlobalBoundaryConditionGet - Get global Boundary Conditions
- GlobalBoundaryConditionSet - Set global boundary conditions
- GlobalFilterShapeGet - Get global filter shape value
- GlobalFilterShapeSet - Set the global filter shape value
- GlobalGaussianTruncationGet - Get the global gaussian truncation
- GlobalGaussianTruncationSet - Set the global gaussian truncation
- Initialise - Initialise DIPlib
- macros.h - Various macros
- ovl.h - Call an overloaded function
- PhysicalDimensionsCopy - Copy a Physical Dimensions
- PhysicalDimensionsFree - Free a Physical Dimensions data structure
- PhysicalDimensionsIsIsotropic - Checks if the Physical Dimensions are isotropic
- PhysicalDimensionsNew - Allocates a new Physical Dimensions structure
- Register - Generic registry function
- RegisterClass - Register a registry class
- RegistryArrayNew - Allocate a registry array
- RegistryGet - Get a registry item
- RegistryList - Get an array of registry IDs
- RegistryValid - Validate an registry item
- TimerGet - Timing functions
- TimerSet - Timing functions
- tpi.h - Type iterator
- Unregister - Remove a registry item

1.2 File I/O functions

1.2.1 File IO

- Colour2Gray - Convert ND image with colour information to a (n-1)D grayvalue image (in dipIO)
- ImageFileGetInfo - Get information about image in file (in dipIO)
- ImageFileInformationFree - Free a Image File Information structure (in dipIO)
- ImageFileInformationNew - Allocate an Image File Information structure (in dipIO)
- ImageIsGIF - Confirm that a file is a GIF file (in dipIO)
- ImageIsICS - Confirm that a file is an ICS file (in dipIO)
- ImageIsJPEG - Confirm that a file is a JPEG file (in dipIO)
- ImageIsLSM - Confirm that a file is a Zeiss LSM file (in dipIO)
- ImageIsTIFF - Confirm that a file is a TIFF file (in dipIO)
- ImageRead - Read grey-value image from file (in dipIO)
- ImageReadColour - Read colour image from file (in dipIO)
- ImageReadCSV - Read comma-separated values from file (in dipIO)
- ImageReadCSVInfo - Get information about image in comma-separated values file (in dipIO)
- ImageReadGIF - Read a GIF image from file (in dipIO)
- ImageReadGIFInfo - Get information about image in GIF file (in dipIO)
- ImageReadICS - Read ICS image from file (in dipIO)
- ImageReadICSInfo - Get information about image in ICS file (in dipIO)
- ImageReadJPEG - Read JPEG image from file (in dipIO)
- ImageReadJPEGInfo - Get information about image in JPEG file (in dipIO)
- ImageReadLSM - Read Zeiss LSM image from file (in dipIO)
- ImageReadLSMInfo - Get information about image in LSM file (in dipIO)
- ImageReadPIC - Read BioRad PIC image from file (in dipIO)
- ImageReadPICInfo - Get information about image in BioRad PIC file (in dipIO)
- ImageReadROI - Read a portion of a grey-value image from file (in dipIO)
- ImageReadTIFF - Read TIFF image from file (in dipIO)

- ImageReadTIFFInfo - Get information about image in TIFF file (in dipIO)
- ImageWrite - Write grey-value image to file (in dipIO)
- ImageWriteColour - Write colour image to file (in dipIO)
- ImageWriteCSV - Write image to a comma-separated-value file (in dipIO)
- ImageWriteEPS - Write image to Encapsulated PostScript file (in dipIO)
- ImageWriteFLD - Write image to AVS field file (in dipIO)
- ImageWriteGIF - Write image to a GIF file (in dipIO)
- ImageWriteICS - Write ICS image to file (in dipIO)
- ImageWriteJPEG - Write JPEG image to file (in dipIO)
- ImageWritePS - Write image to PostScript file (in dipIO)
- ImageWriteTIFF - Write TIFF image to file (in dipIO)
- MeasurementRead - Read measurement results from a file
- MeasurementWrite - Write measurement results to a file
- MeasurementWriteCSV - Write measurement results to a CSV file
- MeasurementWriteHTML - Write measurement results to an HTML file
- MeasurementWriteText - Write measurement results as readable text

1.3 Image Processing Functions

1.3.1 Mathematics

- Abs - Arithmetic function
- Acos - trigonometric function
- Add - arithmetic function
- AddComplex - arithmetic function
- AddFloat - arithmetic function
- AddInteger - arithmetic function
- And - logic operation
- Arith - arithmetic function
- Arith_ComplexSeparated - arithmetic function
- Asin - trigonometric function

- Atan - trigonometric function
- Atan2 - arithmetic function
- BesselJ0 - mathematical function
- BesselJ1 - mathematical function
- BesselJN - mathematical function
- BesselY0 - mathematical function
- BesselY1 - mathematical function
- BesselYN - mathematical function
- Ceil - Arithmetic function
- Compare - Compare grey values in two images
- Cos - trigonometric function
- Cosh - trigonometric function
- CumulativeSum - statistics function
- Div - arithmetic function
- DivComplex - arithmetic function
- DivFloat - arithmetic function
- DivInteger - arithmetic function
- Equal - Compare grey values in two images
- Erf - mathematical function
- Erfc - mathematical function
- Exp - arithmetic function
- Exp10 - arithmetic function
- Exp2 - arithmetic function
- Floor - Arithmetic function
- Fraction - Arithmetic function
- GetMaximumAndMinimum - statistics function
- Greater - Compare grey values in two images
- IDivergence - difference measure
- Imaginary - Arithmetic function

- Invert - logic operation
- Lesser - Compare grey values in two images
- Ln - arithmetic function
- LnGamma - mathematical function
- LnNormError - difference measure
- Log10 - arithmetic function
- Log2 - arithmetic function
- Max - arithmetic function
- MaxFloat - arithmetic function
- Maximum - statistics function
- mBesselJ0 - mathematical function
- mBesselJ1 - mathematical function
- mBesselJN - mathematical function
- mBesselY0 - mathematical function
- mBesselY1 - mathematical function
- mBesselYN - mathematical function
- Mean - statistics function
- MeanAbsoluteError - difference measure
- MeanError - difference measure
- MeanModulus - statistics function
- MeanSquareError - difference measure
- MeanSquareModulus - statistics function
- Median - statistics function
- mErf - mathematical function
- mErfc - mathematical function
- mExp10 - mathematical function
- mExp2 - mathematical function
- mFraction - mathematical function
- mGammaP - mathematical function

- `mGammaQ` - mathematical function
- `Min` - arithmetic function
- `MinFloat` - arithmetic function
- `Minimum` - statistics function
- `mLnGamma` - mathematical function
- `mLog2` - mathematical function
- `mNearestInt` - mathematical function
- `Modulo` - Arithmetic function
- `Modulus` - Arithmetic function
- `mReciprocal` - mathematical function
- `mSign` - mathematical function
- `mSinc` - mathematical function
- `mTruncate` - mathematical function
- `Mul` - arithmetic function
- `MulComplex` - arithmetic function
- `MulConjugate` - arithmetic function
- `MulConjugateComplex` - arithmetic function
- `MulFloat` - arithmetic function
- `MulInteger` - arithmetic function
- `NearestInt` - Arithmetic function
- `NormaliseSum` - Normalise the sum of the pixel values
- `NotEqual` - Compare grey values in two images
- `NotGreater` - Compare grey values in two images
- `NotLesser` - Compare grey values in two images
- `Or` - logic operation
- `Percentile` - statistics function
- `Phase` - Arithmetic function
- `RadialMaximum` - statistics function
- `RadialMean` - statistics function

- RadialMinimum - statistics function
- RadialSum - statistics function
- Real - Arithmetic function
- Reciprocal - arithmetic function
- RootMeanSquareError - difference measure
- Select - Configurable selection function
- Sign - Arithmetic function
- Sin - trigonometric function
- Sinc - mathematical function
- SingularValueDecomposition - Singular value decomposition
- Sinh - trigonometric function
- Sqrt - arithmetic function
- StandardDeviation - statistics function
- Sub - arithmetic function
- SubComplex - arithmetic function
- SubFloat - arithmetic function
- SubInteger - arithmetic function
- Sum - statistics function
- SumModulus - statistics function
- Tan - trigonometric function
- Tanh - trigonometric function
- TensorImageInverse - Invert tensor image
- Truncate - Arithmetic function
- Variance - statistics function
- WeightedAdd - arithmetic function
- WeightedDiv - arithmetic function
- WeightedMul - arithmetic function
- WeightedSub - arithmetic function
- Xor - logic operation

1.3.2 Statistics

- `ChordLength` - Compute the chord lengths of the different phases
- `CumulativeSum` - statistics function
- `GetMaximumAndMinimum` - statistics function
- `IDivergence` - difference measure
- `LnNormError` - difference measure
- `Maximum` - statistics function
- `Mean` - statistics function
- `MeanAbsoluteError` - difference measure
- `MeanError` - difference measure
- `MeanModulus` - statistics function
- `MeanSquareError` - difference measure
- `MeanSquareModulus` - statistics function
- `Median` - statistics function
- `Minimum` - statistics function
- `PairCorrelation` - Compute the pair correlation function
- `Percentile` - statistics function
- `ProbabilisticPairCorrelation` - Compute the probabilistic pair correlation function
- `RadialMaximum` - statistics function
- `RadialMean` - statistics function
- `RadialMinimum` - statistics function
- `RadialSum` - statistics function
- `RootMeanSquareError` - difference measure
- `StandardDeviation` - statistics function
- `Sum` - statistics function
- `SumModulus` - statistics function
- `Variance` - statistics function

1.3.3 Manipulation

- Crop - Remove the outer parts of an image
- dip_PixelGetFloat - Midlevel PixelIO function
- dip_PixelGetInteger - Midlevel PixelIO function
- dip_PixelSetFloat - Midlevel PixelIO function
- dip_PixelSetInteger - Midlevel PixelIO function
- ExtendRegion - Image manipulation functions
- Get - Get a pixel value
- GetComplex - Get complex pixel value
- GetFloat - Get float pixel value
- GetInteger - Get integer pixel value
- GetLine - Get a line from an image
- GetSlice - Get a slice from an image
- Map - Remaps an image
- Mirror - Mirrors an image
- PutLine - Put a line in an image
- PutSlice - Put a slice in an image
- Resampling - Interpolation function
- Rotation - Interpolation function
- Rotation3d - Interpolation function
- Rotation3d_Axis - Interpolation function
- Set - the value of a pixel
- SetComplex - Set a pixel value
- SetFloat - Set a pixel value
- SetInteger - Set a pixel value
- Shift - an image manipulation function
- Skewing - Interpolation function
- Subsampling - Interpolation function
- Wrap - Wrap an image

1.3.4 Interpolation

- `Resampling` - Interpolation function
- `Rotation` - Interpolation function
- `Rotation3d` - Interpolation function
- `Rotation3d_Axis` - Interpolation function
- `Skewing` - Interpolation function
- `SubpixelLocation` - Gets coordinates of an extremum with sub-pixel precision
- `SubpixelMaxima` - Gets coordinates of local maxima with sub-pixel precision
- `SubpixelMinima` - Gets coordinates of local minima with sub-pixel precision
- `Subsampling` - Interpolation function

1.3.5 Painting

- `PaintBox` - Paint a box
- `PaintDiamond` - Paint a diamond-shaped object
- `PaintEllipsoid` - Paint an ellipsoid

1.3.6 Linear Filters

- `Convolve1d` - Perform a 1D convolution
- `ConvolveFT` - Fourier transform–based convolution filter
- `Derivative` - Derivative filter
- `FiniteDifference` - A linear gradient filter
- `FiniteDifferenceEx` - A linear gradient filter
- `GaborIIR` - Infinite impulse response filter
- `Gauss` - Gaussian Filter
- `GaussFT` - Gaussian Filter through the Fourier Domain
- `GaussIIR` - Infinite impulse response filter
- `GeneralConvolution` - Genaral convolution filter
- `Laplace` - Second order derivative filter
- `SeparableConvolution` - FrameWork for separable convolution filters

- Sharpen - Enhance an image
- SobelGradient - A linear gradient filter
- Uniform - Uniform filter

1.3.7 Derivative Filters

- Derivative - Derivative filter
- Dgg - Second order derivative filter
- FiniteDifference - A linear gradient filter
- FiniteDifferenceEx - A linear gradient filter
- Gauss - Gaussian Filter
- GaussFT - Gaussian Filter through the Fourier Domain
- GradientDirection2D - Derivative filter
- GradientMagnitude - Derivative filter
- Laplace - Second order derivative filter
- LaplaceMinDgg - Second order derivative filter
- LaplacePlusDgg - Second order derivative filter
- SobelGradient - A linear gradient filter

1.3.8 Non-Linear Filters

- BiasedSigma - Adaptive edge sharpening & contrast enhancing filter
- Closing - Morphological closing operation
- Dilation - Local maximum filter
- Erosion - Local minimum filter
- GaussianSigma - Adaptive Gaussian smoothing filter
- GeneralisedKuwahara - Generalised Kuwahara filter
- GeneralisedKuwaharaImproved - Generalised Kuwahara filter
- Kuwahara - Edge perserving smoothing filter
- KuwaharaImproved - Edge perserving smoothing filter
- MedianFilter - Non-linear smoothing filter

- `MorphologicalSmoothing` - Morphological smoothing filter
- `Opening` - Morphological opening operation
- `PercentileFilter` - Rank-order filter
- `Sigma` - Adaptive uniform smoothing filter
- `VarianceFilter` - Sample Variance Filter

1.3.9 Binary Filters

- `BinaryClosing` - Binary morphological closing operation
- `BinaryDilation` - Binary morphological dilation operation
- `BinaryErosion` - Binary morphological erosion operation
- `BinaryOpening` - Binary morphological opening operation
- `BinaryPropagation` - Morphological propagation of binary objects
- `EdgeObjectsRemove` - Remove binary edge objects
- `EuclideanSkeleton` - binary skeleton operation
- `GrowRegions` - Dilate the regions in a labelled image
- `Label` - Label a binary image

1.3.10 Mathematical Morphology

- `AreaOpening` - Morphological filter
- `BinaryClosing` - Binary morphological closing operation
- `BinaryDilation` - Binary morphological dilation operation
- `BinaryErosion` - Binary morphological erosion operation
- `BinaryOpening` - Binary morphological opening operation
- `BinaryPropagation` - Morphological propagation of binary objects
- `Closing` - Morphological closing operation
- `Dilation` - Local maximum filter
- `DirectedPathOpening` - Morphological filter
- `EdgeObjectsRemove` - Remove binary edge objects
- `Erosion` - Local minimum filter

- EuclideanSkeleton - binary skeleton operation
- GrowRegions - Dilate the regions in a labelled image
- GrowRegionsWeighted - Grow labelled regions using grey-weighted distances
- Lee - Morphological edge detector
- LocalMinima - Marks local minima (or regional minima)
- Maxima - Detects local maxima
- Minima - Detects local minima
- MorphologicalGradientMagnitude - Morphological edge detector
- MorphologicalRange - Morphological edge detector
- MorphologicalReconstruction - Morphological filter
- MorphologicalSmoothing - Morphological smoothing filter
- MorphologicalThreshold - Morphological smoothing filter
- MultiScaleMorphologicalGradient - Morphological edge detector
- Opening - Morphological opening operation
- PathOpening - Morphological filter
- SeededWatershed - Morphological segmentation
- Tophat - Morphological high-pass filter
- UpperEnvelope - Upper envelope transform (a flooding and an algebraic closing)
- Watershed - Morphological segmentation

1.3.11 Point Operations

- Clip - Point operation
- Compare - Compare grey values in two images
- ContrastStretch - Point operation
- Equal - Compare grey values in two images
- ErfClip - Point Operation
- Greater - Compare grey values in two images
- HysteresisThreshold - Point Operation
- IsodataThreshold - Point operation

- `Lesser` - Compare grey values in two images
- `NotEqual` - Compare grey values in two images
- `NotGreater` - Compare grey values in two images
- `NotLesser` - Compare grey values in two images
- `NotZero` - Point Operation
- `RangeThreshold` - Point Operation
- `Select` - Configurable selection function
- `SelectValue` - Point Operation
- `Threshold` - Point Operation

1.3.12 Transforms

- `FourierTransform` - Computes the Fourier transform
- `HartleyTransform` - Computes the Hartley transform

1.3.13 Distance Transforms

- `EuclideanDistanceTransform` - Euclidean distance transform
- `GreyWeightedDistanceTransform` - Grey weighted distance transform
- `GrowRegionsWeighted` - Grow labelled regions using grey-weighted distances
- `VectorDistanceTransform` - Euclidean vector distance transform

1.4 Application Functions

1.4.1 Smoothing

- `Closing` - Morphological closing operation
- `Gauss` - Gaussian Filter
- `GaussFT` - Gaussian Filter through the Fourier Domain
- `Kuwahara` - Edge perserving smoothing filter
- `KuwaharaImproved` - Edge perserving smoothing filter
- `MedianFilter` - Non-linear smoothing filter
- `MorphologicalSmoothing` - Morphological smoothing filter

- MorphologicalThreshold - Morphological smoothing filter
- Opening - Morphological opening operation
- PercentileFilter - Rank-order filter
- Uniform - Uniform filter
- UpperEnvelope - Upper envelope transform (a flooding and an algebraic closing)

1.4.2 Sharpening

- Sharpen - Enhance an image

1.4.3 Line and Edge Detection

- DanielsonLineDetector - Line detector
- GradientMagnitude - Derivative filter
- Laplace - Second order derivative filter
- Lee - Morphological edge detector
- MorphologicalGradientMagnitude - Morphological edge detector
- MorphologicalRange - Morphological edge detector
- MultiScaleMorphologicalGradient - Morphological edge detector

1.4.4 Extrema Detection

- LocalMinima - Marks local minima (or regional minima)
- Maxima - Detects local maxima
- Minima - Detects local minima
- SubpixelLocation - Gets coordinates of an extremum with sub-pixel precision
- SubpixelMaxima - Gets coordinates of local maxima with sub-pixel precision
- SubpixelMinima - Gets coordinates of local minima with sub-pixel precision

1.4.5 Object Generation

- CityBlockDistanceToPoint - Distance generation function
- EllipticDistanceToPoint - Distance generation function

- EuclideanDistanceToPoint - Distance generation function
- FTBox - Generates the Fourier transform of a box
- FTCross - Generates the Fourier transform of a cross
- FTCube - Generates the Fourier transform of a cube
- FTEllipsoid - Generates Fourier transform of a ellipsoid
- FTGaussian - Generates the Fourier transform of a Gaussian
- FTSphere - Generated Fourier transform of a sphere
- IncoherentOTF - Generates an incoherent OTF
- IncoherentPSF - Generates an incoherent PSF
- PaintBox - Paint a box
- PaintDiamond - Paint a diamond-shaped object
- PaintEllipsoid - Paint an ellipsoid
- TestObjectAddNoise - TestObject generation function
- TestObjectBlur - TestObject generation function
- TestObjectCreate - TestObject generation function
- TestObjectModulate - TestObject generation function

1.4.6 Noise Generation

- BinaryNoise - Generates an image disturbed by binary noise
- BinaryRandomVariable - Binary random variable generator
- GaussianNoise - Generate an image disturbed by Gaussian noise
- GaussianRandomVariable - Gaussian random variable generator
- PoissonNoise - Generate an image disturbed by Poisson noise
- PoissonRandomVariable - Poisson random variable generator
- RandomSeed - Initialise random number generator
- RandomSeedVector - Initialise random number generator
- RandomVariable - Random number generator
- UniformNoise - Generate an image disturbed by uniform noise
- UniformRandomVariable - Uniform random variable generator

1.4.7 Image Restoration

- `AttenuationCorrection` - Attenuation correction algorithm
- `ExponentialFitCorrection` - Exponential fit based attenuation correction
- `PseudoInverse` - Image restoration filter
- `SimulatedAttenuation` - Simulation of the attenuation process
- `TikhonovMiller` - Image restoration filter
- `TikhonovRegularizationParameter` - Determine the value of the regularisation parameter
- `Wiener` - Image Restoration Filter

1.4.8 Shift Estimation

- `CrossCorrelationFT` - Normalized cross-correlation using the Fourier Transform
- `FindShift` - Estimate the shift between images

1.4.9 Segmentation

- `Canny` - Edge detector
- `HysteresisThreshold` - Point Operation
- `IsodataThreshold` - Point operation
- `RangeThreshold` - Point Operation
- `SeededWatershed` - Morphological segmentation
- `Threshold` - Point Operation
- `Watershed` - Morphological segmentation

1.4.10 Analysis

- `Canny` - Edge detector
- `ChordLength` - Compute the chord lengths of the different phases
- `DanielsonLineDetector` - Line detector
- `ImageChainCode` - Extracts all chain codes from a labeled image
- `Label` - Label a binary image

- Measure - Measure object features
- PairCorrelation - Compute the pair correlation function
- ProbabilisticPairCorrelation - Compute the probabilistic pair correlation function
- StructureTensor2D - Two dimensional Structure Tensor
- SubpixelLocation - Gets coordinates of an extremum with sub-pixel precision
- SubpixelMaxima - Gets coordinates of local maxima with sub-pixel precision
- SubpixelMinima - Gets coordinates of local minima with sub-pixel precision

1.4.11 Measurement

- ChainCodeArrayFree - Chain code array deallocation
- ChainCodeArrayNew - Chain code array allocation
- ChainCodeConvexHull - Compute convex hull from chain code
- ChainCodeFree - Chain code object deallocation
- ChainCodeGetChains - Chain code access function
- ChainCodeGetConnectivity - Chain code access function
- ChainCodeGetFeret - Chain code measurement function
- ChainCodeGetLabel - Chain code access function
- ChainCodeGetLength - Chain code measurement function
- ChainCodeGetLongestRun - Chain code measurement function
- ChainCodeGetRadius - Chain code measurement function
- ChainCodeGetSize - Chain code access function
- ChainCodeGetStart - Chain code access function
- ChainCodeNew - Chain code object allocation
- ConvexHullGetArea - Convex hull measurement function
- ConvexHullGetFeret - Convex hull measurement function
- ConvexHullGetPerimeter - Convex hull measurement function
- FeatureAnisotropy2D - Measure the anisotropy in a labeled region
- FeatureBendingEnergy - Undocumented measurement function
- FeatureCenter - Measure the object's center

- FeatureChainCodeBendingEnergy - Undocumented measurement function
- FeatureChainCodeFunction - Measurement feature #measure function
- FeatureComposeFunction - Measurement feature #compose function
- FeatureCompositeFunction - Measurement feature #measure function
- FeatureConvertFunction - Measurement feature #convert function
- FeatureConvexArea - Measure the area of the object's convex hull
- FeatureConvexity - Measure the object's convexity
- FeatureConvexPerimeter - Measure the perimeter of the object's convex hull
- FeatureConvHullFunction - Measurement feature #measure function
- FeatureCreateFunction - Measurement feature #create function
- FeatureDescriptionFree - Free a Feature Description
- FeatureDescriptionFunction - Measurement feature #description function
- FeatureDescriptionGetDescription - Get the description of the described feature
- FeatureDescriptionGetLabels - Get the labels of the described feature
- FeatureDescriptionGetName - Get the name of the described feature
- FeatureDescriptionGetUnits - Get the Units of the described feature
- FeatureDescriptionNew - Allocate a new FeatureDescription
- FeatureDescriptionSetDescription - Set the description of the described feature
- FeatureDescriptionSetDimensionLabels - Label set convenience function
- FeatureDescriptionSetLabel - Set the name of a particular feature label
- FeatureDescriptionSetLabels - Set the labels of the described feature
- FeatureDescriptionSetName - Set the name of the described feature
- FeatureDescriptionSetUnit - Set the units of a particular feature dimension
- FeatureDescriptionSetUnits - Set the units of a described feature
- FeatureDimension - Measure the object's dimensions
- FeatureExcessKurtosis - Undocumented measurement function
- FeatureFeret - Measure the object's Feret diameters
- FeatureGinertia - Measure the object's inertia
- FeatureGmu - Measure the object's inertia

- `FeatureGravity` - Measure the object's gravity
- `FeatureImageFunction` - Measurement feature #measure function
- `FeatureInertia` - Measure the object's inertia
- `FeatureLineFunction` - Measurement feature #measure function
- `FeatureLongestChaincodeRun` - Undocumented measurement function
- `FeatureMass` - Measure the mass of the object (sum of grey-values)
- `FeatureMaximum` - Measure the object's maximum coordinate value
- `FeatureMaxVal` - Measure the object's maximum intensity
- `FeatureMean` - Measure the object's mean intensity
- `FeatureMinimum` - Measure the object's minimum coordinate value
- `FeatureMinVal` - Measure the object's minimum intensity
- `FeatureMu` - Measure the object's inertia
- `FeatureOrientation2D` - Undocumented measurement function
- `FeatureP2A` - Measure the circularity of the object
- `FeaturePerimeter` - Measure the object's perimeter length
- `FeatureRadius` - Measure the object's radius statistics
- `FeatureShape` - Measure shape parameters of the object
- `FeatureSize` - Measure the object's area/volume
- `FeatureSkewness` - Undocumented measurement function
- `FeatureStdDev` - Measure the standard deviation of the object's intensity
- `FeatureSum` - Measure the sum of the grey values of the object
- `FeatureSurfaceArea` - Measure the area of the object's surface
- `FeatureValueFunction` - Measurement feature #value function
- `GetObjectLabels` - Lists object labels in image
- `ImageChainCode` - Extracts all chain codes from a labeled image
- `Label` - Label a binary image
- `Measure` - Measure object features
- `MeasurementFeatureConvert` - Convert the data of a measurement feature
- `MeasurementFeatureDescription` - Measurement Description access function

- MeasurementFeatureFormat - Feature data format convenience function
- MeasurementFeatureRegister - Register a measurement function
- MeasurementFeatureRegistryFeatureDescription - Get the feature description of a registered measurement feature
- MeasurementFeatureRegistryFeatureNeedsIntensityImage - Checks whether the measurement function needs an intensity image
- MeasurementFeatureRegistryGet - Get the registry information of a measurement feature
- MeasurementFeatureRegistryList - Obtain a list of the registered measurement features
- MeasurementFeatures - Get the measurement ID array
- MeasurementFeatureSize - Feature data convenience function
- MeasurementFeatureValid - Verify a measurement feature ID
- MeasurementForge - Allocate the data of a measurement data structure
- MeasurementFree - Free a measurement data structure
- MeasurementGetName - Get the name of a Measurement structure
- MeasurementGetPhysicalDimensions - Get the physical dimensions info of a measurement
- MeasurementID - Get the ID of a Measurement structure
- MeasurementIsValid - Checks whether a measurement is valid
- MeasurementNew - Create new measurement data structure
- MeasurementNumberOfFeatures - Get the number of measurement feature IDs
- MeasurementNumberOfObjects - Get the number of object IDs
- MeasurementObjectData - Object data access function
- MeasurementObjects - Get an object ID array
- MeasurementObjectValid - Verify an object ID
- MeasurementObjectValue - Object value access function
- MeasurementSetName - Set the name of a measurement structure
- MeasurementSetPhysicalDimensions - Set the physical dimensions info of the measurement
- MeasurementToHistogram - Creats a histogram for a measurement

- `MeasurementToImage` - Exports the data in a measurement structure to an image
- `ObjectToMeasurement` - Convert object label value to measurement value
- `PhysicalDimensionsCopy` - Copy a Physical Dimensions
- `PhysicalDimensionsFree` - Free a Physical Dimensions data structure
- `PhysicalDimensionsIsIsotropic` - Checks if the Physical Dimensions are isotropic
- `PhysicalDimensionsNew` - Allocates a new Physical Dimensions structure
- `SmallObjectsRemove` - Remove small objects from an image

1.4.12 Functions for Microscopy

- `AttenuationCorrection` - Attenuation correction algorithm
- `ExponentialFitCorrection` - Exponential fit based attenuation correction
- `IncoherentOTF` - Generates an incoherent OTF
- `IncoherentPSF` - Generates an incoherent PSF
- `SimulatedAttenuation` - Simulation of the attenuation process

Chapter 2

Function reference

Abs

Arithmetic function

SYNOPSIS

`dip_Error dip_Abs ( in, out )`

DATA TYPES

binary, integer, **integer**, **float**, **complex**

FUNCTION

Computes the absolute value of the input image values.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output

SEE ALSO

`Ceil`, `Floor`, `Sign`, `Truncate`, `Fraction`, `NearestInt`

Acos

trigonometric function

SYNOPSIS

`dip_Error dip_Acos ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the arc cosine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Atan, Atan2, Sinh, Cosh, Tanh

AdaptiveBanana

Performs Gaussian filtering steered by paramter images

SYNOPSIS

```
#include "dip_adaptive.h"
dip_Error DIP_TPI_FUNC(dip_AdaptiveBanana)( in, out, para_images, curv_image,
filterSize, order, truncation )
```

DATA TYPES

sfloat

FUNCTION

This function performs Gaussian filtering steerd by the information stored in the paramter images (local orientation) and in the curvature image. The meaning of the parameter images depends on the dimensionality of the input image. Up to now only 2 and 3D images are supported for adaptive filtering. If the input image is not of type **float** it is converted to that type.

`para_images`: ImageArray containing orientation images.

2D: angle of the orientation.

3D: polar coordinate phi, theta for intrinsic 1D structures polar coordinates of two orientations for intrinsic 2D structures.

`filterSize`: Array containing the sigmas of the derivatives.

For intrinsic 1D structures, the first value is along the contour, the second perpendicular to it.

For intrinsic 2D structures, the first two are in the plane, whereas the other is perpendicular to them. If a value is zero no convolution is done is this direction.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_ImageArray	para_images	Parameter images
dip_Image	curv	Curvature image
dip_FloatArray	filterSize	Size of the filter
dip_IntegerArray	order	Order of the Gaussian derivative
dip_int	truncation	Truncation of the Gaussian

SEE ALSO

AdaptivePercentile, AdaptiveGauss, Gauss

LITERATURE

P. Bakker, *"Image structure analysis for seismic interpretation"*, PhD Thesis, TU Delft, The Netherlands, 2001

L. Haglund, *Adaptive Mulitdimensional Filtering"*, PhD Thesis, Link"oping University, Sweden, 1992

W.T. Freeman, *" Steerable Filters and Local Analysis of Image Structure"*, PhD Thesis, MIT, USA, 1992

AdaptiveGauss

Performs Gaussian filtering steered by paramter images

SYNOPSIS

```
#include "dip_adaptive.h"
dip_Error dip_AdaptiveGauss( in,out,para_images,filterSize,order,truncation )
```

DATA TYPES

sfloat

FUNCTION

This function performs Gaussian filtering steerd by the information stored in the paramter images. The meaning of the parameter images depends on the dimensionality of the input image. Up to now only 2 and 3D images are supported for adaptive filtering. If the input image is not of type **float** it is converted to that type.

`para_images`: ImageArray containing orientation images.

2D: angle of the orientation

3D: polar coordinate phi, theta for intrinsic 1D structures polar coordinates of two orientations for intrinsic 2D structures

`filterSize`: Array containing the sigmas of the derivatives.

For intrinsic 1D structures, the first value is along the contour, the second perpendicular to it.

For intrinsic 2D structures, the first two are in the plane, whereas the other is perpendicular to them. If a value is zero no convolution is done is this direction.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_ImageArray`	`para_images`	Parameter images
`dip_FloatArray`	`filterSize`	Size of the filter
`dip_IntegerArray`	`order`	Order of the Gaussian derivative
`dip_int`	`truncation`	Truncation of the Gaussian

SEE ALSO

AdaptivePercentile, AdaptiveBanana, Gauss

LITERATURE

P. Bakker, *"Image structure analysis for seismic interpretation"*, PhD Thesis, TU Delft, The Netherlands, 2001

L. Haglund, *Adaptive Mulitdimensional Filtering"*, PhD Thesis, Link"oping University, Sweden, 1992

W.T. Freeman, *" Steerable Filters and Local Analysis of Image Structure"*, PhD Thesis, MIT, USA, 1992

AdaptivePercentile

Performs Percentile filtering steered by paramter images

SYNOPSIS

```
#include "dip_adaptive.h"
dip_Error DIP_TPI_FUNC(dip_AdaptivePercentile)( in, out, para_images,
filterSize, precentile )
```

DATA TYPES

sfloat

FUNCTION

This function performs percentile filtering steerd by the information stored in the paramter images (local orientation). The meaning of the parameter images depends on the dimensionality of the input image. Up to now only 2 and 3D images are supported for adaptive filtering. If the input image is not of type **float** it is converted to that type.

`para_images`: ImageArray containing orientation images.

2D: angle of the orientation.

3D: polar coordinate phi, theta for intrinsic 1D structures polar coordinates of two orientations for intrinsic 2D structures.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_ImageArray	para_images	Parameter images
dip_FloatArray	filterSize	Size of the filter
dip_float	percentile	Percentile value

SEE ALSO

AdaptiveBanana, AdaptiveGauss, PercentileFilter, MedianFilter

LITERATURE

P. Bakker, *"Image structure analysis for seismic interpretation"*, PhD Thesis, TU Delft, The Netherlands, 2001

L. Haglund, *Adaptive Mulitdimensional Filtering"*, PhD Thesis, Link"oping University, Sweden, 1992

W.T. Freeman, *" Steerable Filters and Local Analysis of Image Structure"*, PhD Thesis, MIT, USA, 1992

Add

arithmetic function

SYNOPSIS

`dip_Error dip_Add ( in1, in2, out )`

Calls `Arith` `( in1, in2, out, DIP_ARITHOP_ADD, DIP_DT_MINIMUM )`

AddComplex

arithmetic function

SYNOPSIS

`dip_Error dip_AddComplex ( in, out, constant )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

This function computes `out` = `in` + `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_complex	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, AddInteger, AddFloat, SubComplex, MulComplex, MulConjugateComplex, DivComplex

AddFloat

arithmetic function

SYNOPSIS

`dip_Error dip_AddFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` + `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

`Arith`, `Arith_ComplexSeparated`, `AddInteger`, `AddComplex`, `SubFloat`, `MulFloat`, `DivFloat`

AddInteger

arithmetic function

SYNOPSIS

`dip_Error AddInteger ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` + `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, AddFloat, AddComplex, SubInteger, MulInteger, DivInteger

And

logic operation

SYNOPSIS

`dip_Error dip_And ( in1, in2, out )`

DATA TYPES

binary

FUNCTION

The function `And` performs the logic AND operation between the corresponding pixels in `in1` and `in2`, and stores the result in `out`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in1`	First binary input image
`dip_Image`	`in2`	Second binary input image
`dip_Image`	`out`	Output image

SEE ALSO

`Arith`, `Xor`, `Or`, `Invert`

AreaOpening

Morphological filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_AreaOpening ( grey, mask, out, filtersize, connectivity,
closing )
```

DATA TYPES

integer, **float**

FUNCTION

The image `grey` will be filtered to remove local maxima (`closing` is `DIP_FALSE`) or local minima (`closing` is `DIP_TRUE`) with an area smaller than `filtersize` (in pixels).

Theoretically, the area opening can be written as the supremum of all the openings with each of the possible compact structuring elements of `filtersize` pixels. The `connectivity` parameter indicates which shapes are considered compact (i.e. all pixels are connected). See The connectivity parameter for more information.

ARGUMENTS

Data type	Name	Description
dip_Image	grey	Grey-value input image
dip_Image	mask	Mask image for ROI processing
dip_Image	out	Output image
dip_int	filtersize	Size of structuring element
dip_int	connectivity	Connectivity
dip_Boolean	closing	DIP_FALSE for area opening, DIP_TRUE for area closing

LITERATURE

L. Vincent, Grayscale area openings and closings, their efficient implementation and applications, Mathematical Morphology and Its Applications to Signal Processing, pages 22-27, 1993.

SEE ALSO

`Opening`, `Closing`, `PathOpening`, `DirectedPathOpening`, `MorphologicalReconstruction`

Arith

arithmetic function

SYNOPSIS

`dip_Error dip_Arith ( in1, in2, out, op, dt )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in1 op in2` on a pixel by pixel basis. The data types of the `in1` and `in2` image may be of different types. `dt` may be any of DIPlib's data types, or the constants `DIP_DT_MINIMUM`, `DIP_DT_FLEX` or `DIP_DT_FLEXBIN`, and determines what the output data type will be. `DIP_DT_MINIMUM` selects a data type according to the default for dyadic operations, see Information about dyadic operations for more information. `DIP_DT_FLEX` will choose a floating point (real or complex) type, the precision depends on the input types. `DIP_DT_FLEXBIN` is the same as `DIP_DT_FLEX`, except that two binary inputs will produce a binary output.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dipf_ArithOperation	op	Dyadic arithmetic operation
dip_DataType	dt	Data type for output

The `dipf_ArithOperation` flag can be one of:

Name	Description
DIP_ARITHOP_ADD	Addition (`in1+in2`)
DIP_ARITHOP_SUB	Subtraction (`in1-in2`)
DIP_ARITHOP_MUL	Multiplication (`in1*in2`)
DIP_ARITHOP_DIV	Division (`in1/in2`)
DIP_ARITHOP_MUL_CONJUGATE	Conjugate multiplication (`in1*conj(in2)`)

For two binary inputs, and with `dt` = `DIP_DT_FLEXBIN`, the operations performed are equivalent to logical operations:

Name	Description
DIP_ARITHOP_ADD	Or (in1|in2)
DIP_ARITHOP_SUB	And not (in1&!in2)
DIP_ARITHOP_MUL	And (in1&in2)
DIP_ARITHOP_DIV	Xor (in1^in2)
DIP_ARITHOP_MUL_CONJUGATE	And (in1&in2)

SEE ALSO

Arith_ComplexSeparated, Add, Sub, Mul, Div, MulConjugate, AddInteger, AddFloat, AddComplex, SubInteger, SubFloat, SubComplex, MulInteger, MulFloat, MulComplex, MulConjugateComplex, DivInteger, DivFloat, DivComplex

Arith_ComplexSeparated

arithmetic function

SYNOPSIS

`dip_Error dip_Arith ( in1_real, in2_imag, in2_real, in2_imag, out_real, out_imag, op, dt )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = `in1 op in2` on a pixel by pixel basis. The data types of the `in1` and `in2` image may be of different types. The two input images and the output images have the complex portion of the data as a separate image, that is, `in1` = `in1_real` + i`in1_imag`. `in1_imag` and `in2_imag` may be `0`. `dt` may be any of DIPlib's data types, or the constants `DIP_DT_MINIMUM` or `DIP_DT_FLEX`, and determines what the output data type will be. `DIP_DT_MINIMUM` selects a data type according to the default for dyadic operations, see Information about dyadic operations for more information. `DIP_DT_FLEX` will choose a floating point (real or complex) type, the precision depends on the input types.

ARGUMENTS

Data type	Name	Description
dip_Image	in1_real	First input, real part
dip_Image	in1_imag	First input, imaginary part (or NULL)
dip_Image	in2_real	Second input, real part
dip_Image	in2_imag	Second input, imaginary part (or NULL)
dip_Image	out_real	Output, real part
dip_Image	out_imag	Output, imaginary part
dipf_ArithOperation	op	Dyadic arithmetic operation
dip_DataType	dt	Data type for output

The `dipf_ArithOperation` flag can be one of:

Name	Description
DIP_ARITHOP_ADD	Addition (in1+in2)
DIP_ARITHOP_SUB	Subtraction (in1-in2)
DIP_ARITHOP_MUL	Multiplication (in1*in2)
DIP_ARITHOP_DIV	Division (in1/in2)
DIP_ARITHOP_MUL_CONJUGATE	Conjugate multiplication (in1*conj(in2))

SEE ALSO

Arith, Add, Sub, Mul, MulConjugate, Div, AddInteger, AddFloat, AddComplex, SubInteger, SubFloat, SubComplex, MulInteger, MulFloat, MulComplex, MulConjugateComplex, DivInteger, DivFloat, DivComplex

ArrayFree

Array free function

SYNOPSIS

`dip_Error dip_ArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
`dip_Array *`	`array`	pointer to a dip_Array

SEE ALSO

ArrayNew, ArrayFree

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

ArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_ArrayNew ( array, size, elementSize, resources )`

FUNCTION

This functions allocates the `size` elements of a `dip_Array` and sets the size of the array to `size`. The size of each element is determined by `elementSize`.

ARGUMENTS

Data type	Name	Description
dip_Array *	array	Array
dip_int	size	Size
dip_int	elementSize	ElementSize
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ArrayNew, ArrayFree

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

Asin

trigonometric function

SYNOPSIS

`dip_Error dip_Asin ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the arc sine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Acos, Atan, Atan2, Sinh, Cosh, Tanh

Atan

trigonometric function

SYNOPSIS

`dip_Error dip_Atan ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the arc tangent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan2, Sinh, Cosh, Tanh

Atan2

arithmetic function

SYNOPSIS

`dip_Error dip_Atan2 ( in1, in2, out )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = atan2(`in1` , `in2`) on a pixel by pixel basis. The data types of the `in1` and `in2` image may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Tanh

AttenuationCorrection

Attenuation correction algorithm

SYNOPSIS

```
#include "dip_microscopy.h"
dip_Error dip_AttenuationCorrection ( in, out, fAttenuation, bAttenuation,
background, threshold, NA, refIndex, ratio, method )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function implements an attenuation correction using three different recursive attenuation correction algorithms. The RAC-DET algorithm is the most accurate one, since it takes both forward and backward attenuation into account. It is however considerably slower that the RAC-LT2 and RAC-LT1 algorithms which take only forward attenuation into account. These last two algorithms assume a constant attenuation (`background`) for pixels with an intensity lower than the `threshold`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	fAttenuation	Forward attenuation factor
dip_float	bAttenuation	Backward attenuation factor
dip_float	background	Background attenuation factor
dip_float	threshold	Background threshold
dip_float	NA	Numerical aperture
dip_float	refIndex	Refractive index
dip_float	ratio	Z/X sampling ratio
dipf_AttenuationCorrection	method	Correction method

The `dipf_AttenuationCorrection` enumaration consists of the following flags:

Name	Description
DIP_ATTENUATION_RAC_LT2	Recursive Attenuation Correction algorithm using two Light Cone convolutions
DIP_ATTENUATION_RAC_LT1	Recursive Attenuation Correction algorithm using one Light Cone convolution
DIP_ATTENUATION_RAC_DET	Recursive Attenuation Correction algorithm using Directional Extinction Tracking

LITERATURE

K.C. Strasters, H.T.M. van der Voort, J.M. Geusebroek, and A.W.M. Smeulders, *Fast attenuation correction in fluorescence confocal imaging: a recursive approach*, BioImaging, vol. 2, no. 2, 1994, 78-92.

AUTHOR

Karel Strasters, adapted to DIPlib by Geert van Kempen.

SEE ALSO

SimulatedAttenuation, ExponentialFitCorrection

BesselJ0

mathematical function

SYNOPSIS

`dip_Error dip_BesselJ0 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function J0 of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ1, BesselJN, BesselY0, BesselY1, BesselYN, LnGamma, Erf, Erfc, Sinc

BesselJ1

mathematical function

SYNOPSIS

`dip_Error dip_BesselJ1 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function J1 of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJN, BesselY0, BesselY1, BesselYN, LnGamma, Erf, Erfc, Sinc

BesselJN

mathematical function

SYNOPSIS

`dip_Error dip_BesselJN ( in, out, n )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function J of the order `n` of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	n	Order of the Bessel function

SEE ALSO

BesselJ0, BesselJ1, BesselY0, BesselY1, BesselYN, LnGamma, Erf, Erfc, Sinc

BesselY0

mathematical function

SYNOPSIS

`dip_Error dip_BesselY0 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function Y0 of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY1, BesselYN, LnGamma, Erf, Erfc, Sinc

BesselY1

mathematical function

SYNOPSIS

`dip_Error dip_BesselY1 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function Y1 of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselYN, LnGamma, Erf, Erfc, Sinc

BesselYN

mathematical function

SYNOPSIS

`dip_Error dip_BesselYN ( in, out, n )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the Bessel function Y of the order `n` of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	n	Order of the Bessel function

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselY1, LnGamma, Erf, Erfc, Sinc

BiasedSigma

Adaptive edge sharpening & contrast enhancing filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_BiasedSigma ( in, out, se, boundary, param, shape, sigma,
outputCount )
```

DATA TYPES

integer, **float**

FUNCTION

The Biased Sigma filter is an adaptive edge sharpening and contrast enhancing filter. Its operation differs from the Sigma filter by separating the pixels with intensities higher than the pixel being filtered, from the pixels with lower intensities. The output for this pixel is the average closest in value to this pixel. If `outputCount` is `DIP_TRUE`, the output values represent the number of pixels over which the average has been calculated. When `threshold` is `DIP_TRUE`, the pixel intensities are thresholded at `+/- 2 sigma`, when it is set to `DIP_FALSE`, the intensities are weighted with the Gaussian difference with the intensity of the center pixel.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape
dip_float	sigma	Sigma
dip_Boolean	outputCount	Output the Count

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use se as filter window, can be any size

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

LITERATURE

John-Sen Lee, *Digital Image Smoothing and the Sigma Filter*, Computer Vision, Graphics and Image Processing, 24, 255-269, 1983

SEE ALSO

`Sigma`, `GaussianSigma`

BinaryClosing

Binary morphological closing operation

SYNOPSIS

```
#include "dip_binary.h"
dip_Error dip_BinaryClosing ( in, out, connectivity, iterations, edge )
```

DATA TYPES

binary

FUNCTION

The connectivity parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The edge parameter specifies whether the border of the image should be treated as object (DIP_TRUE) or as background (DIP_FALSE). Additionally, you can set it to -1 for special handling: DIP_FALSE for the dilation, DIP_TRUE for the erosion; this avoids the border effect you can get in the corners of the image in some cases.

See section 9.6, "Morphology-based operations", in Fundamentals of Image Processing for a description of binary mathematical morphology operations.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	connectivity	Connectivity
dip_int	iterations	Iterations
dip_int	edge	Edge condition

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

BinaryDilation, BinaryErosion, BinaryOpening, BinaryPropagation

BinaryDilation

Binary morphological dilation operation

SYNOPSIS

```
#include "dip_binary.h"
dip_Error dip_BinaryDilation ( in, out, connectivity, iterations, edge )
```

DATA TYPES

binary

FUNCTION

The `connectivity` parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`).

See section 9.6, "Morphology-based operations", in Fundamentals of Image Processing for a description of binary mathematical morphology operations.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	connectivity	Connectivity
dip_int	iterations	Iterations
dip_Boolean	edge	Edge pixels on

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

BinaryErosion, BinaryClosing, BinaryOpening, BinaryPropagation

BinaryErosion

Binary morphological erosion operation

SYNOPSIS

```
#include "dip_binary.h"
dip_Error dip_BinaryErosion ( in, out, connectivity, iterations, edge )
```

DATA TYPES

binary

FUNCTION

The `connectivity` parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`).

See section 9.6, "Morphology-based operations", in Fundamentals of Image Processing for a description of binary mathematical morphology operations.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_int`	`connectivity`	Connectivity
`dip_int`	`iterations`	Iterations
`dip_Boolean`	`edge`	Edge condition

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

`BinaryDilation`, `BinaryClosing`, `BinaryOpening`, `BinaryPropagation`

BinaryImageToPixelTable

Convert a binary image to a pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_BinaryImageToPixelTable ( im, table, resources )
```

DATA TYPES

binary

FUNCTION

This functions converts a binary image to a newly allocated pixel table `table`.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Binary image
dip_PixelTable *	table	Pixel table
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Description of DIPlib's pixel tables

PixelTableCreateFilter, GreyValuesInPixelTable, PixelTableToBinaryImage

BinaryNoise

Generates an image disturbed by binary noise

SYNOPSIS

```
#include "dip_noise.h"
dip_Error dip_BinaryNoise ( in, out, p10, p01, random )
```

DATA TYPES

binary

FUNCTION

Generate an image disturbed by binary noise. See BinaryRandomVariable for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	p10	Probability of a one to zero transition
dip_float	p01	Probability of a zero to one transition
dip_Random *	random	Pointer to a random value structure

EXAMPLE

Get a binary noise disturbed image as follows:

```
dip_Image in, out;
dip_float p10, p01;
dip_Random random;

p10 = 0.1;
p01 = 0.2;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_BinaryNoise( in, out, p10, p01, &random ));
```

SEE ALSO

`BinaryRandomVariable`, `RandomVariable`, `RandomSeed`, `RandomSeedVector`, `UniformNoise`, `GaussianNoise`, `PoissonNoise`

BinaryOpening

Binary morphological opening operation

SYNOPSIS

```
#include "dip_binary.h"
dip_Error dip_BinaryOpening ( in, out, connectivity, iterations, edge )
```

DATA TYPES

binary

FUNCTION

The `connectivity` parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`). Additionally, you can set it to -1 for special handling: `DIP_TRUE` for the erosion, `DIP_FALSE` for the dilation; this avoids the border effect you can get in the corners of the image in some cases.

See section 9.6, “Morphology-based operations”, in Fundamentals of Image Processing for a description of binary mathematical morphology operations.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	connectivity	Connectivity
dip_int	iterations	Iterations
dip_int	edge	Edge condition

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

`BinaryDilation`, `BinaryErosion`, `BinaryClosing`, `BinaryPropagation`

BinaryPropagation

Morphological propagation of binary objects

SYNOPSIS

`#include "dip_binary.h"`

`dip_Error dip_BinaryPropagation ( seed, mask, out, connectivity, iterations, edge )`

DATA TYPES

binary

FUNCTION

The `connectivity` parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`).

See section 9.6, "Morphology-based operations", in Fundamentals of Image Processing for a description of binary mathematical morphology operations, and section 10.3, "Segmentation", for applications of binary propagation.

ARGUMENTS

Data type	Name	Description
dip_Image	seed	Input seed
dip_Image	mask	Input mask
dip_Image	out	Output
dip_int	connectivity	Connectivity
dip_int	iterations (0)	Iterations
dip_Boolean	edge	Edge condition

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

`BinaryDilation`, `BinaryErosion`, `BinaryClosing`, `BinaryOpening`, `EdgeObjectsRemove`, `GrowRegions`

BinaryRandomVariable

Binary random variable generator

SYNOPSIS

`#include "dip_noise.h"`

`dip_Error dip_BinaryRandomVariable ( random, input, p10, p01, output )`

FUNCTION

The binary random variable is generated by altering the input value, if the value of a generated random variable is higher than the `p10` probability, if `input` is `DIP_TRUE`, or higher than `p01` otherwise.

See `RandomVariable` for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_Boolean	input	Input
dip_float	p10	Probability of a one to zero transition
dip_float	p01	Probability of a zero to one transition

EXAMPLE

Get a binary random variable as follows:

```
dip_Random random;
dip_float p10, p01, value;

p10 = 0.1;
p01 = 0.2;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_BinaryRandomVariable( &random, 1, p10, p01,  &value ));
```

SEE ALSO

`RandomVariable`, `RandomSeed`, `RandomSeedVector`, `UniformRandomVariable`, `GaussianRandomVariable`, `PoissonRandomVariable`

BooleanArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_BooleanArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the boolean array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_BooleanArray *	dest	Destination array
dip_BooleanArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

BooleanArrayNew, BooleanArrayFree, BooleanArrayCopy, BooleanArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

BooleanArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_BooleanArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and “returns” its index in the array. If `found` is zero, `BooleanArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
`dip_BooleanArray`	`array`	Array to find value in
`dip_Boolean`	`value`	Value to find
`dip_int *`	`index`	Index of the found value
`dip_Boolean *`	`found`	Value found or not

SEE ALSO

BooleanArrayNew, BooleanArrayFree, BooleanArrayCopy, BooleanArrayFind

IntegerArrayFind, FloatArrayFind, ComplexArrayFind, DataTypeArrayFind, BooleanArrayFind, VoidPointerArrayFind

BooleanArrayFree

Array free function

SYNOPSIS

`dip_Error dip_BooleanArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_BooleanArray *	array	Array

SEE ALSO

VoidPointerArrayNew, VoidPointerArrayFree, VoidPointerArrayCopy, VoidPointerArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

BooleanArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_BooleanArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_BooleanArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
`dip_BooleanArray *`	`array`	Array
`dip_int`	`size`	Size
`dip_Boolean`	`value`	Initial value
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`BooleanArrayNew`, `BooleanArrayFree`, `BooleanArrayCopy`, `BooleanArrayFind`

`ArrayNew`, `IntegerArrayNew`, `FloatArrayNew`, `ComplexArrayNew`, `BoundaryArrayNew`, `FrameWorkProcessArrayNew`, `DataTypeArrayNew`, `ImageArrayNew`, `BooleanArrayNew`, `VoidPointerArrayNew`, `StringArrayNew`, `CoordinateArrayNew`

BoundaryArrayFree

Array free function

SYNOPSIS

`dip_Error dip_BoundaryArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_BoundaryArray *	array	Boundary conditions

SEE ALSO

BoundaryArrayNew, BoundaryArrayFree

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

BoundaryArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_BoundaryArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_BoundaryArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_BoundaryArray *	array	Boundary conditions
dip_int	size	Size
dip_Boundary	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

BoundaryArrayNew, BoundaryArrayFree

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

Canny

Edge detector

SYNOPSIS

```
#include "dip_detection.h"
dip_Error dip_Canny ( in, out, sigma, upper, lower )
```

DATA TYPES

Input is integer or float; output is **binary**.

FUNCTION

The Canny edge detector finds the ridges in the gradient magnitude, which correspond to the edges in the image. The gradient magnitude (see `GradientMagnitude`) is computed using Gaussian derivatives, with a sigma of `sigma` in both dimensions. The found ridges are pruned to remove the less salient edges. A threshold `t1` is computed so that the `1-upper` fraction of pixels with the highest gradient magnitude are kept. A second threshold, `t2 = t1*lower`, is selected that determines the minimal gradient magnitude expected of an edge. All edge pixels that exceed `t2`, and are in the same connected region as at least one pixel that exceeds `t1`, are selected as the output of this function (see `HysteresisThreshold`).

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	sigma	Sigma parameter for Gaussian derivatives
dip_float	lower	Lower threshold, as a fraction of upper threshold
dip_float	upper	Percentile used to compute upper threshold

LIMITATIONS

This function only works on 2D images.

Ceil

Arithmetic function

SYNOPSIS

`dip_Error dip_Ceil ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the ceil of the input image values, and outputs a signed integer typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Floor, Sign, Truncate, Fraction, NearestInt

ChainCodeArrayFree

Chain code array deallocation

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeArrayFree ( array )
```

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_ChainCodeArray *	array	Pointer to chain code array

SEE ALSO

ImageChainCode, ChainCodeNew, ChainCodeFree, ChainCodeArrayNew

ChainCodeArrayNew

Chain code array allocation

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeArrayNew ( array, size, resources )
```

FUNCTION

This function allocates the `size` elements of a `dip_ChainCodeArrayNew` and sets the size of the array to `size`.

ARGUMENTS

Data type	Name	Description
`dip_ChainCodeArray *`	`array`	Receives pointer to allocated structure
`dip_int`	`size`	Number of chains to allocate space for
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeArrayFree`

ChainCodeConvexHull

Compute convex hull from chain code

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeConvexHull ( chaincode, polygon, resources )
```

FUNCTION

(To be documented)

We're using Melkman's algorithm to determine the convex hull of a polygonal representation of the boundary encoded by `chaincode`.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Input chain code
dip_Polygon*	polygon	Output convex polygon
dip_Resources	resources	Resources tracking structure. See ResourcesNew

LITERATURE

Avraham A. Melkman, "On-line construction of the convex hull of a simple polyline," Information Processing Letters 25(1):11-12, 1987.

SEE ALSO

ImageChainCode, ConvexHullGetArea, ConvexHullGetPerimeter, ConvexHullGetFeret

ChainCodeFree

Chain code object deallocation

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeFree ( chaincode )
```

FUNCTION

Deallocates the `chaincode` object created by `ChainCodeNew`, and sets the pointer to zero.

ARGUMENTS

Data type	Name	Description
dip_ChainCode *	chaincode	Pointer to chain code

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeArrayNew`, `ChainCodeArrayFree`

ChainCodeGetChains

Chain code access function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetChains ( chaincode, chain )
```

FUNCTION

Returns a pointer to the first element of the chain. Each chain element contains a pointer to the next element. The last element has a `NULL` pointer. ChainCodeGetSize returns the number of elements in the chain. See ChainCodeNew for more information.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Chain code
dip_Chain **	chain	Receives the pointer to the first element in the chain

SEE ALSO

ImageChainCode, ChainCodeNew, ChainCodeFree, ChainCodeGetSize, ChainCodeGetStart, ChainCodeGetLabel, ChainCodeGetConnectivity

ChainCodeGetConnectivity

Chain code access function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetConnectivity ( chaincode, connectivity )
```

FUNCTION

Returns the connectivity used when extracting the boundary described in `chaincode`. `connectivity==1` indicates 4-connected neighbours, and the code uses integers 0 through 3. `connectivity==2` indicates 8-connected neighbours, and the code uses values 0 through 7. See The connectivity parameter.

ARGUMENTS

Data type	Name	Description
`dip_ChainCode`	`chaincode`	Chain code
`dip_int *`	`connectivity`	Receives the connectivity value in `chaincode`

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetSize`, `ChainCodeGetChains`, `ChainCodeGetStart`, `ChainCodeGetLabel`

ChainCodeGetFeret

Chain code measurement function

SYNOPSIS

`#include "dip_chaincode.h"`

`dip_Error dip_ChainCodeGetFeret ( chaincode, stepSize, feret )`

FUNCTION

This function measures the longest and shortest projections of the object encoded by `chaincode`. The chain code is rotated in `stepSize` degree intervals and the length of the projection on the x and y axes is computed for each orientation. The sizes of maximum and minimum projections, as well as the rotation at which they were obtained, are returned in the `feret` structure, which contains the following elements:

Data type	Name	Description
dip_float	maxDiameter	The widest projection of the object
dip_float	minDiameter	The narrowest projection of the object
dip_float	maxPerpendicular	The width of the projection perpendicular to `minDiameter`
dip_float	maxAngle	The angle of the projection for `maxDiameter`
dip_float	minAngle	The angle of the projection for `minDiameter`

`ChainCodeGetFeret` is the function formerly used by `Measure` for the `FeatureFeret` measurement. This measurement now uses `ConvexHullGetFeret` instead.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Input chain code
dip_float	stepSize	The step size, in degrees
dip_Feret *	feret	Output measurement

CREDITS

The original code on which the current implementation is based, was donated by Gerie van der Heijden.

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetLength`, `ChainCodeGetLongestRun`, `ChainCodeGetRadius`

ChainCodeGetLabel

Chain code access function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetLabel ( chaincode, label )
```

FUNCTION

Returns the label ID of the object whose boundary is described by `chaincode`.

ARGUMENTS

Data type	Name	Description
`dip_ChainCode`	`chaincode`	Chain code
`dip_int *`	`label`	Receives the label ID in `chaincode`

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetSize`, `ChainCodeGetChains`, `ChainCodeGetStart`, `ChainCodeGetConnectivity`

ChainCodeGetLength

Chain code measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetLength ( chaincode, length )
```

FUNCTION

Computes the length of the boundary encoded by `chaincode`. See `FeaturePerimeter` for a description of the algorithm. `ChainCodeGetLength` is the function used by `Measure` for the `FeaturePerimeter` measurement.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Input chain code
dip_float *	length	Output measurement

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetLongestRun`, `ChainCodeGetFeret`, `ChainCodeGetRadius`

ChainCodeGetLongestRun

Chain code measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetLongestRun ( chaincode, longestRun )
```

FUNCTION

Returns the number of pixels in the longest run of identical codes in `chaincode`. This represents the longest straight section of the boundary. `ChainCodeGetLongestRun` is the function used by `Measure` for the `FeatureLongestChaincodeRun` measurement.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Input chain code
dip_int *	longestRun	Receives the pixel count for the longest run

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetLength`, `ChainCodeGetFeret`, `ChainCodeGetRadius`

ChainCodeGetRadius

Chain code measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetRadius ( chaincode, radius )
```

FUNCTION

This function computes statistics on the radius of an object. The centre of gravity of the object's border pixels is used as the centre of the object. The distance from each border pixel to this centre is is computed. The maximum, minimum, mean and variance of these distances are returned in the `radius` structure, which contains the following elements:

Data type	Name	Description
dip_float	max	Maximum object radius
dip_float	mean	Mean object radius
dip_float	min	Minimum object radius
dip_float	var	Variance of object radius

`ChainCodeGetRadius` is the function used by `Measure` for the `FeatureRadius` measurement.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Input chain code
dip_CCRadius *	radius	Output measurement

SEE ALSO

`ImageChainCode`, `ChainCodeNew`, `ChainCodeFree`, `ChainCodeGetLength`, `ChainCodeGetLongestRun`, `ChainCodeGetFeret`

ChainCodeGetSize

Chain code access function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetSize ( chaincode, number )
```

FUNCTION

Returns the number of elements in the chain code.

Note: this is not a correct measure for the object's perimeter, use ChainCodeGetLength instead.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Chain code
dip_int *	number	Receives the number of elements in the chain.

SEE ALSO

ImageChainCode, ChainCodeNew, ChainCodeFree, ChainCodeGetChains, ChainCodeGetStart, ChainCodeGetLabel, ChainCodeGetConnectivity

ChainCodeGetStart

Chain code access function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeGetStart ( chaincode, startX, startY )
```

FUNCTION

Returns the start coordinates of the chain.

ARGUMENTS

Data type	Name	Description
dip_ChainCode	chaincode	Chain code
dip_int *	startX	Receives the start x-coordinate in chaincode
dip_int *	startY	Receives the start y-coordinate in chaincode

SEE ALSO

ImageChainCode, ChainCodeNew, ChainCodeFree, ChainCodeGetSize, ChainCodeGetChains, ChainCodeGetLabel, ChainCodeGetConnectivity

ChainCodeNew

Chain code object allocation

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ChainCodeNew ( chaincode, resources )
```

FUNCTION

Allocates an object of type dip_ChainCode. However, since its fields are private and currently there exist only read access functions, it is of little use creating such an object.

A dip_ChainCode object stores the following data:

Data type	Name	Description
dip_int	startX	Start coordinates for chain, ChainCodeGetStart
dip_int	startY	Start coordinates for chain, ChainCodeGetStart
dip_int	label	Label ID of object, ChainCodeGetLabel
dip_int	connectivity	Connectivity of chain, ChainCodeGetConnectivity
dip_int	number	Number of elements in chain, ChainCodeGetSize
dip_Chain *	chain	Pointer to first element in chain, ChainCodeGetChains

The dip_Chain structure has the following elements:

Data type	Name	Description
dip_uint8	code	Direction of step taken from previous to this pixel (Freeman code)
dip_Boolean	border	Pixel is on the border
dip_Chain *	next	Pointer to the next element in the chain

The chain parameter points to the first dip_Chain object in the chain, which points to the next through its next value. The last element in the chain has a NULL pointer.

Each chain element contains the code value (between 0 and 3 or between 0 and 7, depending on the connectivity) as well as a border value, which indicates whether the pixel is on the edge of the image or not. The border value is important because it indicates that the object is cut by the imaging window and needs to be treated differently.

The chain code for an object always has as many elements as the object has border pixels.

ARGUMENTS

Data type	Name	Description
dip_ChainCode *	chaincode	Receives pointer to allocated structure
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ImageChainCode, ChainCodeFree, ChainCodeArrayNew, ChainCodeArrayFree, ChainCodeGetSize, ChainCodeGetChains, ChainCodeGetStart, ChainCodeGetLabel, ChainCodeGetConnectivity, ChainCodeGetLength, ChainCodeGetLongestRun, ChainCodeGetFeret

ChangeDataType

Change the data type of an image

SYNOPSIS

`dip_Error dip_ChangeDataType( example, target, dataType )`

FUNCTION

Inherit all properties of the input image except the data type. The data type is explicitly specified through `dataType`. When `dataType` is zero, the data type of the output image is not modified. The example image may be either "raw" or "forged".

ARGUMENTS

Data type	Name	Description
dip_Image	example	An example image
dip_Image	target	The target image
dip_DataType	dataType	The data type

SEE ALSO

DIPlib's data types

ImageCopyProperties, ImageAssimilate, ChangeTo0d

ChangeDimensions

Changes the order of the dimensions in an image

SYNOPSIS

`dip_Error dip_ChangeDimensions( image, neworder )`

FUNCTION

Re-orders the dimensions in an image, optionally removing or adding singleton dimensions (those dimensions with size 1), without copying the data. `neworder` is a list of the dimension numbers in the new order, for example (1,0,2) will swap the first two dimensions. Setting `neworder` to 0 removes all singleton dimensions without altering the order. This is useful, for example, after calling a function such as `Maximum` to compute a maximum projection over one dimension. The output image of `Maximum` keeps the dimensionality of the input image, and thus has a singleton dimension. To add singleton dimensions, use a negative value in the `neworder` array. For example, (0,1,-1) adds a 3rd dimension of size 1 to the image.

ARGUMENTS

Data type	Name	Description
dip_Image	image	The image to modify
dip_IntegerArray	neworder	The new order of the dimensions

SEE ALSO

The image structure

`ImageGetDimensions`, `ImageSetDimensions`, `ImageGetStride`

ChangeTo0d

Make an image zero dimensional

SYNOPSIS

`dip_Error dip_ChangeTo0d( example, target, dataType )`

FUNCTION

Inherit all properties of the input image except the data type and the dimensionality. The data type is explicitly specified through `dataType`. When `dataType` is zero, the data type of the output image is not modified. The dimensionality is set to zero. The example image may be either “raw” or “forged”.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`example`	An example image
`dip_Image`	`target`	The target image
`dip_DataType`	`dataType`	The data type. See DIPlib’s data types

SEE ALSO

`ImageCopyProperties`, `ImageAssimilate`, `ChangeDataType`

ChordLength

Compute the chord lengths of the different phases

SYNOPSIS

```
#include "dip_analysis.h"
dip_Error dip_ChordLength ( object, mask, dist, probes, length, sampling )
```

DATA TYPES

binary, **integer**

FUNCTION

This function computes the chord lengths of the different phases in `object`. If `object` is a binary image, the image is a regarded as a two phase image. In case `object` is of the integer type, the image is regarded as a labeled image, with each integer value encoding a phase. Optionally a `mask` image can be provided to select which pixels in `object` should be used to compute the chord lengths. The `probes` variable specifies how many random point pairs should be drawn to compute the lengths. `Length` specifies the maximum correlation length. The correlation function can be computed using a random (`DIP_CORRELATION_ESTIMATOR_RANDOM`) or grid method (`DIP_CORRELATION_ESTIMATOR_GRID`), as specified by `sampling`.

ARGUMENTS

Data type	Name	Description
dip_Image	object	Object image
dip_Image	mask	Mask image
dip_Distribution	dist	Ouput distribution
dip_int	probes	Number of probes
dip_int	length	Maximum chord length
dipf_CorrelationEstimator	sampling	Samplings method

SEE ALSO

PairCorrelation, ProbabilisticPairCorrelation

CityBlockDistanceToPoint

Distance generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_CityBlockDistanceToPoint ( output, origin, scale )
```

DATA TYPES

Output: sfloat

FUNCTION

Computes the cityblock distance of each pixel in the output image to a point at `origin`. The coordinates of `origin` may lie outside the image. The `scale` parameter may be used to specify the relative distance between pixels in each dimension.

ARGUMENTS

Data type	Name	Description
dip_Image	output	Output Image
dip_FloatArray	origin	Origin
dip_FloatArray	scale	Relative scale of the pixel distances for each dimension

SEE ALSO

EllipticDistanceToPoint, EuclideanDistanceToPoint

Clip

Point operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_Clip ( in, out, clipLow, clipHigh, clipFlag )
```

DATA TYPES

integer, **float**

FUNCTION

Clips `in` at either the minimum value `clipLow` of the maximum value `clipHigh` or both. If the flag `DIP_CLIP_THRESHOLD_AND_RANGE` is specified, the clip bound are defined by `clipLow +/- clipHigh/2`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	clipLow	Lower clip bound value
dip_float	clipHigh	Higher clip bound value
dipf_Clip	clipFlag	Clip flag

The following `dipf_Clip` flags are defined:

Name	Description
DIP_CLIP_BOTH	clip both the lower and upper bound
DIP_CLIP_LOW	clip lower bound only
DIP_CLIP_HIGH	clip upper bound only
DIP_CLIP_THRESHOLD_AND_RANGE	use `clipLow` and `clipHigh` as threshold and range value
DIP_CLIP_LOW_AND_HIGH_BOUNDS	same as DIP_CLIP_BOTH

SEE ALSO

Threshold, RangeThreshold, ErfClip, ContrastStretch

Closing

Morphological closing operation

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Closing ( in, out, se, boundary, param, shape )
```

DATA TYPES

integer, **float**, **binary**

FUNCTION

Grey-value closing with different structuring elements.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

Opening, Dilation, Erosion

Colour2Gray

Convert ND image with colour information to a (n-1)D grayvalue image (in dipIO)

SYNOPSIS

```
#include "dipio_tools.h"
dip_Error dipio_Colour2Gray ( in, out, photometric )
```

FUNCTION

This function converts a colour image, as read by ImageReadColour, to a grayvalue intensity image. `in` is expected to contain the colour information along the last axis. `out` will be a scalar image with one less dimension than the input.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dipio_PhotometricInterpretation	photometric	Photometric interpretation

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

KNOWN BUGS

Some colourspaces are not converted correctly. R'G'B' (DIPIO_PHM_RGB_NONLINEAR), is treated like RGB. From a CIE Lab (DIPIO_PHM_CIELAB) or Luv (DIPIO_PHM_CIELUV) the luminosity channel is extracted, which is also a non-linear conversion away from the intensity. From HCV (DIPIO_PHM_HCV) and HSV (DIPIO_PHM_HSV) the value channel is extracted, which again is a non-linear conversion away from the intensity. CMYK (DIPIO_PHM_CMYK) and CMY (DIPIO_PHM_CMY) conversion is not implemented. Specifying these values will result in an error.

SEE ALSO

ImageRead, ImageReadColour, ImageReadROI

Compare

Compare grey values in two images

SYNOPSIS

`dip_Error dip_Compare ( in1, in2, out, selector )`

DATA TYPES

binary, integer, float

FUNCTION

This function can perform various pixel-by-pixel comparisons (smaller, smaller- equal, equal, not equal, greater-equal, greater) between `in1` ans `in2`. `out` contains the binary result. This is implemented with a call to `Select` whose `in3` and `in4` are set to binary true and false, respectively.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to the functionality of `Threshold`, but with more options.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dipf_Select	selector	Select flag

The `dipf_Select` flag can be one of:

Name	Description
DIP_SELECT_LESSER	<, Lesser than
DIP_SELECT_LESSER_EQUAL	<=, Lesser or equal
DIP_SELECT_NOT_EQUAL	!=, Unequal
DIP_SELECT_EQUAL	==, Equal
DIP_SELECT_GREATER_EQUAL	>=, Greater or equal
DIP_SELECT_GREATER	>, Greater

SEE ALSO

`Select`, `Threshold`, `Equal`, `Greater`, `Lesser`, `NotEqual`, `NotGreater`, `NotLesser`, `SelectValue`, `NotZero`

ComplexArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_ComplexArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the complex array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_ComplexArray *	dest	Destination array
dip_ComplexArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ComplexArrayNew, ComplexArrayFree, ComplexArrayCopy, ComplexArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

ComplexArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_ComplexArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and "returns" its index in the array. If `found` is zero, `ComplexArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
dip_ComplexArray	array	Array to find value in
dip_complex	value	Value to find
dip_int *	index	Index of the found value
dip_Boolean *	found	Value found or not

SEE ALSO

ComplexArrayNew, ComplexArrayFree, ComplexArrayCopy, ComplexArrayFind

IntegerArrayFind, FloatArrayFind, ComplexArrayFind, DataTypeArrayFind, BooleanArrayFind, VoidPointerArrayFind

ComplexArrayFree

Array free function

SYNOPSIS

`dip_Error dip_ComplexArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_ComplexArray *	array	Array

SEE ALSO

ComplexArrayNew, ComplexArrayFree, ComplexArrayCopy, ComplexArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

ComplexArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_ComplexArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_ComplexArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_ComplexArray *	array	Array
dip_int	size	Size
dip_complex	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ComplexArrayNew, ComplexArrayFree, ComplexArrayCopy, ComplexArrayFind

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

ContrastStretch

Point operation

SYNOPSIS

```
#include "dip_point.h"
```

```
dip_Error dip_ContrastStretch ( in, out, lowerBound, upperBound, outMaximum,
outMinimum, method, sigmoidSlope, sigmoidPoint, maxDecade )
```

DATA TYPES

integer, **float**

FUNCTION

`ContrastStretch` stretches the pixel values of the input image. Pixel values higher or equal to `UpperBound` are stretched to the `OutMaximum` value. A similar thing holds for `LowerBound` and `OutMinimum`. `Method` determines how pixel values are stretched. `SigmoidSlope` and `SigmoidPoint` are used by the `DIP_CST_SIGMOID` method. `MaxDecade` determines the maximum number of decades the method `DIP_CST_DECADE` will stretch (values lower than `MaxDecade` will be set to zero).

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_float`	`lowerBound`	LowerBound (%)
`dip_float`	`upperBound`	UpperBound (%)
`dip_float`	`outMax`	OutMaximum
`dip_float`	`outMin`	OutMinimum
`dipf_ContrastStretch`	`method`	Method
`dip_float`	`sigmoidSlope`	SigmoidSlope
`dip_float`	`sigmoidPoint`	SigmoidPoint
`dip_float`	`maxDecade`	MaxDecade

The following `dipf_ContrastStretch` flags are defined:

Name	Description
DIP_CST_LINEAR	linear contrast stretch
DIP_CST_SIGNED_LINEAR	linear stretch with zero at fixed value
DIP_CST_LOGARITHMIC	logarithmic contrast stretch
DIP_CST_SIGNED_LOGARITHMIC	signed logarithmic contrast stretch
DIP_CST_ERF	linear contrast stretch with erf clipping
DIP_CST_DECADE	Decade contrast stretching
DIP_CST_SIGMOID	Contrast stretched by sigmoid function
DIP_CST_CLIP	Simple clipping
DIP_CST_01	Stretching of [0,1] input values
DIP_CST_PI	Stretching of [-Pi,Pi] input values

In the explanaition of the different contrast stretch flags, the variables `input`, `output`, `inMin`, `inMax`, `outMin` and `outMax` are used. With `input` and `output` is meant the pixel being processed of respecitively the input and output image. `inMin` and `inMax` are the pixel values corresponding to the `lowerBound` and `upperBound` of the input image. `outMin` and `outMax` are parameters passed to the function `dip_ContrastStretch`.

The `DIP_CST_LINEAR` stretches the `input` in the following way:

```
scale  = (outMax - outMin) / (inMax - inMin)
output = scale * (MIN(inMax, MAX(inMin, input )) - inMin) + outMin
```

The `DIP_CST_SIGNED_LINEAR` stretches the `input` in the following way:

```
max    = MAX(inMax, ABS( inMin ));
scale  = (outMax - outMin) / (2 * max)
offset = (outMax - outMin)/ 2
output = scale * (MIN(inMax, MAX(inMin, input)) - offset) + outMin
```

The `DIP_CST_LOGARITHMIC` stretches the `input` in the following way:

```
scale  = (outMax - outMin) / log( inMax - inMin + 1)
offset = inMin - 1
output = scale * log(MIN(inMax, MAX(inMin, input)) - offset) + outMin
```

The `DIP_CST_SIGNED_LOGARITHMIC` stretches the `input` in the following way:

```
max    = MAX(inMax, ABS( inMin ))
scale  = (outMax - outMin) / (2 * log( max + 1))
offset = (outMax + outMin)/ 2
output = scale * log(MIN(inMax, MAX(inMin, input))- offset) + outMin
```

The `DIP_CST_ERF` stretches the `input` in the following way:

```
scale     = (outMax - outMin) / (inMax - inMin)
```

```
threshold = (inMax + inMin)/ 2
range     = inMax - inMin
in        = MIN(inMax, MAX(inMin, input))
out       = (range / 2) * erf( SQRT_PI * (in - threshold) / range )
output    = scale * (out + threshold ) + outMin
```

The `DIP_CST_DECADE` stretches the `input` in the following way:

```
inScale   = inMax - inMin
outScale  = outMax - outMin
in        = MIN(inMax, DIP_MAX(inMin, input))
decade    = log10(inScale / ( in - inMin + EPSILON))
if(decade < maxDecade)
   decade -= floor(decade)
   output  = outScale * (1 - decade) + outMin
else
   output  = 0
```

The `DIP_CST_SIGMOID` stretches the `input` in the following way:

```
SIGMOID(x) = x / (1. + ABS(x))
min        = SIGMOID(sigmoidSlope * inMin + sigmoidPoint)
max        = SIGMOID(sigmoidSlope * inMax + sigmoidPoint)
scale      = (outMax - outMin) /(max - min)
in         = MIN(inMax, MAX(inMin, input))
output     = scale * (SIGMOID(slope * in + point) - min) + outMin
```

The `DIP_CST_CLIP` stretches the `input` in the following way:

```
output = MIN(outMax, MAX(outMin, input))
```

The `DIP_CST_01` stretches the `input` in the following way:

```
scale  = (outMax - outMin)
output = scale * input + outMin
```

The `DIP_CST_01` stretches the `input` in the following way:

```
scale  = (outMax - outMin) / 2 * Pi
output = scale * (input + Pi) + outMin
```

SEE ALSO

See section 9.1, "Histogram-based operations", in Fundamentals of Image Processing.

`Clip`, `ErfClip`

ConvertArray

converts the data type of an array

SYNOPSIS

```
#include "dip_convert_array.h"
dip_Error DIP_TWO_FUNC(dip_ConvertArray)( in, inStride, inPlane, out,
outStride, outPlane, number )
```

FUNCTION

Converts the `in` array to the `out` array.

ARGUMENTS

Data type	Name	Description
void *	in	input array
dip_int	inStride	Stride of the input array
dip_int	inPlane	plane number in case in is a binary array
void *	out	output array
dip_int	outStride	Stride of the output array
dip_int	outPlane	plane number in case out is a binary array
dip_int	number	size of the arrays

ConvertDataType

Converts the data type of an image

SYNOPSIS

`dip_Error dip_ConvertDataType ( in, out, dataType )`

FUNCTION

Convert the data type of the input data to `dataType` and stores the result in `out`.

Conversion from a *complex* type to another (non-complex) type, is done by taking the real part.

Conversion to a *binary* type from another (non-binary) type, is done as follows; any non-zero number becomes 1, zero becomes zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_DataType	dataType	Data type. See DIPlib's data types

ConvexHullGetArea

Convex hull measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ConvexHullGetArea ( polygon, area )
```

FUNCTION

`ConvexHullGetArea` is the function used by `Measure` for the `FeatureConvexArea` measurement.

ARGUMENTS

Data type	Name	Description
dip_Polygon	polygon	Input convex polygon
dip_float*	area	Output measurement

SEE ALSO

ChainCodeConvexHull, ConvexHullGetPerimeter, ConvexHullGetFeret

ConvexHullGetFeret

Convex hull measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ConvexHullGetFeret ( polygon, feret )
```

FUNCTION

This function measures the longest and shortest projections of the object encoded by `polygon`. A “rotating calipers” algorithm finds all antipodal edges and vertices. Then the distances between each of these pairs is computed. These distances correspond to the lengths of the projection under an orientation perpendicular to the edge used. The maximum and minimum distance, as well as the corresponding orientations, are returned in the `feret` structure, which contains the following elements:

Data type	Name	Description
dip_float	maxDiameter	The widest projection of the object
dip_float	minDiameter	The narrowest projection of the object
dip_float	maxPerpendicular	The width of the projection perpendicular to `minDiameter`
dip_float	maxAngle	The angle of the projection for `maxDiameter`
dip_float	minAngle	The angle of the projection for `minDiameter`

`ConvexHullGetFeret` is the function used by `Measure` for the `FeatureFeret` measurement.

ARGUMENTS

Data type	Name	Description
dip_Polygon	polygon	Input convex polygon
dip_Feret*	feret	Output measurements

NOTE

This function is more accurate than `ChainCodeGetFeret`, given a correct polygonal representation of the convex hull of the object. Because antipodal pairs are identified, and the angle of the edges is used, this algorithm doesn't depend on an angle step size, as `ChainCodeGetFeret` does.

LITERATURE

Algorithm A3.7 in M.I. Shamos, “Computational geometry,“ Ph.D. thesis, Yale University, 1978.

SEE ALSO

`ChainCodeConvexHull`, `ConvexHullGetArea`, `ConvexHullGetPerimeter`, `ChainCodeGetFeret`

ConvexHullGetPerimeter

Convex hull measurement function

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ConvexHullGetPerimeter ( polygon, length )
```

FUNCTION

`ConvexHullGetPerimeter` is the function used by `Measure` for the `FeatureConvexPerimeter` measurement.

ARGUMENTS

Data type	Name	Description
dip_Polygon	polygon	Input convex polygon
dip_float*	length	Output measurement

SEE ALSO

`ChainCodeConvexHull`, `ConvexHullGetArea`, `ConvexHullGetFeret`, `ChainCodeGetLength`

Convolve1d

Perform a 1D convolution

SYNOPSIS

```
#include "dip_linear.h"
dip_Error DIP_TPI_FUNC(dip_Convolve1d)( in, out, filter, size, filterSize,
origin, flags, boundary )
```

DATA TYPES

integer,**float**

FUNCTION

This function performs a one-dimensional convolution of the input data with the given filter kernel. In general your filter will be centered around the origin. The origin is uniquely defined if the filter size is odd, but if the filter size is even you'll have to specify whether the origin of the filter lies to the left or the right. Words cannot possibly suffice here, so here is a small pictorial representation:

```
filter size is odd :             kernel data :     x x x x x
                                                       ^
                                                       0

filter size is even and
   DIP_CNV_LEFT is specified :   kernel data :     x x x x x x
                                                       ^
                                                       0

   DIP_CNV_RIGHT is specified :  kernel data :     x x x x x x
                                                         ^
                                                         0
```

When the filter size is even, one of the flags DIP_CNV_LEFT or DIP_CNV_RIGHT must be specified. When the filter size is odd both flags are ignored. It is also possible to specify the origin of the filter directly by using the DIP_CNV_USE_ORIGIN flag in combination with the origin parameter. Again a small pictorial representation:

```
                  0 1 2 3 4 5 6 7 8
kernel data :     x x x x x x x x x     when origin = 2
```

```
                    ^
                    0
```

when DIP_CNV_USE_ORIGIN is NOT specified origin is computed as follows :

```
filter size odd      origin = ( filterSize - 1 ) / 2
filter size even _and_
   DIP_CNV_LEFT      origin = ( filterSize / 2 ) - 1
   DIP_CNV_RIGHT     origin = filterSize / 2
```

The input data is copied to a temporary buffer, after which the input data is extended according to the boundary condition specified. You can use the flags `DIP_CNV_HAS_BORDER` to indicate that the input data already has a border. In this case you must make sure that there are enough pixels on either side of the array:

```
   on the left  :  ( ( filterSize - 1 ) - origin )  pixels
   on the right :  ( origin )  pixels
```

If `DIP_CNV_HAS_BORDER` is specified and `in` != `out` no auxiliary storage is used.

You must also specify the symmetry of the filter as follows:

```
   odd filter size    :   a  b  c  b  a      DIP_CNV_EVEN
                          a  b  c -b -a      DIP_CNV_ODD
                          a  b  c  d  e      DIP_CNV_GENERAL

   even filter size   :   a  b  c  c  b  a   DIP_CNV_EVEN
                          a  b  c -c -b -a   DIP_CNV_ODD
                          a  b  c  d  e  f   DIP_CNV_GENERAL
```

ARGUMENTS

Data type	Name	Description
`void *`	`in`	Pointer to the input data
`void *`	`out`	Pointer to the output data
`void *`	`filter`	Pointer to the filter data
`dip_int`	`size`	Size of the input data
`dip_int`	`filterSize`	Size of the filter
`dip_int`	`origin`	Origin of the filter. Only valid in conjunction with `DIP_CNV_USE_ORIGIN`
`dipf_Convolve`	`flags`	A combination of the flags described above
`dip_Boundary`	`boundary`	One of the standard boundary conditions. See Boundary conditions

SEE ALSO

General information about convolution

SeparableConvolution, SeparableFrameWork

ConvolveFT

Fourier transform–based convolution filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_ConvolveFT ( in, psf, out, inrep, psfrep, outrep )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This function convolves the `in` image with the point spread function `psf`, by multiplying their Fourier transforms. The `inrep`, `psfrep` and `outrep` specify whether the images are spatial images (`DIP_IMAGE_REPRESENTATION_SPATIAL`) or their Fourier transform. (`DIP_IMAGE_REPRESENTATION_SPECTRAL`).

`out` is cast to a real type if and only if both `in` and `psf` are real and in the spatial domain. That is, no effort is made to check for evenness of images in the Fourier domain, nor to check the values of the imaginary component of the result. To convert the output to a real-valued type, use the function `ConvertDataType`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Psf image
dip_Image	out	Output image
dipf_ImageRepresentation	inrep	Input spatial or spectral
dipf_ImageRepresentation	psfrep	PSF spatial or spectral
dipf_ImageRepresentation	outrep	Output spatial or spectral

SEE ALSO

General information about convolution

CoordinateArrayFree

Array free function

SYNOPSIS

`dip_Error dip_CoordinateArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_CoordinateArray *	array	Array

SEE ALSO

CoordinateArrayNew, CoordinateArrayFree

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

CoordinateArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_CoordinateArrayNew ( array, ndims, size, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_CoordinateArray` and sets the size of the array to `size`. Each element has `ndims` values, to store coordinates of an `ndims`-dimensional image. Each array element is initialized to 0.

ARGUMENTS

Data type	Name	Description
`dip_CoordinateArray *`	`array`	Array
`dip_int`	`ndims`	Dimensionality
`dip_int`	`size`	Size
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`CoordinateArrayNew`, `CoordinateArrayFree`

`ArrayNew`, `IntegerArrayNew`, `FloatArrayNew`, `ComplexArrayNew`, `BoundaryArrayNew`, `FrameWorkProcessArrayNew`, `DataTypeArrayNew`, `ImageArrayNew`, `BooleanArrayNew`, `VoidPointerArrayNew`, `StringArrayNew`, `CoordinateArrayNew`

CoordinateToIndex

Convert coordinate to pixel index

SYNOPSIS

```
#include "dip_coordsindx.h"
dip_Error dip_CoordinateToIndex ( coordinates, index, stride )
```

FUNCTION

This function converts a pixel coordinate to an pixel index which is specific for the image from which `stride` was obtained. `coordinages` and `stride` must have the same number of elements.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray	coordinates	Coordinate array
dip_int *	index	Pointer to pixel index
dip_IntegerArray	stride	stride array

SEE ALSO

IndexToCoordinate

Cos

trigonometric function

SYNOPSIS

`dip_Error dip_Cos ( in, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Computes the cosine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Atan2, Sinh, Cosh, Tanh

Cosh

trigonometric function

SYNOPSIS

`dip_Error dip_Cosh ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the hyperbolic cosine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Atan2, Sinh, Tanh

Crop

Remove the outer parts of an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Crop ( in, out, origin, size )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

Crop a part of the image. The requested part is selected by specifying its upper left corner (`origin`), and its size (`size`). If `in` has a different type than `out`, it will be converted to the type of `out`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input Image
`dip_Image`	`out`	Output Image
`dip_IntegerArray`	`origin`	Coordinate in `in` of the upper left corner of the section
`dip_IntegerArray`	`size`	Size of the new image

SEE ALSO

GetSlice, GetLine

CrossCorrelationFT

Normalized cross-correlation using the Fourier Transform

SYNOPSIS

```
#include "dip_findshift.h"
dip_Error dip_CrossCorrelationFT ( in1, in2, out, in1rep, in2rep, outrep )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

This function calculates the cross-correlation between two images of equal size. The returned image is the cross-correlation normalized in such a way that only the phase information is of importance. This results as a very sharp peak in the spatial domain. This function performs `out = (Conj(in1)*in2)/((Abs(in1))^2)` in the Fourier domain. It is used by FindShift. The `inrep`, `psfrep` and `outrep` specify whether the images are spatial images (`DIP_IMAGE_REPRESENTATION_SPATIAL`) or their Fourier transform. (`DIP_IMAGE_REPRESENTATION_SPECTRAL`).

ARGUMENTS

Data type	Name	Description
dip_Image	in1	Input image
dip_Image	in2	Input image
dip_Image	out	Output image
dipf_ImageRepresentation	in1rep	Input 1 spatial or spectral
dipf_ImageRepresentation	in2rep	Input 2 spatial or spectral
dipf_ImageRepresentation	outrep	Output spatial or spectral

SEE ALSO

FindShift

CumulativeSum

statistics function

SYNOPSIS

`dip_Error dip_CumulativeSum ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the cumulative sum of the pixel values over all those dimensions which are specified by `ps`, i.e.:

`out(x,y)=sum_i=0:x,j=0:y in(i,j)` when ps specifies both x and y `out(x,y)=sum_j=0:y in(x,j)` when ps specifies only y

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Mean, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

DanielsonLineDetector

Line detector

SYNOPSIS

```
#include "dip_orientation.h"
dip_Error dip_DanielsonLineDetector ( in, line, energy, angle, boundary,
sigma, truncation, flavour )
```

DATA TYPES

binary, integer, **float**

FUNCTION

The Danielson line dectector uses second derivatives to detect lines in 2D images and to estimate their orientation. See the literature reference for an in-depth information on this detector.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	line	Line image
dip_Image	energy	Energy image
dip_Image	angle	Angle image
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	sigma	Sigma of second derivatives
dip_float	truncation	Gauss Truncation, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative filter flavour

LITERATURE

P.E. Danielson, Q. Lin and Q-Z Yes, i"Efficient detection of second degree variations in 2D and 3D images", Report LiTH-ISY-R-2155, Linkoping University, Linkoping, Sweden, 1999

SEE ALSO

Derivative, StructureTensor2D

DataTypeAllowed

Check whether a data type is allowed

SYNOPSIS

`dip_Error dip_DataTypeAllowed( dataType, allow, allowedTypes, allowed )`

FUNCTION

This function checks whether the `dataType` is (or is not) in the set of data types specified by `allowedTypes`. If `allow` is `DIP_TRUE`, the data type should be in this set. If `allow` is `DIP_FALSE`, the data type should not be in this set. If the `allowed` parameter is zero, the routine returns `dip_errorDataTypeNotSupported` if the required condition is not satisfied. If nonzero, it should point to a boolean variable. This boolean variable will be set to `DIP_TRUE` if the condition is satisfied, or `DIP_FALSE` if not.

ARGUMENTS

Data type	Name	Description
`dip_DataType`	`dataType`	The data type to check
`dip_Boolean`	`allow`	`DIP_TRUE`: check if the data type is included. `DIP_FALSE`: check if the data type is not included
`dip_DataTypeProperties`	`allowedTypes`	The set of data types to check against, see `DataTypeGetInfo`
`dip_Boolean *`	`allowed`	Pointer to a boolean to store the answer, or 0 to indicate that `dip_errorDataTypeNotSupported` should be returned if the condition is not satisfied

SEE ALSO

DIPlib's data types

`DataTypeGetInfo`

DataTypeArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_DataTypeArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the data type array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_DataTypeArray *	dest	Destination array
dip_DataTypeArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

DIPlib's data types

DataTypeArrayNew, DataTypeArrayFree, DataTypeArrayCopy, DataTypeArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

DataTypeArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_DataTypeArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and "returns" its index in the array. If `found` is zero, `DataTypeArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
`dip_DataTypeArray`	`array`	Array to find value in
`dip_DataType`	`value`	Value to find
`dip_int *`	`index`	Index of the found value
`dip_Boolean *`	`found`	Value found or not

SEE ALSO

DIPlib's data types

`DataTypeArrayNew`, `DataTypeArrayFree`, `DataTypeArrayCopy`, `DataTypeArrayFind`

`IntegerArrayFind`, `FloatArrayFind`, `ComplexArrayFind`, `DataTypeArrayFind`, `BooleanArrayFind`, `VoidPointerArrayFind`

DataTypeArrayFree

Array free function

SYNOPSIS

`dip_Error dip_DataTypeArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_DataTypeArray *	array	Array

SEE ALSO

DIPlib's data types

DataTypeArrayNew, DataTypeArrayFree, DataTypeArrayCopy, DataTypeArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

DataTypeArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_DataTypeArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_DataTypeArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_DataTypeArray *	array	Array
dip_int	size	Size
dip_DataType	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

DIPlib's data types

DataTypeArrayNew, DataTypeArrayFree, DataTypeArrayCopy, DataTypeArrayFind

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

DataTypeGetInfo

Get information about a data type

SYNOPSIS

`dip_Error dip_DataTypeGetInfo( dataType, info, whatInfo )`

ARGUMENTS

Data type	Name	Description
`dip_DataType`	`dataType`	The data type to get information about
`void *`	`info`	Pointer to a variable to put the information in
`dipf_DataTypeGetInfo`	`whatInfo`	What information should be returned

FUNCTION

Get information about a data type. Depending on the `whatInfo` flag this routine will return information about the data type through the `info` parameter. A pointer must be passed to this routine which must point to a variable of the proper type to contain the information which will be returned. This pointer is passed as a void pointer through the `info` parameter. Below is a table of the flags that determine what information is returned, the type of the variable that is used to store the information in and a description of the information that is returned.

`dipf_DataTypeGetInfo`	**type**	**description**
`DIP_DT_INFO_PROPS`	`dip_DataTypeProperties`	a set of flags as shown in the table below
`DIP_DT_INFO_SIZEOF`	`dip_int`	`sizeof( data type )`
`DIP_DT_INFO_C2R`	`dip_DataType`	for complex types returns the corresponding floating point type (i.e. `dip_scomplex` -> `dip_sfloat`) for other data types returns the data type itself

The following table shows which `dip_DataTypeProperties` flags are set for which data types:

Data type identifier group	data types
DIP_DT_IS_UINT	unsigned integer
DIP_DT_IS_UNSIGNED	unsigned integer
DIP_DT_IS_SINT	signed integer
DIP_DT_IS_INT	signed and unsigned integer
DIP_DT_IS_INTEGER	signed and unsigned intege
DIP_DT_IS_FLOAT	floating-point
DIP_DT_IS_REAL	integer and floating-point
DIP_DT_IS_COMPLEX	complex floating-point
DIP_DT_IS_SIGNED	signed integer, floating-point and complex
DIP_DT_IS_BINARY	binary
DIP_DT_IS_ANY	all

SEE ALSO

DIPlib's data types

Derivative

Derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_Derivative ( in, out, boundary, ps, sigmas, order, truncation,
flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

This function provides a common interface to different families of regularised derivative operators. Which family is used, is specified by the `flavour` parameter. The order of the derivative operator along each of the cartesian axes may be specified independently.

Be sure to read the documentation on the underlying implementation to learn about the properties and limitations of the various families.

For the Gaussian family of filters, `sigmas` must be given, but `order` can be 0 (only smooth, don't take the derivative).

For the finite difference filter, `sigmas` can be 0, in which case the non-derivative dimensions will not be processed. Any element of `sigmas` that is non-zero where the corresponding `order` is zero, indicates a dimension that will be smoothed. Note it's possible to reporduce the `SobelGradient` filter this way.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	bc	Boundary conditions
dip_BooleanArray	ps (0)	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_int	order (0)	Derivative order
dip_float	truncation	Truncation, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative filter flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

See section 9.5, "Derivative-based operations", in Fundamentals of Image Processing.

`Gauss`, `GaussFT`, `GaussIIR`, `FiniteDifferenceEx`, `GradientMagnitude`, `GradientDirection2D`, `Laplace`, `SobelGradient`

Dgg

Second order derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_Dgg ( in, out, boundary, ps, sigmas, tc, flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes the second derivative in gradient direction of an image using the `Derivative` function.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_float	tc	Truncation of Gaussian, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

See section 9.5, “Derivative-based operations”, in Fundamentals of Image Processing (Dgg is called SDGD in the text).

Derivative, GradientMagnitude, GradientDirection2D, Laplace, LaplacePlusDgg, LaplaceMinDgg

Dilation

Local maximum filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Dilation ( in, out, se, boundary, param, shape )
```

DATA TYPES

integer, **float**, **binary**

FUNCTION

Grey-value dilation with different structuring elements.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

`Closing`, `Opening`, `Erosion`

dip__PixelGetFloat

Midlevel PixelIO function

SYNOPSIS

`dip_Error dip__PixelGetFloat ( vptr, type, position, stride, plane, val )`

FUNCTION

The dip__PixelGet/SetInteger and dip__PixelGet/SetFloat functions provide midlevel access to image pixel values. These functions are faster than the highlevel `Get` and `Set` functions, but are easier to use than the lowlevel `DIP_PIXEL_GET` and `DIP_PIXEL_SET` macros as defined in `dip_macros.h`.

ARGUMENTS

Data type	Name	Description
`void *`	`vptr`	Void pointer to the image data
`dip_DataType`	`type`	Image data type. See DIPlib's data types
`dip_IntegerArray`	`position`	Position of the pixel in the image
`dip_IntegerArray`	`stride`	Image data stride
`dip_int`	`plane`	Plane of the pixel (binary images)
`dip_float *`	`val`	Pointer to the variable receiving the obtained pixel value

SEE ALSO

`dip__PixelGetInteger`, `dip__PixelSetInteger`, `dip__PixelSetFloat`, `Get`, `Set`, `GetInteger`, `SetInteger`, `GetFloat`, `SetFloat`

dip__PixelGetInteger

Midlevel PixelIO function

SYNOPSIS

`dip_Error dip__PixelGetInteger ( vptr, type, position, stride, plane, val )`

FUNCTION

The dip__PixelGet/SetInteger and dip__PixelGet/SetFloat functions provide midlevel access to image pixel values. These functions are faster than the highlevel `Get` and `Set` functions, but are easier to use than the lowlevel `DIP_PIXEL_GET` and `DIP_PIXEL_SET` macros as defined in `dip_macros.h`.

ARGUMENTS

Data type	Name	Description
`void *`	`vptr`	Void pointer to the image data
`dip_DataType`	`type`	Image data type. See DIPlib's data types
`dip_IntegerArray`	`position`	Position of the pixel in the image
`dip_IntegerArray`	`stride`	Image data stride
`dip_int`	`plane`	Plane of the pixel (binary images)
`dip_int *`	`val`	Pointer to the variable receiving the obtained pixel value

SEE ALSO

`dip__PixelGetFloat`, `dip__PixelSetInteger`, `dip__PixelSetFloat`, `Get`, `Set`, `GetInteger`, `SetInteger`, `GetFloat`, `SetFloat`

dip__PixelSetFloat

Midlevel PixelIO function

SYNOPSIS

`dip_Error dip__PixelSetFloat ( val, vptr, type, position, stride, plane )`

FUNCTION

The dip__PixelGet/SetInteger and dip__PixelGet/SetFloat functions provide midlevel access to image pixel values. These functions are faster than the highlevel `Get` and `Set` functions, but are easier to use than the lowlevel `DIP_PIXEL_GET` and `DIP_PIXEL_SET` macros as defined in `dip_macros.h`.

ARGUMENTS

Data type	Name	Description
`dip_float`	`val`	Value to write to the pixel
`void *`	`vptr`	Void pointer to the image data
`dip_DataType`	`type`	Image data type. See DIPlib's data types
`dip_IntegerArray`	`position`	Position of the pixel in the image
`dip_IntegerArray`	`stride`	Image data stride
`dip_int`	`plane`	Plane of the pixel (binary images)

SEE ALSO

`dip__PixelGetInteger`, `dip__PixelGetFloat`, `dip__PixelSetInteger`, `Get`, `Set`, `GetInteger`, `SetInteger`, `GetFloat`, `SetFloat`

dip__PixelSetInteger

Midlevel PixelIO function

SYNOPSIS

`dip_Error dip__PixelSetInteger ( val, vptr, type, position, stride, plane )`

FUNCTION

The dip__PixelGet/SetInteger and dip__PixelGet/SetFloat functions provide midlevel access to image pixel values. These functions are faster than the highlevel `Get` and `Set` functions, but are easier to use than the lowlevel `DIP_PIXEL_GET` and `DIP_PIXEL_SET` macros as defined in `dip_macros.h`.

ARGUMENTS

Data type	Name	Description
dip_int	val	Value to write to the pixel
void *	vptr	Void pointer to the image data
dip_DataType	type	Image data type. See DIPlib's data types
dip_IntegerArray	position	Position of the pixel in the image
dip_IntegerArray	stride	Image data stride
dip_int	plane	Plane of the pixel (binary images)

SEE ALSO

`dip__PixelGetInteger`, `dip__PixelGetFloat`, `dip__PixelSetFloat`, `Get`, `Set`, `GetInteger`, `SetInteger`, `GetFloat`, `SetFloat`

DirectedPathOpening

Morphological filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error DirectedPathOpening ( grey, mask, out, param, closing, constrained
)
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

Theoretically, the path opening can be written as the supremum of all the openings with each of the possible linear structuring elements of composed of a set number of pixels in the general orientation of `param`. The `param` parameter is interpreted as follows: `length` is set to `max(param)`. `direction` is set to `round(param/length)`. `direction` now contains only values 0, -1 or 1. A 90 degree cone is defined around the given `direction`, and this cone gives the neighbourhood connectivity. The structuring element is formed by `length` pixels connected according to this neighbourhood connectivity. For example, in 2D, if `param` is `[10,0]`, the structuring element will be formed by 10 pixels connected either diagonally or horizontally. It will extend across exactly 10 horizontal pixels, but can vary in shape to adapt to local image content.

If `closing` is `DIP_TRUE`, the path closing will be computed instead of the opening.

If `constrained` is `DIP_TRUE`, the algorithm is modified as follows: Only one consecutive step is allowed in a direction other than the exact direction specified. For example, following the `[10,0]` example above, a diagonal step must be followed by at least one horizontal step. This avoids zig-zag lines, especially if the main direction is diagonal. It also reduces the maximal angle that a straight line can deviate from the chosen direction. The unconstrained algorithm will keep lines rotated by up to 45 degrees; the constrained algorithm limits this to 22.5 degrees.

The algorithm uses a boundary condition such that any line connected to the border is considered infinite in length. To constrain the lines to the image domain, set a **two** pixel border around the image to the minimum value, e.g. 0 (for the opening), or the maximum value, e.g. 255 (for the closing).

ARGUMENTS

Data type	Name	Description
dip_Image	grey	Grey-value input image
dip_Image	mask	Mask image for ROI processing
dip_Image	out	Output image
dip_FloatArray	param	Size of structuring element
dip_Boolean	closing	DIP_FALSE for path opening, DIP_TRUE for path closing
dip_Boolean	constrained	DIP_TRUE for constrained paths, DIP_FALSE for the original path opening algorithm

LITERATURE

H. Talbot and B. Appleton, Efficient complete and incomplete path openings and closings, Image and Vision Computing 25:416-425, 2007.

C.L. Luengo Hendriks, Constrained and dimensionality-independent path openings, IEEE Transactions on Image Processing 19(6):1587-1595, 2010.

SEE ALSO

Opening, Closing, PathOpening, AreaOpening

DistributionSort

Sort a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_DistributionSort ( data, size, dataType )
```

FUNCTION

Sorts a block of data (of size `size` and data type `dataType`) using the distribution sort algorithm.

ARGUMENTS

Data type	Name	Description
`void *`	`data`	Data
`dip_int`	`size`	Size
`dip_DataType`	`dataType`	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

`DistributionSortIndices`, `DistributionSortIndices16`, `Sort`, `ImageSort`, `SortIndices`, `SortIndices16`, `ImageSortIndices`

DistributionSortIndices

Sort indices to block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_DistributionSortIndices ( data, indices, size, dataType )
```

FUNCTION

Sorts a list of indices rather than the data itself using the distribution sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint32 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

DistributionSort, DistributionSortIndices16, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

DistributionSortIndices16

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_DistributionSortIndices16 ( data, indices, size, dataType )
```

FUNCTION

Sorts a list of (16 bit) indices rather than the data itself using the distribution sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint16 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

DistributionSort, DistributionSortIndices, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

Div

arithmetic function

SYNOPSIS

`dip_Error dip_Div ( in1, in2, out )`

Calls `Arith ( in1, in2, out, DIP_ARITHOP_DIV, DIP_DT_MINIMUM )`

DivComplex

arithmetic function

SYNOPSIS

`dip_Error dip_DivComplex ( in, out, constant )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

This function computes `out` = `in` / `constant` on a pixel by pixel basis. If `constant` is zero, `out` will be set to zero. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_complex	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, DivInteger, DivFloat, AddComplex, SubComplex, MulComplex, MulConjugateComplex

DivFloat

arithmetic function

SYNOPSIS

`dip_Error dip_DivFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` / `constant` on a pixel by pixel basis. If `constant` is zero, `out` will be set to zero. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, DivInteger, DivComplex, AddFloat, SubFloat, MulFloat

DivInteger

arithmetic function

SYNOPSIS

`dip_Error dip_DivInteger ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` / `constant` on a pixel by pixel basis. If `constant` is zero, `out` will be set to zero. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_int`	`constant`	Constant

SEE ALSO

`Arith`, `Arith_ComplexSeparated`, `DivFloat`, `DivComplex`, `AddInteger`, `SubInteger`, `MulInteger`

EdgeObjectsRemove

Remove binary edge objects

SYNOPSIS

`#include "dip_binary.h"`

`dip_Error dip_EdgeObjectsRemove ( in, out, connectivity )`

DATA TYPES

binary

FUNCTION

The function `EdgeObjectsRemove` removes those binary objects from `in` which are connected to the edges of the image. The connectivity of the objects is determined by `connectivity`. This function is a front-end to `BinaryPropagation`. It calls `BinaryPropagation` with no seed image and the edge pixels turned on. The result of the propagation is xor-ed with the input image. The `connectivity` parameter defines the metric, that is, the shape of the structuring element. 1 indicates city-block metric, or a diamond-shaped structuring element. 2 indicates chessboard metric, or a square structuring element. -1 and -2 indicate alternating connectivity and produce an octagonal structuring element. See The connectivity parameter for more information. The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`).

See section 10.3, "Segmentation", in Fundamentals of Image Processing for a description of the edge object removal operation.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Binary input image
`dip_Image`	`out`	Output
`dip_int`	`connectivity`	Pixel connectivity

KNOWN BUGS

This function is only implemented for images with a dimension up to three.

SEE ALSO

BinaryPropagation, Xor

EllipticDistanceToPoint

Distance generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_EllipticDistanceToPoint ( output, origin, scale )
```

DATA TYPES

Output: sfloat

FUNCTION

Computes the elliptic distance of each pixel in the output image to a point at `origin`. The coordinates of `origin` may lie outside the image. The `scale` parameter may be used to specify the relative distance between pixels in each dimension.

ARGUMENTS

Data type	Name	Description
dip_Image	output	Output Image
dip_FloatArray	origin	Coordinates of the Origin
dip_FloatArray	scale	Relative scale of the pixel distances for each dimension

SEE ALSO

EuclideanDistanceToPoint, CityBlockDistanceToPoint

Equal

Compare grey values in two images

SYNOPSIS

`dip_Error dip_Equal ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to "true" when corresponding pixels in `in1` and `in2` are equal. This is the same as `Compare` with the `DIP_SELECT_EQUAL` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to a functionality similar to `SelectValue`.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

`Compare`, `Threshold`, `Greater`, `Lesser`, `NotEqual`, `NotGreater`, `NotLesser`, `SelectValue`, `NotZero`

Erf

mathematical function

SYNOPSIS

`dip_Error dip_Erf ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the error function of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselY1, BesselYN, LnGamma, Erfc, Sinc

Erfc

mathematical function

SYNOPSIS

`dip_Error dip_Erfc ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the complementary error function of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselY1, BesselYN, LnGamma, Erf, Sinc

ErfClip

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_ErfClip ( in, out, threshold, range, clipFlag )
```

DATA TYPES

integer, **float**

FUNCTION

Clips `in` using the erf function at either or both the values `threshold +/- range/2`. If the flag `DIP_CLIP_LOW_AND_HIGH_BOUNDS` is specified, `threshold` and `range` are used as lower and upper bounds respectively.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_float`	`threshold`	Threshold value
`dip_float`	`range`	Range value
`dipf_Clip`	`clipFlag`	clipFlag

The following `dipf_Clip` flags are defined:

Name	Description
`DIP_CLIP_BOTH`	clip both the lower and upper bound
`DIP_CLIP_LOW`	clip lower bound only
`DIP_CLIP_HIGH`	clip upper bound only
`DIP_CLIP_THRESHOLD_AND_RANGE`	same as `DIP_CLIP_BOTH`
`DIP_CLIP_LOW_AND_HIGH_BOUNDS`	use `threshold` and `range` as lower and upper bounds

LITERATURE

L.J. van Vliet, *Grey-Scale Measurements in Multi-Dimensional Digitized Images*, Ph.D. thesis Delft University of Technology, Delft University Press, Delft, 1993

SEE ALSO

Clip, ContrastStretch

Erosion

Local minimum filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Erosion ( in, out, se, boundary, param, shape )
```

DATA TYPES

integer, **float**, **binary**

FUNCTION

Grey-value erosion with different structuring elements.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use se as filter window, can be any size

SEE ALSO

Closing, Opening, Dilation

error.h

Contains error messages

SYNOPSIS

```
#include "dip_error.h"
```

FUNCTION

Contains a lot of definitions to do with DIPlib's error mechanism. In particular, this include file contains definitions for a number of error messages. These are all of the type `extern const char *`. A list of the error sorted by category follows below:

Memory allocation

Name	Description
dip_errorCouldNotAllocateMemory	No memory could be allocated

Image creation errors

Name	Description
dip_errorImageIsLocked	Image is locked
dip_errorImageNotRaw	Image is not in the RAW state
dip_errorImageNotValid	Image is not in the VALID state
dip_errorImagesNotUnique	Image is used as an output image more than once
dip_errorImageLockInvalidKey	Cannot unlock. Wrong key

Image type errors

Name	Description
dip_errorIllegalImageType	Illegal image type
dip_errorImageTypeDoesNotExist	Image type does not exist
dip_errorImageTypeAlreadyExists	Adding image type failed. Type already exists
dip_errorImageTypeNotSupported	Image type not supported
dip_errorImageTypeHandlerMissing	No type handler for image type

Image data type errors

Name	Description
dip_errorDataTypeNotSupported	Data type not supported
dip_errorIllegalDataType	Illegal data type

Image dimension(ality) errors

Name	Description
dip_errorIllegalDimensionality	Illegal dimensionality
dip_errorDimensionalityNotSupported	Dimensionality not supported
dip_errorIllegalDimension	Illegal dimension

ErrorFree

Free a DIPlib call tree

SYNOPSIS

`void dip_ErrorFree( error )`

FUNCTION

Free a DIPlib call tree.

ARGUMENTS

Data type	Name	Description
dip_Error	error	DIPlib call tree

RETURNS

Nothing

EuclideanDistanceToPoint

Distance generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_EuclideanDistanceToPoint ( output, origin )
```

DATA TYPES

Output: sfloat

FUNCTION

Computes the Euclidean distance of each pixel in the output image to a point at `origin`. The coordinates of `origin` may lie outside the image.

ARGUMENTS

Data type	Name	Description
dip_Image	output	Output Image
dip_FloatArray	origin	Coordinates of the Origin

SEE ALSO

EllipticDistanceToPoint, CityBlockDistanceToPoint

EuclideanDistanceTransform

Euclidean distance transform

SYNOPSIS

`#include "dip_distance.h"`

`dip_Error dip_EuclideanDistanceTransform ( in, out, distance, border, method )`

DATA TYPES

binary

FUNCTION

This function computes the Euclidean distance transform of an input binary image using the vector-based method as opposed to the chamfer method. This method computes distances from the objects (binary 1's) to the nearest background (binary 0's) of `in` and stored the result in `out`. The `out` image is a `sfloat` type image.

The `distance` parameter can be used to specify anisotropic sampling densities. If it is set to zero, the sampling density is assumed to be 1.0 along all axes.

The `border` parameter specifies whether the edge of the image should be treated as objects (`border` = `DIP_TRUE`) or as background (`border` = `DIP_FALSE`).

Individual vector components of the Euclidean distance transform can be obtained with the `VectorDistanceTransform`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_FloatArray`	`distance`	Sampling distances
`dip_Boolean`	`border`	Image border type
`dipf_DistanceTransform`	`method`	Transform method

`dipf_DistanceTransform` defines the following distance transform types:

Name	Description
DIP_EDT_FAST	fastest, but most errors
DIP_EDT_TIES	slower, but fewer errors
DIP_EDT_TRUE	slow, uses lots of memory, but is "error free"
DIP_EDT_BRUTE_FORCE	gives a result from which errors are calculated for the other methods. This method is extremly slow and should only be used for testing purposes.

LITERATURE

Danielsson, P.E. (1980). *"Euclidean distance mapping."* Computer Graphics and Image Processing 14: 227-248.

Mullikin, J.C. (1992). *"The vector distance transform in two and three dimensions."* CVGIP: Graphical Models and Image Processing 54(6): 526-535.

Ragnemalm, I. (1990). *Generation of Euclidean Distance Maps*, Thesis No. 206. Licentiate thesis. Linkoing University, Sweden.

Ye, Q.Z. (1988). *"The signed Euclidean distance transform and its applications."* in Proceedings, 9th International Conference on Pattern Recognition, Rome, 495-499.

KNOWN BUGS

The `EDT_TRUE` transform type is prone to produce an internal buffer overflow when applied to larger (almost) spherical objects. It this cases use `EDT_TIES` or `EDT_BRUTE_FORCE` instead.

The option `border` = `DIP_FALSE` is not supported for `EDT_BRUTE_FORCE`.

This function supports 2 and 3-dimensional images.

AUTHOR

James C. Mullikin, adapted to DIPlib by Geert M.P. van Kempen

SEE ALSO

`VectorDistanceTransform`, `GreyWeightedDistanceTransform`

EuclideanSkeleton

binary skeleton operation

SYNOPSIS

```
#include "dip_binary.h"
dip_Error dip_EuclideanSkeleton ( in, out, endpixelCondition, edgeCondition )
```

DATA TYPES

binary

FUNCTION

This function calculates an accurate (euclidean)skeleton. It tests Hilditch conditions to preserve topology. The algorithms uses the following distance metrics:

2D

5	4-connected neighbor
7	8-connected neighbor
11	neighbors reachable with a knight's move

3D

4	6-connected neighbors
6	18-connected neighbors
7	26-connected neighbors
9	neighbors reachable with knight's move
10	(2,1,1) neighbors
12	(2,2,1) neighbors

The `edge` parameter specifies whether the border of the image should be treated as object (`DIP_TRUE`) or as background (`DIP_FALSE`). See section 9.6, "Morphology-based operations", in Fundamentals of Image Processing for a description of the skeleton operation.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Binary input image
`dip_Image`	`out`	Output image
`dip_EndpixelCondition`	`endpixelCondition`	Endpixel condition
`dip_Boolean`	`edgeCondition`	Edge condition

The `dip_EndpixelCondition` enumeration consists of the following flags:

Name	Description
DIP_ENDPIXEL_CONDITION_LOOSE_ENDS_AWAY	Loose ends are eaten away
DIP_ENDPIXEL_CONDITION_NATURAL	“natural” endpixel condition of this algorithm
DIP_ENDPIXEL_CONDITION_KEEP_WITH_ONE_NEIGHBOR	Keep endpoint if it has a neighbor
DIP_ENDPIXEL_CONDITION_KEEP_WITH_TWO_NEIGHBORS	Keep endpoint if it has two neighbors
DIP_ENDPIXEL_CONDITION_KEEP_WITH_THREE_NEIGHBORS	Keep endpoint if it has three neighbors

KNOWN BUGS

`EuclideanSkeleton` is only implemented for 2 and 3 D images.

`EuclideanSkeleton` does not process pixels in a 2-pixel border around the edge. If this is an issue, consider adding 2 pixels on each side of your image.

The function is buggy for 3D images. `DIP_ENDPIXEL_CONDITION_LOOSE_ENDS_AWAY` and `DIP_ENDPIXEL_CONDITION_KEEP_WITH_ONE_NEIGHBOR` produce the same result as `DIP_ENDPIXEL_CONDITION_KEEP_WITH_THREE_NEIGHBORS`. Both `DIP_ENDPIXEL_CONDITION_NATURAL` and `DIP_ENDPIXEL_CONDITION_KEEP_WITH_TWO_NEIGHBORS` produce resonable results under most circumstances, but don't count on it!

LITERATURE

“Improved metrics in image processing applied to the Hilditch skeleton”, B.J.H. Verwer, 9th ICPR, Rome, November 14-17, 1988.

AUTHOR

Ben Verwer, adapted to DIPlib by Geert van Kempen.

SEE ALSO

`BinaryPropagation`

Exit

Clean up before exiting

SYNOPSIS

```
dip_Error dip_Exit( void )
dip_Error dipio_Exit( void )
```

FUNCTION

Free all memory used internally by DIPlib. Call this function when you stop using DIPlib (before exiting your program).

SEE ALSO

Initialise

Exp

arithmetic function

SYNOPSIS

`dip_Error dip_Exp ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the natural exponent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp2, Exp10, Ln, Log2, Log10

Exp10

arithmetic function

SYNOPSIS

`dip_Error dip_Exp10 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the base ten exponent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp, Exp2, Ln, Log2, Log10

Exp2

arithmetic function

SYNOPSIS

`dip_Error dip_Exp2 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the base two exponent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp, Exp10, Ln, Log2, Log10

ExponentialFitCorrection

Exponential fit based attenuation correction

SYNOPSIS

```
#include "dip_microscopy.h"
dip_Error dip_ExponentialFitCorrection ( in, out, method, percentile,
fromWhere, hysteresis, varWeighted )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This routine implements a simple absorption, reflection and bleaching correction based upon the assumption that the sum of these effects result in a exponential extinction of the signal as a function of depth. Only pixels that are non-zero are taken into account. Depending upon the chosen method, the mean or a percentile of all the non-zero pixels are calculated as a function of the slice number (depth). Then an exponential function is fitted through these slice-representing values. The starting point of the fit is determined by fromWhere. The first maximum is found with point[z+1] > hysteresis * point[z]. If the mean variant is chosen one can chose to apply a variance weighting to the fit.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dipf_ExpFitData`	`method`	Data statistic to fit on
`dip_float`	`percentile`	Percentile
`dipf_ExpFitStart`	`fromWhere`	From where to start the fit
`dip_float`	`hysteresis`	First maximum hysteresis
`dip_Boolean`	`varWeighted`	Fit with variance weights

The `dipf_ExpFitData` enumaration consists of the following flags:

Name	Description
`DIP_ATTENUATION_EXP_FIT_DATA_MEAN`	Fit on the mean values
`DIP_ATTENUATION_EXP_FIT_DATA_PERCENTILE`	Fit on the specified percentile of the data

The `dipf_ExpFitStart` enumaration consists of the following flags:

Name	Description
DIP_ATTENUATION_EXP_FIT_START_FIRST_PIXEL	Start fit on first pixel
DIP_ATTENUATION_EXP_FIT_START_GLOBAL_MAXIMUM	Start fit on global maximum
DIP_ATTENUATION_EXP_FIT_START_FIRST_MAXIMUM	Start fit on first maximum

LITERATURE

K.C. Strasters, H.T.M. van der Voort, J.M. Geusebroek, and A.W.M. Smeulders, *"Fast attenuation correction in fluorescence confocal imaging: a recursive approach"*, BioImaging, vol. 2, no. 2, 1994, 78-92.

AUTHOR

Karel Strasters, adapted to DIPlib by Geert van Kempen.

SEE ALSO

AttenuationCorrection, SimulatedAttenuation

ExtendRegion

Image manipulation functions

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_ExtendRegion ( image, origin, regDims, bc, ordering, imValues )
```

FUNCTION

This functions extends a region in an image, defined by `origin` and `regDims`, with a specified boundary condition `bc`. The pixels outside the region are modified according to `bc`. `ordering` changes the order in which the dimensions are processed, set to 0 to use default process order.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Image, will be modified
dip_IntegerArray	origin	Origin of region
dip_IntegerArray	regDims	Size of region
dip_BoundaryArray	bc	Boundary conditions
dip_IntegerArray	ordering	Ordering of dimensions
dip_Image *	imValues	Unused, set to 0.

NOTE

Boundary conditions `DIP_BC_ZERO_ORDER_EXTRAPOLATE`, `DIP_BC_FIRST_ORDER_EXTRAPOLATE` and `DIP_BC_SECOND_ORDER_EXTRAPOLATE` are not supported.

FeatureAnisotropy2D

Measure the anisotropy in a labeled region

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureAnisotropy2DID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureAnisotropy2DID` returns the ID value of this measurement function, that is registered by `Initialise`.

The grey value input image should contain an orientation field. For each labeled region, a tensor is constructed at each of the region's pixels. This tensor is as follow:

```
  cos^2(phi)         cos(phi)sin(phi)
[                                      ]
  cos(phi)sin(phi)   sin^2(phi)
```

The next step is to compute a new tensor, each element computed by averaging the corresponding elements of all the individual tensors. This average tensor represents the orientation information of the region as a whole. Eigenvalue analysis of this tensor yields two eigenvalues, the largest `l0`, the smallest `l1`. The anisotropy measure is:

```
( l0 - l1 ) / ( l0 + l1 )
```

which is zero for a fully isotropic regions (i.e. one where there is no preferred orientation), and one for a fully anisotropic region (i.e. when there is a single orientation).

NOTE

This function ignores any physical dimensions passed through the `Measure` function.

SEE ALSO

`Measure`

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureBendingEnergy

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use. Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureCenter

Measure the object's center

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureCenterID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureCenterID` returns the ID value of this measurement function, that is registered by `Initialise`.

This functions measures the centre of an object by calculating the first moments of the object using the object labels as binary mask. The intensity information is not taken into account.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureChainCodeBendingEnergy

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use.

NOTE: this function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call Measure with a value of 1 for the connectivity.

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureChainCodeFunction

Measurement feature #measure function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureChainCodeFunction) ( measurement, featureID, objectID,
chaincode, iterations )
```

FUNCTION

The chaincode measure function is meant for 2-D measurement functions that only require information on the shape of the object's contour, such as FeaturePerimeter. The chaincode function is called for each object seperately, with the contour of that object stored in chaincode.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	ID of the object to be measured
dip_ChainCode	chaincode	Chaincode data structure encoding the object's contour
dip_int	iterations	Number of iterations the measure function needs to scan the data

SEE ALSO

MeasurementFeatureRegister, FeatureLineFunction, FeatureImageFunction, FeatureConvHullFunction, FeatureCompositeFunction, FeatureCreateFunction

FeatureComposeFunction

Measurement feature #compose function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureComposeFunction) ( measurement, featureID, label,
intensity, compositeFeatureID, resources )
```

FUNCTION

The `compose` function is called to obtain a list of measurement features. These features are measured before the `measure` function of a composite feature is called (`FeatureCompositeFunction`). This parameter is ignored for other measurement types. The `compose` function is called after the `create` function (`FeatureCreateFunction`).

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement feature function ID
dip_Image	label	Image with pixel intensities represending object IDs
dip_Image	intensity	Image containing corresponding intensity values
dip_IntegerArray *	compositeFeatureID	Pointer to an integer array containing the the IDs of the measurement features this function requires
dip_Resources	resources	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`MeasurementFeatureRegister`, `FeatureCompositeFunction`

FeatureCompositeFunction

Measurement feature #measure function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureCompositeFunction) ( measurement, featureID, objectID,
composite, iterations )
```

FUNCTION

The `composite` measure function is meant for features that derive their measurements from the results of other measurement functions. The measurement IDs this function is based on is obtained by calling the `compose` function (`FeatureComposeFunction`). The `composite` measure function obtains the results of these measurements through its `composite` function parameter. Use the regular measurement structure access method to read the values in this parameter (i.e. `MeasurementObjectValue`).

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	ID of the object to be measured
dip_Measurement	composite	Measurement structure containing the measurement data this function is based on
dip_int	iterations	Number of iterations the `measure` function needs to scan the data

SEE ALSO

`MeasurementFeatureRegister`, `FeatureLineFunction`, `FeatureImageFunction`, `FeatureChainCodeFunction`, `FeatureConvHullFunction`, `FeatureCreateFunction`, `FeatureComposeFunction`

FeatureConvertFunction

Measurement feature #convert function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureConvertFunction) ( in, featureID, inID, out, outID,
resources )
```

FUNCTION

The `convert` function should convert the measurement data of the feature `feaureID` for the object `inID` in the measurement `in` to object `outID` of measurement `out`. This function is called by `MeasurementFeatureConvert`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	in	Input measurement data structure
dip_int	featureID	Measurement function ID
dip_int	inID	ID of the object in `in`
dip_Measurement	out	Output measurement data structure
dip_int	outID	ID of the object in `out`
dip_Resources	resources	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`MeasurementFeatureRegister`, `MeasurementFeatureConvert`

FeatureConvexArea

Measure the area of the object's convex hull

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureConvexAreaID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureConvexAreaID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the area of the convex hull of the object. The convex hull is a polygon derived from the border pixels, and thus its area is not necessarily an integer. This function supports 2D images only.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

SEE ALSO

`Measure`, `ImageChainCode`, `ChainCodeConvexHull`, `ConvexHullGetArea`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureConvexity

Measure the object's convexity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureConvexityID ( void )
```

OUTPUT DATA TYPE

dip_float

FUNCTION

dip_FeatureConvexityID returns the ID value of this measurement function, that is registered by Initialise.

This function is a composite measurement function, that returns the ratio between FeatureSize and FeatureConvexArea. A convex object will have a convexity of 1 (or slightly smaller due to discretization issues). Convexity values smaller than 1 indicate that the object boundary has concavities or that the object has holes. This function supports 2D images only.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call Measure with a value of 1 for the connectivity.

SEE ALSO

Measure, ImageChainCode, ChainCodeConvexHull

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureConvexPerimeter

Measure the perimeter of the object's convex hull

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureConvexPerimeterID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureConvexPerimeterID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the perimeter of the convex hull of the object. This function supports 2D images only.

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

SEE ALSO

`Measure`, `ImageChainCode`, `ChainCodeConvexHull`, `ConvexHullGetPerimeter`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`,

FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureConvHullFunction

Measurement feature #measure function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureConvHullFunction) ( measurement, featureID, objectID,
convhull, iterations )
```

FUNCTION

The `convhull` measure function is meant for 2-D measurement functions that only require information on the convex hull of the object, such as `FeatureFeret`. The `convhull` function is called for each object seperately, with the convex hull of that object stored in `convhull`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	ID of the object to be measured
dip_Polygon	convhull	Polygon data structure representing the object's convex hull
dip_int	iterations	Number of iterations the `measure` function needs to scan the data

SEE ALSO

`MeasurementFeatureRegister`, `FeatureLineFunction`, `FeatureImageFunction`, `FeatureChainCodeFunction`, `FeatureCompositeFunction`, `FeatureCreateFunction`

FeatureCreateFunction

Measurement feature #create function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureCreateFunction) ( measurement, featureID, label,
intensity, physDims, params, data, resources )
```

FUNCTION

The `create` function is called to initialise the measurement function. It should allocate and initialise a memory block for internal use, assign this block to the pointer `*data`, and register it in `resources`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement feature function ID
dip_Image	label	Image with pixel intensities representing object IDs
dip_Image	intensity	Image containing corresponding intensity values
dip_PhysicalDimensions	physDims	Physical dimensions data structure
void *	params	For future expansion, is currently always NULL
void **	data	Pointer to a data block that can later be accessed using MeasurementObjectData
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureRegister, FeatureLineFunction, FeatureImageFunction, FeatureChainCodeFunction, FeatureConvHullFunction, FeatureCompositeFunction

FeatureDescriptionFree

Free a Feature Description

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionFree ( description )
```

FUNCTION

This function frees a Feature Description data structure. This is not the preferred way of freeing a Feature Description. Use the resources mechanism instead (Resources tracking structure. See ResourcesNew).

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription *	description	Feature Description to be freed

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionFunction

Measurement feature #description function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureDescriptionFunction) ( measurement, featureID,
physDims, decription, resources )
```

FUNCTION

The `description` function should return a `dip_FeatureDescription` structure containing information on the measurement function, such as its name, a short description, labels for each value measured, and units of its measurement. This function is called by `MeasurementFeatureDescription`.

The `description` structure should be allocated by this function using `FeatureDescriptionNew`, and registered in `resources`. The functions `FeatureDescriptionSetName`, `FeatureDescriptionSetDescription`, `FeatureDescriptionSetDimensionLabels` and `FeatureDescriptionSetUnits` should be used to populate the structure.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_PhysicalDimensions	physDims	Physical dimensions data structure
dip_FeatureDescription *	description	Pointer to a structure containing descriptive information of the measurement feature function
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

`MeasurementFeatureRegister`, `MeasurementFeatureDescription`, `FeatureDescriptionSetName`, `FeatureDescriptionSetDescription`, `FeatureDescriptionSetDimensionLabels`, `FeatureDescriptionSetUnits`

FeatureDescriptionGetDescription

Get the description of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionGetDescription ( description, text, resources )
```

FUNCTION

Gets the description of the feature described by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_String *	text	Description text
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionGetLabels

Get the labels of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionGetLabels ( description, labels, resources )
```

FUNCTION

Gets the labels of the data of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_StringArray *	labels	Feature Labels
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionGetName

Get the name of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionGetName ( description, name, resources )
```

FUNCTION

Gets the name of the feature described by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_String *	name	Name of the measurement feature
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionGetUnits

Get the Units of the described feature

SYNOPSIS

`#include "dip_measurement.h"`

`dip_Error dip_FeatureDescriptionGetUnits ( description, units, resources )`

FUNCTION

Gets the units of the data of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_StringArray *	units	Array of Unit texts
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionNew

Allocate a new FeatureDescription

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionNew ( description, resources )
```

FUNCTION

This function allocates a new dip_FeatureDescription data structure. A feature description contains the name, a short description of a measurement feature, as well as the labels and units of the data measured by the feature.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetDescription

Set the description of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetDescription ( description, text )
```

FUNCTION

Sets the description of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
char *	text	Description text

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetDimensionLabels

Label set convenience function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetDimensionLabels ( description,
measurement, featureID, baseLabel )
```

FUNCTION

This function set the labels of the feature, described by `description`, by adding for each label a dimension indicator to `baseLabel`. For dimensions 0 to 3, X, Y or Z is added. For dimensions higher, the numerical value of the dimension is added.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	ID of the measurement feature
char *	baseLabel	Base label

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetLabel

Set the name of a particular feature label

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetLabel ( description, number, label )
```

FUNCTION

This function sets the name of a particular label of the described feature.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_int	number	Index of the label
char *	label	Label text

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetLabels

Set the labels of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetLabels ( description, measurement,
featureID, labels, label )
```

FUNCTION

Sets the labels of the data of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
`dip_FeatureDescription`	`description`	Feature description data structure
`dip_Measurement`	`measurement`	Measurement data structure
`dip_int`	`featureID`	ID of the measurement feature
`dip_StringArray`	`labels`	Array of label describing strings, one for each label
`char *`	`label`	Single description of all feature labels

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetName

Set the name of the described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetName ( description, name )
```

FUNCTION

Sets the name of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
`dip_FeatureDescription`	`description`	Feature description data structure
`char *`	`name`	Name of the measurement feature

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetUnit

Set the units of a particular feature dimension

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetUnit ( description, number, unit )
```

FUNCTION

This function sets the name of the unit along a particular dimension of the described feature.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_int	number	Index of the dimension
char *	unit	Units text

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDescriptionSetUnits

Set the units of a described feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_FeatureDescriptionSetUnits ( description, measurement,
featureID, units, unit )
```

FUNCTION

Sets the units of the data of the feature descripted by `description`.

ARGUMENTS

Data type	Name	Description
dip_FeatureDescription	description	Feature description data structure
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	ID of the measurement feature
dip_StringArray	units	Array of Unit texts, one for each unit
char *	unit	Single text for all units

SEE ALSO

MeasurementFeatureDescription, FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

FeatureDimension

Measure the object's dimensions

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureDimensionID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureDimensionID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the length of an object along the principal axes of the label image (e.g. the length object along the X, Y & Z axes).

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureExcessKurtosis

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use. Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureFeret

Measure the object's Feret diameters

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureFeretID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureFeretID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the Feret maximum and minimum diameters of an object. The Feret diameters are found by a "rotating caliper" algorithm on the convex hull polygon, see `ConvexHullGetFeret`. This function supports 2D images only.

The values returned are:

FeretMax	The widest projection of the object
FeretMin	The narrowest projection of the object
FeretPerpMin	The width of the projection perpendicular to "FeretMin"
FeretMaxAng	The angle of the projection for "FeretMax"
FeretMinAng	The angle of the projection for "FeretMin"

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

NOTE

In DIPlib version 2.3 and earlier, this measure was computed from the chain code directly, using the function `ChainCodeGetFeret`.

SEE ALSO

`Measure`, `ImageChainCode`, `ChainCodeConvexHull`, `ConvexHullGetFeret`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureGinertia

Measure the object's inertia

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureGinertiaID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureGinertiaID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function calculates the inertia (weighted by its grey values) of an object by calculating the eigenvalues of the object's second order moments tensor. This measure only supports 2D and 3D objects. `FeatureGinteria` supports a measurement parameter (see `Measure`). If a pointer to a non-zero Boolean is supplied, this function will not only measure the eigenvalues of the second order moments tensor, but also the angles of its eigenvectors.

ARGUMENTS

Data type	Name	Description
dip_Boolean *	angles	Pointer to a Boolean specifying that "eigenangles" should be measured (not yet implemented)

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

LITERATURE

"Practical Handbook on Image Processing for Scientific Applications, chapter 16", Bernd Jahne, CRC Press, 1999.

SEE ALSO

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureGmu

Measure the object's inertia

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureGmuID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureGmuID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function calculates the inertia (weighted by its grey values) of an object by calculating the object's second order moments tensor. This measure only supports 2D and 3D objects. The output tensor is ordered as follows:

2D: xx, xy, yy

3D: xx, xy, xz, yy, yz, zz

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

LITERATURE

"Mechanics", Florian Scheck, Springer, 1999.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`,

FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureGravity

Measure the object's gravity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureGravityID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureGravityID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the point of gravity of the object, by calculating the object's first moment weighted by the intensity of each object pixel.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureImageFunction

Measurement feature #measure function

SYNOPSIS

```
#include "dip_measurement.h"
```

```
dip_Error (*dip_FeatureImageFunction) ( measurement, featureID, label,
intensity, objectID, iterations )
```

FUNCTION

The `image` measurement function is meant for measurement operation that need neighborhood or global object shape information for its operation (e.g. the `FeatureSurfaceArea` function needs to evaluate the 6 connected neighborhood of each boundary voxel). The object ID image `label` can contain values that are not present in `objectID`. These labels should be ignored.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_Image	label	Image with pixel intensities representing object IDs
dip_Image	intensity	Image containing corresponding intensity values
dip_IntegerArray	objectID	Array of objectIDs to be measured
dip_int	iterations	Number of iterations the `measure` function needs to scan the image

SEE ALSO

`MeasurementFeatureRegister`, `FeatureLineFunction`, `FeatureChainCodeFunction`, `FeatureConvHullFunction`, `FeatureCompositeFunction`, `FeatureCreateFunction`

FeatureInertia

Measure the object's inertia

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureInertiaID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureInertiaID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function calculates the inertia of an object by calculating the eigenvalues of the object's second order moments tensor. This measure only supports 2D and 3D objects. `FeatureInteria` supports a measurement parameter (see `Measure`). If a pointer to a non-zero Boolean is supplied, this function will not only measure the eigenvalues of the second order moments tensor, but also the angles of its eigenvectors.

ARGUMENTS

Data type	Name	Description
dip_Boolean *	angles	Pointer to a Boolean specifying that "eigenangles" should be measured (not yet implemented)

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

LITERATURE

"Practical Handbook on Image Processing for Scientific Applications, chapter 16", Bernd Jahne, CRC Press, 1999.

SEE ALSO

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureLineFunction

Measurement feature #measure function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error (*dip_FeatureLineFunction) ( measurement, featureID,
label,intensity, size, objectID, dim, iterations )
```

FUNCTION

The `line` measure function obtains two arrays (`label` and `intensity`) with label and intensity information of the objects to be measured. The `line` measurement function is called for every line in the image (the scan dimension is determined at run time to be optimal). Since `label` can contain more than one different label, `line` itself is responsible for storing the measurement results for the appropriate object (using, for example, `MeasurementObjectData`). The object ID array `label` can contain values that are not present in `objectID`. These labels should be ignored.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_sint32 *	label	Pointer to a list (image line) of object IDs
dip_float *	intensity	Pointer to a list of corresponding intensity values
dip_int	size	Size of the `label` and `intensity` list
dip_IntegerArray	objectID	Array of objectIDs to be measured
dip_int	dim	Dimension of the line, see `ScanFrameWork`
dip_int	iterations	Number of iterations the `measure` function needs to scan the line

SEE ALSO

`MeasurementFeatureRegister`, `FeatureImageFunction`, `FeatureChainCodeFunction`, `FeatureConvHullFunction`, `FeatureCompositeFunction`, `FeatureCreateFunction`

FeatureLongestChaincodeRun

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use.

NOTE: this function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureMass

Measure the mass of the object (sum of grey-values)

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMassID ( void )
```

OUTPUT DATA TYPE

FUNCTION

dip_FeatureMassID returns the ID value of this measurement function, that is registered by Initialise. This function is just an alias for dip_FeatureSumID.

This function measures the sum of the grey-value in the intensity image (see Measure) of pixels inside the object, and is equivalent to FeatureSum

SEE ALSO

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureMaximum

Measure the object's maximum coordinate value

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMaximumID ( void )
```

OUTPUT DATA TYPE

`dip_IntegerArray`, `dip_FloatArray`

FUNCTION

`dip_FeatureMaximumID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the maximum coordinate value of each dimension of the object.

If a `dip_PhysicalDimensions` parameter is given to `Measure`, the maximum coordinate of the object is given in physical units, and is a `dip_FloatArray` rather than a `dip_IntegerArray`.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureMaxVal

Measure the object's maximum intensity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMaxValID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureMaxValID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the maximum intensity in the intensity image (see `Measure`) of pixels inside the object.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureMean

Measure the object's mean intensity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMeanID ( void )
```

OUTPUT DATA TYPE

dip_float

FUNCTION

dip_FeatureMeanID returns the ID value of this measurement function, that is registered by Initialise.

This function measures the mean intensity in the intensity image (see Measure) of pixels inside the object.

SEE ALSO

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureMinimum

Measure the object's minimum coordinate value

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMinimumID ( void )
```

OUTPUT DATA TYPE

dip_IntegerArray, dip_FloatArray

FUNCTION

dip_FeatureMinimumID returns the ID value of this measurement function, that is registered by Initialise.

This function measures the minimum coordinate value of each dimension of the object.

If a dip_PhysicalDimensions parameter is given to Measure, the minimum coordinate of the object is given in physical units, and is a dip_FloatArray rather than a dip_IntegerArray.

SEE ALSO

Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureMinVal

Measure the object's minimum intensity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMinValID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureMinValID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the minimum intensity in the intensity image (see `Measure`) of pixels inside the object.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureMu

Measure the object's inertia

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureMuID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureMuID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function calculates the inertia of an object by calculating the object's second order moments tensor. This measure only supports 2D and 3D objects. The output tensor is ordered as follows:

2D: xx, xy, yy

3D: xx, xy, xz, yy, yz, zz

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

LITERATURE

"Mechanics", Florian Scheck, Springer, 1999.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`,

FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureOrientation2D

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use. Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureP2A

Measure the circularity of the object

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureP2AID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureP2AID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function is a composite measurement function, that uses `FeatureSize`, `FeaturePerimeter`, and `FeatureSurfaceArea` to determine the circularity of an object by calculating: 2D: `P2A = perimeter^2 / (4Pi * size)` 3D: `P2A = surface-area^1.5 / (6 Sqrt(Pi) * size)`

NOTE

This function ignores any physical dimensions passed through the `Measure` function. The units are always pixels.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeaturePerimeter

Measure the object's perimeter length

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeaturePerimeterID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeaturePerimeterID` returns the ID value of this measurement function, that is registered by `Initialise`.

This measures the perimeter of 2D objects by calculating the length of the chain code of its enclosing border. This function assumes that each object has a single connected border. The used method for measuring the length of the chain code is optimal for circles, and for a collection of objects that are randomly oriented, see the referenced literature for details. This function supports 2D images only.

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

LITERATURE

A.M. Vossepoel and A.W.M. Smeulders (1982), *"Vector Code Probability and Metrication Error in the Representation of Straight Lines of Finite Length"*, Computer Graphics and Image Processing 20: 347-364

SEE ALSO

`Measure`, `ImageChainCode`, `ChainCodeGetLength`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureRadius

Measure the object's radius statistics

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureRadiusID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureRadiusID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function obtains various statistics from the distance of each boundary pixel to the object's centre. The centre of the object is obtained from the centre of gravity of the border pixels only. See `ChainCodeGetRadius` for more information. This function supports 2D images only.

The values returned are:

RadiusMax	The maximum radius
RadiusMean	The average radius
RadiusMin	The minimum radius
RadiusStD	The standard deviation of the radii

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

SEE ALSO

Measure, ImageChainCode, ChainCodeGetRadius

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureShape

Measure shape parameters of the object

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureShapeID ( void )
```

OUTPUT DATA TYPE

`dip_FloatArray`

FUNCTION

`dip_FeatureShapeID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function is a composite measurement function, that uses `FeatureSize`, `FeaturePerimeter`, and `FeatureFeret` to measure the following shape characteristics of 2D objects:

Squarity	area / (s * sp)
Circularity	area / (Pi/4 * sp^2)
Triangularity	area / (1/2 * s * sp)
Ellipticity	area / (Pi/4 * s * sp)
Elongation	p / l

with `area` the size, `s` the shortest Feret diameter, `l` the longest Feret diameter, `sp` the Feret diameter perpendicular to `s`, and `p` the perimeter of the object. The values in the output array are given in this order.

When the measured object is either a perfect square, circle, triangle or ellipse, the values obtained by `FeatureShape` will be 1.0.

NOTE

This function assumes isotropic sampling, even if the physical dimensions given through the `Measure` function say otherwise.

NOTE

This function uses chain codes. It expects each measured object to be compact, that is, to have only one chain code. Additional chain codes are ignored, meaning that non-compact objects are not measured properly. Take care in providing the correct connectivity value: if you object is compact only with 2-connectivity, this measure will fail if you call `Measure` with a value of 1 for the connectivity.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureSize

Measure the object's area/volume

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureSizeID ( void )
```

OUTPUT DATA TYPE

`dip_int`, `dip_float`

FUNCTION

`dip_FeatureSizeID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the object's size by counting the number of pixels having the same object ID. This measure is the optimal procedure for estimating the area (2D) or volume (3D) of an object with an arbitrary size. The measurement value's unit are in pixels (pixels^2 in 2D, pixels^3 in 3D).

If a `dip_PhysicalDimensions` parameter is given to `Measure`, the size of the object is given in physical units, and is a `dip_float` rather than a `dip_int`.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureSkewness

Undocumented measurement function

FUNCTION

This measurement function is undocumented and not meant for public use. Measure

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureStdDev

Measure the standard deviation of the object's intensity

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureStdDevID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureStdDevID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the standard deviation of the intensity in the intensity image (see `Measure`) of pixels inside the object.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureSum

Measure the sum of the grey values of the object

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureSumID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureSumID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the sum of the grey-value in the intensity image (see `Measure`) of pixels inside the object.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, `FeatureP2A`, `FeaturePerimeter`, `FeatureRadius`, `FeatureShape`, `FeatureSize`, `FeatureSkewness`, `FeatureStdDev`, `FeatureSum`, `FeatureSurfaceArea`

FeatureSurfaceArea

Measure the area of the object's surface

SYNOPSIS

```
#include "dip_measurement.h"
dip_int dip_FeatureSurfaceAreaID ( void )
```

OUTPUT DATA TYPE

`dip_float`

FUNCTION

`dip_FeatureSurfaceAreaID` returns the ID value of this measurement function, that is registered by `Initialise`.

This function measures the area of a 3D object's surface using six-connected boundary voxels.

NOTE

If any physical dimensions are passed to this function through `Measure`, only the sample distance along the first dimension is used. All other dimensions are assumed to be sampled the same way. This produces incorrect results for anisotropically sampled images.

LITERATURE

J.C. Mullikin and P.W. Verbeek (1993), *"Surface area estimation of digitized planes."*, bioimaging 1(1): 6-16.

SEE ALSO

`Measure`

`FeatureAnisotropy2D`, `FeatureBendingEnergy`, `FeatureCenter`, `FeatureChainCodeBendingEnergy`, `FeatureConvexArea`, `FeatureConvexPerimeter`, `FeatureConvexity`, `FeatureDimension`, `FeatureExcessKurtosis`, `FeatureFeret`, `FeatureGinertia`, `FeatureGmu`, `FeatureGravity`, `FeatureInertia`, `FeatureLongestChaincodeRun`, `FeatureMass`, `FeatureMaxVal`, `FeatureMaximum`, `FeatureMean`, `FeatureMinVal`, `FeatureMinimum`, `FeatureMu`, `FeatureOrientation2D`, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

FeatureValueFunction

Measurement feature #value function

SYNOPSIS

```
#include "dip_measurement.h"
```

```
dip_Error (*dip_FeatureValueFunction) ( measurement, featureID, objectID,
physDims, data, format, resources )
```

FUNCTION

The `value` function should return the measurement values produced by the measurement function, for one specific object. This function is called by `MeasurementObjectValue`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	ID of the object to be measured
dip_PhysicalDimensions	physDims	Physical dimensions data structure
void **	data	Pointer to a measurement-specific internal data block
dipf_MeasurementValueFormat *	format	Pointer to a data format label, See MeasurementObjectValue
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

`MeasurementFeatureRegister`, `MeasurementObjectValue`

FillBoundaryArray

Fill the border of array according to the boundary condition

SYNOPSIS

```
dip_Error DIP_TPI_FUNC(dip_FillBoundaryArray)( in, out, size, border, boundary
)
```

FUNCTION

Set the values of the border pixels of an array. The pixels of `out` outside the range of the array `in` are set to a value determined by the boundary condition and the pixel values of `in`.

ARGUMENTS

Data type	Name	Description
void *	in	input array
void *	out	output array
dip_int	size	size of input array
dip_int	border	size of the extended borders
dip_Boundary	boundary	Boundary conditions

NOTE

The `out` array has to be allocated before this function is called, and should at least has the size of `(size + 2 * border)`. Thus, `border` specifies the length of the border on both sides of the `in` array. Furthermore, the `out` pointer should point to that element in the `out` array that corresponds to the first element in the `in` array:

```
                       <-        size        ->
input:                 ************************
                       |
                       in

           <- border -><-        size        -><- border ->
output:    ------------************************------------
                       |
                       out
```

The enumerator `dip_boundary` contains the following constants:

Name	Description
DIP_BC_SYM_MIRROR	Symmetric mirroring
DIP_BC_ASYM_MIRROR	Asymmetric mirroring
DIP_BC_PERIODIC	Periodic copying
DIP_BC_ASYM_PERIODIC	Asymmetric periodic copying
DIP_BC_ADD_ZEROS	Extending the image with zeros
DIP_BC_ADD_MAX_VALUE	Extending the image with +infinity
DIP_BC_ADD_MIN_VALUE	Extending the image with -infinity

SEE ALSO

SeparableFrameWork

FindShift

Estimate the shift between images

SYNOPSIS

`#include "dip_findshift.h"`

`dip_Error dip_FindShift ( in1, in2, out, method, parameter )`

DATA TYPES

binary, integer, float, complex

FUNCTION

This function estimates the (sub-pixel) global shift between `in1` and `in2`. The numbers found represent the shift of `in1` with respect to `in2`, or the position of the first pixel of `in2` in the coordinate system of `in1`. There are two methods that can be used: CPF and MTS. Both methods require that the shift be small. Therefore, first the integer pixel is calculated, and both images are cropped to the common part.

If `method` is 0, `DIP_FSM_MTS` is used. `method` can also be `DIP_FSM_INTEGER_ONLY`. Integer shifts can be calculated for images of any dimensionality.

CPF

The CPF method (marked as FFTS in the literature below) uses the phase of the cross-correlation (as calculated by `CrossCorrelationFT`) to estimate the shift. `parameter` sets the amount of frequencies used in this estimation. The maximum value that makes sense is sqrt(1/2). Any larger value will give the same result. Choose smaller values to ignore the higher frequencies, which have a smaller SNR and are more affected by aliasing. If `parameter` is set to 0, the optimal found for images sub-sampled by a factor four will be used (parameter = 0.2).

This method only supports 2-D images (until further notice).

MTS

The MTS method (marked as GRS in the literature below) uses a first order Taylor approximation of the equation `in1(t) = in2(t-s)` at scale `parameter`. Setting `parameter` to zero, a scale of 1 will be used. This means that the images will be smoothed with a Gaussian kernel of 1. This is the more accurate one of the two methods, and therefore is the default.

This method supports images with a dimensionality between 1 and 3.

ITER

The ITER method is an iterative version of the MTS method. It is known that a single gradient based shift estimation have bias due to truncation of the Taylor expansion series (see Pham et.al.) The bias can be expressed as a polynomial of the subpixel displacements. As a result, if Taylor method is applied iteratively and the shift is refined after each iteration, the bias eventually become negligible. By using just 3 iterations, it is possible to correct bias that results in high precision O(1e-6).

PROJ

The PROJ method compute shift in each dimension from images' projections. It is fast and fairly accurate for high SNR. Should not be used for low SNR

ARGUMENTS

Data type	**Name**	**Description**
`dip_Image`	`in1`	Input image
`dip_Image`	`in2`	Input image
`dip_FloatArray`	`out`	Estimated shift
`dipf_FindShiftMethod`	`method`	Estimation method
`dip_float`	`parameter`	Parameter

The `dipf_FindShiftMethod` enumeration consists of the following flags:

Name	**Description**
`DIP_FSM_DEFAULT`	Default method (MTS)
`DIP_FSM_INTEGER_ONLY`	Find only integer shift
`DIP_FSM_CPF`	Use cross-correlation method
`DIP_FSM_FFTS`	Same
`DIP_FSM_MTS`	Use Taylor series method
`DIP_FSM_GRS`	Same

LITERATURE

C.L. Luengo Hendriks, *Improved Resolution in Infrared Imaging Using Randomly Shifted Images*, M.Sc. Thesis, Delft University of Technology, 1998 T.Q. Pham, M. Bezuijen, L.J. van Vliet, K. Schutte, C.L. Luengo Hendriks, Performance of Optimal Registration Estimators, In *Proc. of SPIE 5817 - Visual Information Processing XIV*, Defense and Security Symposium, Orlando, 2005

SEE ALSO

`CrossCorrelationFT`

FiniteDifference

A linear gradient filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_FiniteDifference ( in, out, boundary, processDim, filter )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

The FiniteDifference filter implements several basic one dimensional FIR convolution filters. The dimension in which the operation is to be performed is specified by `processDim`. The operation itself is selected with `filter`. The (1 0 -1)/2, (1 -1 0) & (0 1 -1) are difference filters that approximate a first order derivative, the (1 -2 1) filter approximates a second order derivative operation. The triangular (1 2 1)/4 filter is a local smoothing filter.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_int`	`processDim`	ProcessDim
`dipf_FiniteDifference`	`filter`	Filter selection

The `dipf_FiniteDifference` enumeration consists of the following flags:

Name	Description
`DIP_FINITE_DIFFERENCE_M101`	out[ii] = (in[ii+1] - in[ii-1])/2
`DIP_FINITE_DIFFERENCE_0M11`	out[ii] = in[ii+1] - in[ii]
`DIP_FINITE_DIFFERENCE_M110`	out[ii] = in[ii] - in[ii-1]
`DIP_FINITE_DIFFERENCE_1M21`	out[ii] = in[ii-1] - 2*in[ii] + in[ii+1]
`DIP_FINITE_DIFFERENCE_121`	out[ii] = (in[ii-1] + 2*in[ii] + in[ii+1])/4

SEE ALSO

General information about convolution

FiniteDifferenceEx, SobelGradient, Uniform, Gauss, SeparableConvolution, Convolve1d, Derivative

FiniteDifferenceEx

A linear gradient filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_FiniteDifferenceEx ( in, out, boundary, process, parOrder,
smoothflag )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

The FiniteDifferenceEx filter implements several basic one dimensional FIR convolution filters. The difference between this function and `FiniteDifference` is that this one has an interface more similar to `Gauss` and `Derivative`: it can process different derivatives along different dimensions at the same time. The first derivative is a convolution with (1 0 -1)/2, and the second derivative is a convolution with (1 -2 1). When `parOrder` is 0 for a dimension, either the triangular smoothing filter (1 2 1)/4 is applied (`smoothflag` set to `DIP_TRUE`), or the dimension is not processed at all (`smoothflag` set to `DIP_FALSE`).

Setting all `process` to `DIP_TRUE`, all `parOrder` to 0 except one dimension to 1, and `smoothflag` to `DIP_TRUE` yields the `SobelGradient`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	process	Dimensions to process
dip_IntegerArray	parOrder	Order of Derivative along each dimension
dip_Boolean	smoothflag	Whether or not to smooth in the non-derivative directions

SEE ALSO

General information about convolution

FiniteDifference, SobelGradient, Uniform, Gauss, Derivative

FloatArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_FloatArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the float array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_FloatArray *	dest	Destination array
dip_FloatArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

FloatArrayNew, FloatArrayFree, FloatArrayCopy, FloatArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

FloatArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_FloatArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and "returns" its index in the array. If `found` is zero, `FloatArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
dip_FloatArray	array	Array to find value in
dip_float	value	Value to find
dip_int *	index	Index of the found value
dip_Boolean *	found	Value found or not

SEE ALSO

FloatArrayNew, FloatArrayFree, FloatArrayCopy, FloatArrayFind

IntegerArrayFind, FloatArrayFind, ComplexArrayFind, DataTypeArrayFind, BooleanArrayFind, VoidPointerArrayFind

FloatArrayFree

Array free function

SYNOPSIS

`dip_Error dip_FloatArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_FloatArray *	array	Array

SEE ALSO

FloatArrayNew, FloatArrayFree, FloatArrayCopy, FloatArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

FloatArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_FloatArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_FloatArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_FloatArray *	array	Array
dip_int	size	Size
dip_float	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

FloatArrayNew, FloatArrayFree, FloatArrayCopy, FloatArrayFind

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

Floor

Arithmetic function

SYNOPSIS

`dip_Error dip_Floor ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the floor of the input image values, and outputs a signed integer typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Ceil, Sign, Truncate, Fraction, NearestInt

FourierTransform

Computes the Fourier transform

SYNOPSIS

```
#include "dip_transform.h"
dip_Error dip_FourierTransform ( in, out, trFlags, process, theFuture )
```

DATA TYPES

binary, integer, float, **complex**

FUNCTION

Performs a Fourier transform on `in` and places the result in `out`.

Normalisation: 1/sqrt(dimension) for each dimension.

Defaults: `process` may be zero, indicating that all dimensions should be processed.

Sampling in Fourier Domain (FD): Let one pixel in the spatial domain (SD) be Delta_SD [m], then one pixel in the FD is Delta_FD = 1/(Delta_SD * N) [m^-1], where N is the width of the image in pixels. As a consequence the maximal frequency in the FD image is N/2 * 1/(Delta_SD * N) = 1/(2 * Delta_SD) [m^-1] and is thus independent of the image width N and only related to the Nyquist frequency. The frequency of one FD pixel is therefore related to the image width N.

Note: In consequence of the above the FD resolution will not be isotropic if the image size are not square.

Note: Spatial zero-padding of the image increases the FD resolution only apparently (empty magnification).

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dipf_FourierTransform`	`trFlags`	Transform flags
`dip_BooleanArray`	`process (0)`	Dimensions to process
`void *`	`theFuture`	For future use, should be set to zero

The `dipf_FourierTransform` enumeration consists of the following flags:

Name	**Description**
DIP_TR_FORWARD	Forward transformation
DIP_TR_INVERSE	Inverse transformation

SEE ALSO

HartleyTransform

Fraction

Arithmetic function

SYNOPSIS

`dip_Error dip_Fraction ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the fraction of the input image values, and outputs a float typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Ceil, Floor, Sign, Truncate, NearestInt

FrameWorkProcessArrayFree

Array free function

SYNOPSIS

`dip_Error dip_FrameWorkProcessArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_FrameWorkProcessArray *	array	Array

SEE ALSO

FrameWorkProcessArrayNew, FrameWorkProcessArrayFree

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

FrameWorkProcessArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_FrameWorkProcessArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_FrameWorkProcessArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_FrameWorkProcessArray *	array	Array
dip_int	size	Size
dip_FrameWorkProcess	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

FrameWorkProcessArrayNew, FrameWorkProcessArrayFree

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

FTBox

Generates the Fourier transform of a box

SYNOPSIS

`#include "dip_generation.h"`

`dip_Error dip_FTBox ( image, length, scale, amplitude )`

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of a box with the half length of its sides specified by `length` and `scale`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output Image
dip_float	length	Length
dip_FloatArray	scale	Scale
dip_float	amplitude	Amplitude

SEE ALSO

FTEllipsoid, FTSphere, FTCube, FTCross, FTGaussian

FTCross

Generates the Fourier transform of a cross

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_FTCross ( image, length, scale, amplitude )
```

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of a cross with the length of its sides specified by `length` and `radius`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output Image
dip_float	length	Length of the cross' axes
dip_FloatArray	scale	Scale
dip_float	amplitude	Amplitude

SEE ALSO

FTEllipsoid, FTSphere, FTBox, FTCube, FTGaussian

FTCube

Generates the Fourier transform of a cube

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_FTCube ( image, length, amplitude )
```

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of a cube with the length of its sides equal to two times `length`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output Image
dip_float	length	Length
dip_float	amplitude	Amplitude

SEE ALSO

FTEllipsoid, FTSphere, FTBox, FTCross, FTGaussian

FTEllipsoid

Generates Fourier transform of a ellipsoid

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_FTEllipsoid ( image, radius, scale, amplitude )
```

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of an ellipsoid with the length of its axes specified by `radius` and `scale`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output Image
dip_float	radius	Radius
dip_FloatArray	scale	Scale
dip_float	amplitude	Amplitude

LITERATURE

L.J. van Vliet, *Grey-Scale Measurements in Multi-Dimensional Digitized Images*, Ph.D. thesis Delft University of Technology, Delft University Press, Delft, 1993

KNOWN BUGS

This function is only implemented for images with a dimensionality up to three.

SEE ALSO

FTSphere, FTBox, FTCube, FTCross, FTGaussian

FTGaussian

Generates the Fourier transform of a Gaussian

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_FTGaussian ( output, sigma, volume, cutoff )
```

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of a Gaussian with sigma's `sigma`. (The Fourier transform of a Gaussian, is a Gaussian.) `volume` is the integral of the Gaussian in the spatial domain. The `cutoff` variable can be used to avoid the calculation of the exponent of large negative values, which is can be very time consuming. Values of the exponent that are below `cutoff` yield a 0 value for the exponent. When `cutoff` is set to 0 or a positive value, `DIP_GENERATION_EXP_CUTOFF` is used (it is defined as -50).

ARGUMENTS

Data type	Name	Description
dip_Image	output	Output Image
dip_FloatArray	sigma	Sigma of the Gaussian
dip_float	volume	Total intensity of the Gaussian
dip_float	cutoff	Cutoff value for the exponent

SEE ALSO

FTEllipsoid, FTSphere, FTBox, FTCube, FTCross

FTSphere

Generated Fourier transform of a sphere

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_FTSphere ( image, radius, amplitude )
```

DATA TYPES

Output: sfloat, scomplex

FUNCTION

Generates the Fourier transform of a sphere with radius `radius` and an amplitude of `amplitude`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output Image
dip_float	radius	Radius
dip_float	amplitude	Amplitude

KNOWN BUGS

This function is only implemented for images with a dimensionality up to three.

SEE ALSO

FTEllipsoid, FTBox, FTCube, FTCross, FTGaussian

GaborIIR

Infinite impulse response filter

SYNOPSIS

```
#include "dip_iir.h"
dip_Error dip_GaborIIR ( in, out, boundary, ps, sigmas, frequencies, order,
truncation )
```

DATA TYPES

binary, integer, **float**

FUNCTION

Recursive infinite impulse response implementation of the Gabor filter.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_FloatArray	frequencies	frequencies
dip_IntegerArray	order	order
dip_float	truncation	Truncation, see GlobalGaussianTruncationGet

SEE ALSO

GaussIIR

Gauss

Gaussian Filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_Gauss ( in, out, boundary, process, sigmas, order, truncation )
```

DATA TYPES

binary, integer, **float**

FUNCTION

Finite impulse response implementation of a Gaussian convolution filter and Gaussian derivative convolution filters.

The Gaussian kernel is cut off at `truncation` times the sigma of the filter (in each dimension). The sum of the Gaussian's coefficients is normalised to one. A `truncation` of zero or less indicates that the global preferred truncation ought to be used, see `GlobalGaussianTruncationGet`. For the derivatives, the truncation value is increased slightly: the actual value for `truncation` used is `truncation + 0.5*order`. The minimum filter size is 3 pixels, or 5 pixels for the 3rd order derivative.

Both the `process` and the `order` parameter may be zero. If `process` is zero all dimensions are processed. If `order` is zero no derivatives are taken.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	process (0)	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_IntegerArray	order (0)	Order of Derivative along each dimension
dip_float	truncation	Truncation of Gaussian

LIMITATIONS

The order of the derivative is limited to the interval 0-3. Sigmas considerably smaller than 1.0 will yield nonsensical results.

SEE ALSO

See sections 9.4, "Smoothing operations", and 9.5, "Derivative-based operations", in Fundamentals of Image Processing.

General information about convolution

`GaussFT`, `GaussIIR`, `Derivative`, `GlobalGaussianTruncationGet`

GaussFT

Gaussian Filter through the Fourier Domain

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_GaussFT ( in, out, sigmas, order, truncation )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Fourier Domain implementation of a Gaussian convolution filter and Gaussian derivative convolution filters. The Gaussian kernel in the Fourier Domain is cut off at the equivalent of `truncation` times `sigmas`. If `truncation` is smaller or equal to 0, it is cut off where the argument to `exp` is smaller than -50, as in `FTGaussian`.

The `order` parameter may be zero, in which case no derivatives are taken.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_FloatArray	sigmas	Sigma of Gaussian
dip_IntegerArray	order (0)	Order of Derivative along each dimension
dip_float	truncation	Truncation of Gaussian kernel, see `GlobalGaussianTruncationGet`

SEE ALSO

See sections 9.4, “Smoothing operations”, and 9.5, “Derivative-based operations”, in Fundamentals of Image Processing.

General information about convolution

`Gauss`, `GaussIIR`, `Derivative`

GaussianNoise

Generate an image disturbed by Gaussian noise

SYNOPSIS

`#include "dip_noise.h"`

`dip_Error dip_GaussianNoise ( in, out, variance, random )`

DATA TYPES

integer, **float**

FUNCTION

Generate an image disturbed by additive Gaussian noise. See GaussianRandomVariable for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	variance	Variance of the Gaussian distribution the noise is drawn from
dip_Random *	random	Pointer to a random value structure

EXAMPLE

Get a image with additive Gaussian noise as follows:

```
dip_Image in, out;
dip_float variance;
dip_Random random;

variance = 1.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_GaussianNoise( in, out, variance, &random ));
```

SEE ALSO

GaussianRandomVariable, RandomVariable, RandomSeed, RandomSeedVector, UniformNoise, PoissonNoise, BinaryNoise

GaussianRandomVariable

Gaussian random variable generator

SYNOPSIS

`#include "dip_noise.h"`

`dip_Error dip_GaussianRandomVariable ( random, mean, variance, output1, output2 )`

FUNCTION

`GaussianRandomVariable` uses the algorithm described by D.E. Knuth as the Polar Method to generate two Gaussian distributed random variables. See `RandomVariable` for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_float	mean	Mean of the distribution, the samples are drawn from
dip_float	variance	Variance of the distribution, the samples are drawn from
dip_float *	output1	First output value
dip_float *	output2	Second output value

EXAMPLE

Get two Gaussian random variable as follows:

```
dip_Random random;
dip_float mean, variance, value1, value2;

mean = 0.0;
variance = 1.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_GaussianRandomVariable( &random, mean, variance, &value1, &value2 ));
```

LITERATURE

Knuth, D.E., *Seminumerical algorithms, The art of computer programming, vol. 2, second edition* Addison-Wesley, Menlo Park, California, 1981.

SEE ALSO

RandomVariable, RandomSeed, RandomSeedVector, UniformRandomVariable, PoissonRandomVariable, BinaryRandomVariable

GaussianSigma

Adaptive Gaussian smoothing filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_GaussianSigma ( in, out, boundary, sigma, gaussSigma,
outputCount, truncation )
```

DATA TYPES

integer, **float**

FUNCTION

The GaussianSigma filter is an adaptive Gauss-ian smoothing filter. The value of the pixel under investigation is replaced by the Gaussian-weighted average of the pixelvalues in the filter region which lie in the interval `+/- 2 sigma` from the value of the pixel that is filtered. The filter region is specified by `gaussSigma` and `truncation`. If `outputCount` is `DIP_TRUE`, the output values represent the number of pixel over which the average has been calculated. When `threshold` is `DIP_TRUE`, the pixel intensities are thresholded at `+/- 2 sigma`, when it is set to `DIP_FALSE`, the intensities are weighted with the Gaussian difference with the intensity of the center pixel.

With `threshold` set to `DIP_FALSE`, this filter is also known as the bilateral filter.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_BoundaryArray	boundary	Boundary conditions
dip_float	sigma	Sigma
dip_FloatArray	gaussSigma	Sigma of Gaussian
dip_Boolean	outputCount	Output the Count
dip_float	truncation	Truncation of Gaussian, see GlobalGaussianTruncationGet

LITERATURE

John-Sen Lee, *Digital Image Smoothing and the Sigma Filter*, Computer Vision, Graphics and Image Processing, 24, 255-269, 1983

SEE ALSO

`Sigma`, `BiasedSigma`, `Gauss`

GaussIIR

Infinite impulse response filter

SYNOPSIS

```
#include "dip_iir.h"
dip_Error dip_GaussIIR ( in, out, boundary, process, sigmas, order,
truncation )
```

DATA TYPES

binary, integer, **float**

FUNCTION

Recursive infinite impulse response implementation of the Gauss filter.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	process	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_IntegerArray	order	Order of Derivative
dip_IntegerArray	order	Order of the IIR Filter
dip_int	designMethod	Method of IIR design
dip_float	truncation	Truncation of Gaussian, see GlobalGaussianTruncationGet

SEE ALSO

Gauss, Derivative

GeneralConvolution

Genaral convolution filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_GeneralConvolution ( in, psf, out, boundary )
```

DATA TYPES

integer, **float**, **complex**

FUNCTION

This function convolves the `in` image with the point spread function `psf`, directly in the spatial domain. If the kernel `psf` is separable, use the function `SeparableConvolution` instead. If `psf` is large (and not separable), use the function `ConvolveFT` instead.

If the image `psf` is even in size, the origin is taken as the pixel to the right of the middle.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Psf image
dip_Image	out	Output image
dip_BoundaryArray	boundary	Boundary conditions

SEE ALSO

General information about convolution

`SeparableConvolution`, `ConvolveFT`, `Uniform`

GeneralisedKuwahara

Generalised Kuwahara filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_GeneralisedKuwahara ( in, selection, out, se, boundary, param,
shape, minimum )
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function is a generalisation of the Kuwahara filter in the sense that is does not use the variance criterion to select the smoothed value, but instead accepts an image with the selection values. The algorithm finds, for every pixel, the minimum or maximum (as specified with `minimum`) value of `selection` within the filter window (its size specified by `param`), and outputs the corresponding value in `in`. When `in` is the output of `Uniform`, and `selection` is the output of `VarianceFilter`, this function produces the same result as `Kuwahara`.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	selection	Selection
dip_Image	out	Output
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape
dip_Boolean	minimum	Select minimum or maximum?

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

Kuwahara, KuwaharaImproved, GeneralisedKuwaharaImproved, VarianceFilter, Uniform

GeneralisedKuwaharaImproved

Generalised Kuwahara filter

SYNOPSIS

`#include "dip_filtering.h"`

`dip_Error dip_GeneralisedKuwaharaImproved ( in, selection, out, se, boundary, param, shape, threshold, minimum )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function implements an improved version of GeneralisedKuwahara, see that function's description for more information. This function adds a `threshold` parameter that avoids false edges in uniform regions. If the difference between maximal and minimal values within the filter window is smaller or equal to `threshold`, the centre pixel is taken, instead of the minimum (or maximum). Setting `threshold` to zero yields the same result as GeneralisedKuwahara.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	selection	Selection
dip_Image	out	Output
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape
dip_float	threshold	Minimal value difference within window
dip_Boolean	minimum	Select minimum or maximum?

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

`Kuwahara`, `GeneralisedKuwahara`, `KuwaharaImproved`, `VarianceFilter`, `Uniform`

Get

Get a pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Get ( in, const, cor, adjust )
```

FUNCTION

This functions get the value of a pixel in image `in` at the coordinate `cor`. If `cor` is zero, the first pixel value is retrieved.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`const`	0-D output image
`dip_IntegerArray`	`cor`	Pixel coordinate
`dip_Boolean`	`adjust`	Adjust data type of output image

SEE ALSO

`GetInteger`, `GetFloat`, `GetComplex`, `dip__PixelGetInteger`, `dip__PixelGetFloat`, `Set`

GetComplex

Get complex pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_GetComplex ( in, value, cor )
```

FUNCTION

This functions get the value of a pixel in image `in` at the coordinate `cor`. If `cor` is zero, the first pixel value is retrieved.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_complex *	value	Value
dip_IntegerArray	cor	Pixel coordinate

SEE ALSO

Get, GetInteger, GetFloat, dip__PixelGetInteger, dip__PixelGetFloat, Set

GetFloat

Get float pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_GetFloat ( in, value, cor )
```

FUNCTION

This functions get the value of a pixel in image `in` at the coordinate `cor`. If `cor` is zero, the first pixel value is retrieved.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_float *	value	Value
dip_IntegerArray	cor	Pixel coordinate

SEE ALSO

Get, GetInteger, GetComplex, dip__PixelGetInteger, dip__PixelGetFloat, Set

GetInteger

Get integer pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_GetInteger ( in, value, cor )
```

FUNCTION

This functions get the value of a pixel in image `in` at the coordinate `cor`. If `cor` is zero, the first pixel value is retrieved.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_int *	value	Value
dip_IntegerArray	cor	Pixel coordinate

SEE ALSO

Get, GetFloat, GetComplex, dip__PixelGetInteger, dip__PixelGetFloat, Set

GetLibraryInformation

Support function

SYNOPSIS

```
#include "dip_information.h"
dip_Error dip_GetLibraryInformation ( info )
#include "dipio_image.h"
dip_Error dipio_GetLibraryInformation ( info )
```

FUNCTION

This function fills the given `dip_LibraryInformation` structure with information about the release version and date, copyright information and author information of the DIPlib library.

ARGUMENTS

Data type	Name	Description
`dip_LibraryInformation*`	`info`	DIPlib library information

GetLine

Get a line from an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_GetLine ( in, out, cor, dimension )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

Get a orthogonal line form an image. The position of the line in the image is specified by the coordinates at which its left most pixel (`cor`) should be placed and on which dimension of the image, the dimension of the line maps (`dimension`). If `in` has If `in` has a different type than `out`, it will be converted to the type of `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input Image
dip_Image	out	Output Line Image
dip_IntegerArray	cor	Coordinate in the image of the left most pixel of the line
dip_int	dimension	Dimension of the image on which the line's dimension maps

SEE ALSO

GetSlice, PutSlice, PutLine

GetMaximumAndMinimum

statistics function

SYNOPSIS

`dip_Error dip_GetMaximumAndMinimum ( in, mask, max, min )`

DATA TYPES

integer, **float**

FUNCTION

This function gets both the maximum and minimum of all the pixel values in the `in` image. Optionally, a `mask` image can be specified to exclude pixels from this search.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	mask (0)	Mask image
dip_float	*max	Pointer to maximum variable
dip_float	*min	Pointer to minimum variable

SEE ALSO

Maximum, Minimum

GetObjectLabels

Lists object labels in image

SYNOPSIS

`dip_Error dip_GetObjectLabels ( in, mask, labels, nullIsObject, resources )`

DATA TYPES

binary, **integer**

FUNCTION

This function produces an array of object labels present in the image `in`. Optionally, `mask` can mask the regions in `in` where to search for labels. The boolean `nullIsObject` specifies whether or not to treat the value zero as an object label.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input label image
`dip_Image`	`mask`	Mask image
`dip_IntegerArray *`	`labels`	Array of labels
`dip_Boolean`	`nullIsObject`	treat the value zero ad an object label
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`Label`, `IntegerArrayFind`

GetRank

Value selection function

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_GetRank ( array, datatype, min, max, rank, value )
```

FUNCTION

GetRank gets the value at rank `rank` in the array `array`. `min` should be set to the first index of `array`, `max` to the last. `dip_GetRank` will use `array` for temporary storage, so the values in the array will be changed are this function is ready.

ARGUMENTS

Data type	Name	Description
dip_float *	array	Array to searched in
dip_DataType	datatype	
dip_int	min	minimal array index
dip_int	max	maximal array index
dip_int	rank	Rank
dip_float *	value	Value of the rank element

EXAMPLE

This example finds the median value for the array.

```
dip_float array[ SIZE ], median;
dip_int rank;

/* fill the array with values */

rank = SIZE/2;
DIPXX( dip_GetRank( array, DIP_DT_FLOAT, 0, (SIZE - 1), rank, &median ));
```

SEE ALSO

General information about sorting

DistributionSort, DistributionSortIndices, DistributionSortIndices16, InsertionSort, InsertionSortIndices, InsertionSortIndices16, QuickSort, QuickSortIndices, QuickSortIndices16, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

GetSlice

Get a slice from an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_GetSlice ( in, out, cor, dim1, dim2 )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

Get a orthogonal slice from a image. The requested slice is selected by specifying its upper left corner (`cor`) and on which dimensions of the image, the dimensions of the slice map (`dim1`, `dim2`). If `in` has a different type than `out`, it will be converted to the type of `out`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	3D Input Image
`dip_Image`	`out`	2D Output Image
`dip_IntegerArray`	`cor`	Coordinate in `in` of the upper left corner of the slice
`dip_int`	`dim1`	Dimension of `in` on which the slice's first dimension maps
`dip_int`	`dim2`	Dimension of `in` on which the slice's second dimensionmaps

SEE ALSO

PutSlice, GetLine, PutLine

GetUniqueNumber

Obtain an unique value

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GetUniqueNumber ( number )
```

FUNCTION

This function gives an unique integer value. The value is unique is the sense that its value has not yet been returned by this function nor will it be returned by subsequent calls.

ARGUMENTS

Data type	Name	Description
dip_int *	number	Pointer to an integer in which the number is stored

GlobalBoundaryConditionGet

Get global Boundary Conditions

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalBoundaryConditionGet ( boundary, size, resources )
```

FUNCTION

This function allocates the boundary array `array` of size `size` with the global default boundary conditions for each dimension of the image. The initial values of this global array is `DIP_BC_SYMMETRIC_MIRROR`.

ARGUMENTS

Data type	Name	Description
dip_BoundaryArray *	boundary	Pointer to Boundary conditions
dip_int	size	Size of the new array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Boundary conditions

GlobalBoundaryConditionSet, GlobalGaussianTruncationGet, GlobalGaussianTruncationSet, GlobalFilterShapeGet, GlobalFilterShapeSet

GlobalBoundaryConditionSet

Set global boundary conditions

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalBoundaryConditionSet ( boundary )
```

FUNCTION

This function sets the global boundary conditions equal to `boundary`.

ARGUMENTS

Data type	Name	Description
dip_BoundaryArray	boundary	Boundary conditions

SEE ALSO

GlobalBoundaryConditionGet, GlobalGaussianTruncationGet, GlobalGaussianTruncationSet, GlobalFilterShapeGet, GlobalFilterShapeSet

GlobalFilterShapeGet

Get global filter shape value

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalFilterShapeGet ( shape )
```

FUNCTION

This function gets the global default of the filter shape used by DIPlib's linear and morphology filters. The initial value of this global is DIP_FLT_SHAPE_RECTANGULAR.

This setting currently has no effect on any of the filters in DIPlib.

ARGUMENTS

Data type	Name	Description
dip_FilterShape *	shape	Filter shape

SEE ALSO

GlobalBoundaryConditionGet, GlobalBoundaryConditionSet, GlobalGaussianTruncationGet, GlobalGaussianTruncationSet, GlobalFilterShapeSet

GlobalFilterShapeSet

Set the global filter shape value

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalFilterShapeSet ( shape )
```

FUNCTION

This function sets the global default of the filter shape used by DIPlib's linear and morphology filters. The initial value of this global is `DIP_FLT_SHAPE_RECTANGULAR`.

This setting currently has no effect on any of the filters in DIPlib.

ARGUMENTS

Data type	Name	Description
dip_FilterShape	shape	Filter shape

SEE ALSO

GlobalBoundaryConditionGet, GlobalBoundaryConditionSet, GlobalGaussianTruncationGet, GlobalGaussianTruncationSet, GlobalFilterShapeGet

GlobalGaussianTruncationGet

Get the global gaussian truncation

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalGaussianTruncationGet ( truncation )
```

FUNCTION

This function gets the global default of the truncation used by the finite impluse response implementation of the Gauss (derivative) filter. The initial value of this global is 3.0.

ARGUMENTS

Data type	Name	Description
dip_float *	truncation	Gaussian truncation

SEE ALSO

GlobalBoundaryConditionGet, GlobalBoundaryConditionSet, GlobalGaussianTruncationSet, GlobalFilterShapeGet, GlobalFilterShapeSet

GlobalGaussianTruncationSet

Set the global gaussian truncation

SYNOPSIS

```
#include "dip_globals.h"
dip_Error dip_GlobalGaussianTruncationSet ( truncation )
```

FUNCTION

This function sets the global default of the truncation used by the finite impluse response implementation of the Gauss (derivative) filter. The initial value of this global is 3.0.

ARGUMENTS

Data type	Name	Description
dip_float	truncation	Truncation

SEE ALSO

GlobalBoundaryConditionGet, GlobalBoundaryConditionSet, GlobalGaussianTruncationGet, GlobalFilterShapeGet, GlobalFilterShapeSet

GradientDirection2D

Derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_GradientDirection2D ( in, out, boundary, ps, sigmas, tc,
atanFlavour, flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes the gradient direction of an image using the `Derivative` function. This functions supports only two dimensional images.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_float	tc	Truncation of Gaussian, see GlobalGaussianTruncationGet
dip_GradientDirectionAtanFlavour	atanFlavour	Atan flavour
dip_DerivativeFlavour	flavour	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

See section 9.5, "Derivative-based operations", in Fundamentals of Image Processing.

`Derivative`, `GradientMagnitude`, `Laplace`, `Dgg`, `LaplacePlusDgg`, `LaplaceMinDgg`

GradientMagnitude

Derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_GradientMagnitude ( in, out, boundary, ps, sigmas, tc, flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes the gradient magnitude of an image using the `Derivative` function.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps (0)	Dimensions to process
dip_FloatArray	sigmas	Sigma of the Gaussian
dip_float	tc	Gaussian truncation, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

See section 9.5, “Derivative-based operations”, in Fundamentals of Image Processing.

`Derivative`, `GradientDirection2D`, `Laplace`, `Dgg`, `LaplacePlusDgg`, `LaplaceMinDgg`

Greater

Compare grey values in two images

SYNOPSIS

`dip_Error dip_Greater ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to "true" when for corresponding pixels `in1` > `in2`. This is the same as `Compare` with the `DIP_SELECT_GREATER` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to a functionality similar to that of `Threshold`.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

`Compare`, `Threshold`, `Equal`, `Lesser`, `NotEqual`, `NotGreater`, `NotLesser`, `SelectValue`, `NotZero`

GreyValuesInPixelTable

Copy greyvalues from image in pixel table

SYNOPSIS

`#include "dip_pixel_table.h"`

`dip_Error dip_GreyValuesInPixelTable ( table, image, ptgreyvalues, resources )`

DATA TYPES

integer, float

FUNCTION

This functions converts a grey-value image to a newly allocated floating-point array, in which each element is the grey value associated to a pixel in the pixel table. The image must have the same size and dimensionality as the pixel table's bounding box. For example:

```
dip_Image kernel, binkernel;
dip_PixelTable table;
dip_FloatArray values;
...
dip_NotZero( kernel, binkernel );
dip_BinaryImageToPixelTable( binkernel, &table, resources );
dip_GreyValuesInPixelTable( table, kernel, &values, resources );
...
process->filter->array[0].parameters = values;
dip_PixelTableFrameWork( in, out, boundary, process, table );
```

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_Image	image	Grey-value image
dip_FloatArray *	ptgreyvalues	Array to which to write pixel grey values
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Description of DIPlib's pixel tables

`BinaryImageToPixelTable`, `PixelTableCreateFilter`

GreyWeightedDistanceTransform

Grey weighted distance transform

SYNOPSIS

`#include "dip_distance.h"`

`dip_Error dip_GreyWeightedDistanceTransform ( in, seed, out, distance, chamfer, neighborhood, metric )`

DATA TYPES

`in`: integer, float

`seed`: binary

FUNCTION

`GreyWeightedDistanceTransform` determines the grey weighted distance transform of the object elements in the `in` image and returns the result in the `out` image. The implemented algorithm uses a heap sort for sorting the pixels to be processed.

The images `in` and `seed` must have the same dimensions. The `out` image will be converted to a `sfloat` typed image. The `seed` image defines the elements that are part of the object for which the GDT is determined. It can be any type of image where all image elements not equal to 0 are considered to be part of the object(s). Those elements that are neighboring an object element in the output image are considered seeds. Before any seeds are detected the borders of the `out` image are set to 0. The size of the border is determined by the chamfer metric size (see below). In case of a 3 by 3 chamfer metric the image border is one element, in case of a 5 by 5 chamfer it is 2 elements. Elements in the border are not considered seeds. If no valid seeds are found the routine will terminate with an Illegal value error code.

The chamfer metric is defined by two parameters: `neighborhood` and `metric`. `neigborhood` should supply the different relative addresses of the neighboring elements according to the chamfer metric. The first element `neighborhood[0]` contains the number of elements in the chamfer neighborhood. The next three elements contain the maximum number of elements a chamfer metric exceeds the central element. The rest of the elements (starting from the fifth element) contain addresses of the different chamfer elements relative to the central element. The `metric` array contains the corresponding chamfer metric value. An example of a 3x3 neighborhood array with the corresponding metric is:

```
neighborhood[0] = 8 (number of elements)
neighborhood[1] = 1 (x-border size)
neighborhood[2] = 1 (y-border size)
```

```
neighborhood[3] = 0 (z-border size)
neighborhood[4] = -imagewidth - 1,    metric[0] = 7
neighborhood[5] = -imagewidth,        metric[1] = 5
neighborhood[6] = -imagewidth + 1,    metric[2] = 7
neighborhood[7] = -1,                 metric[3] = 5
neighborhood[8] = 1,                  metric[4] = 5
neighborhood[9] = imagewidth - 1,     metric[5] = 7
neighborhood[10] = imagewidth,        metric[6] = 5
neighborhood[11] = imagewidth + 1,    metric[7] = 7
```

where `imagewidth` represents the width of the image in image pixels. If both `neighborhood` and `metric` pointers are `NULL`, the `chamfer` variable can be set to either 1 (indicating a 3x3 or 3x3x3 chamfer using only 4 or 6 direct neighbors), 3 (indicating a 3x3 or 3x3x3 chamfer, using all neighbors) or 5 (indicating a 5x5 or 5x5x5 chamfer). In these cases a preset `neighborhood` and `metric` arrays will be used.

ARGUMENTS

Data type	**Name**	**Description**
`dip_Image`	`in`	Input image
`dip_Image`	`seed`	Seed image
`dip_Image`	`out`	Integrated grey-value over least-resistance path (output image)
`dip_Image`	`distance`	Metric distance over least-resistance path (output image)
`dip_int`	`chamfer`	Chamfer distance metric
`dip_IntegerArray`	`neighborhood`	Neighborhood
`dip_FloatArray`	`metric`	Metric

LITERATURE

"An efficient uniform cost algorithm applied to distance transforms", B.J.H. Verwer, P.W. Verbeek, and S.T. Dekker, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 4, 1989, 425-429.

"Shading from shape, the eikonal equation solved by grey-weighted distance transform", P.W. Verbeek and B.J.H. Verwer, Pattern Recognition Letters, vol. 11, no. 10, 1990, 681-690.

"Local distances for distance transformations in two and three dimensions", B.J.H. Verwer, Pattern Recognition Letters, vol. 12, no. 11, 1991, 671-682.

"Distance Transforms, Metrics, Algorithms, and Applications", B.J.H. Verwer, Ph.D. thesis Delft University of Technology, Delft University Press, Delft, 1991.

"3-D Texture characterized by Accessibility measurements, based on the grey weighted distance transform", K.C. Strasters, A.W.M. Smeulders, and H.T.M. van der Voort, BioImaging, vol 2, no. 1, 1994, p. 1-21.

"Quantitative Analysis in Confocal Image Cytometry", Karel C. Strasters, Delft University

Press, Delft, 1994. ISBN 90-407-1038-4, NUGI 841

KNOWN BUGS

`GreyWeightedDistanceTransform` works only on 2 or 3-dimensional images. It will not work if any of the images has different strides.

`GreyWeightedDistanceTransform` produces incomplete results in a 2-pixel border around the edge (4 for `chamfer` = 5). If this is an issue, consider adding 2 pixels on each side of your image. Make sure that `in` has high grey values in the border to avoid unexpected output.

The function `GrowRegionsWeighted` produces a grey-weighted distance transform without these limitations and with some other possibilities.

AUTHOR

Karel C. Strasters, adapted to DIPlib by Geert M.P. van Kempen

SEE ALSO

`GrowRegionsWeighted`, `EuclideanDistanceTransform`, `VectorDistanceTransform`

GrowRegions

Dilate the regions in a labelled image

SYNOPSIS

```
#include "dip_regions.h"
dip_Error dip_GrowRegions ( in, grey, mask, out, connectivity, iterations,
order )
```

DATA TYPES

`in`: **binary**, **integer**

`grey`: interger, float (converted to dip_sfloat)

`mask`: `dip_uint8`

FUNCTION

The regions in the input image `in` are grown with several options:

If `grey` is `NULL`, the regions are dilated `iterations` steps, according to `connectivity` (see The connectivity parameter), and optionally constrained by `mask`. This is the labelled equivalent to `BinaryPropagation`. If `iterations` is 0, the objects are dilated until no further change is possible. `order` is ignored.

If an image `grey` is given, the labels are grown in order of the grey-values in `grey`. `order` indicates whether pixels with high grey-values are added first or last. `iterations` is ignored, and `mask` is an optional constraint. This is a watershed algorithm with initial labels. The function `Watershed` does not accept an initial segmentation, so these two functions complement each other. Note that `GrowRegions` does not leave any watershed pixels in between the regions.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input binary or labelled image
dip_Image	grey	Input grey-value image
dip_Image	mask	Mask image
dip_Image	out	Output binary or labelled image
dip_int	connectivity	Connectivity
dip_int	iterations	Number of iterations
dipf_GreyValueSortOrder	order	Whether to grow from low to high or high to low

The `dipf_GreyValueSortOrder` enumeration consists of the following values:

Name	Description
DIP_GVSO_HIGH_FIRST	Process the pixels from high grey-value to low grey-value.
DIP_GVSO_LOW_FIRST	Process the pixels from low grey-value to high grey-value.

SEE ALSO

GrowRegionsWeighted, Watershed, BinaryPropagation, Label

GrowRegionsWeighted

Grow labelled regions using grey-weighted distances

SYNOPSIS

```
#include "dip_regions.h"
dip_Error GrowRegionsWeighted ( in, grey, mask, out, distance, pixelsize,
chamfer, metric )
```

DATA TYPES

`in`: binary, **integer**

`grey`: interger, float (converted to dip_sfloat)

`mask`: dip_uint8

FUNCTION

The regions in the input image `in` are grown according to a grey-weighted distance metric; the weights are given by `grey`. The optional mask image `mask` limits the growing. `out` contains the grown regions, and `distance`, if not 0, contains the grey-weighted distance of each pixel in `mask` to the nearest pixel in `in`. Non-isotropic sampling is supported through `pixelsize`, which can be set to 0 to assume isotropic sampling. `chamfer` selects the size of the chamfer metric: 3 or 5. Set `chamfer` to 0 to use a custom metric given by the image `metric`. This image should be odd in size, and each pixel gives the distance to the center pixel. The pixels set to 0 will not be considered as neighbors.

The chamfer metric used is the following for `chamfer`==3 (with `ps0=pixelsize->array[0]` and `ps1=pixelsize->array[1]`):

sqrt(ps0*ps0+ps1*ps1)	ps1	sqrt(ps0*ps0+ps1*ps1)
ps0	0	ps0
sqrt(ps0*ps0+ps1*ps1)	ps1	sqrt(ps0*ps0+ps1*ps1)

and the following for `chamfer`==5:

0	sqrt(ps0*ps0+4*ps1*ps1)	0	sqrt(ps0*ps0+4*ps1*ps1)	0
sqrt(4*ps0*ps0+ps1*ps1)	sqrt(ps0*ps0+ps1*ps1)	ps1	sqrt(ps0*ps0+ps1*ps1)	sqrt(4*ps0*ps0+ps1*ps1)
0	ps0	0	ps0	0
sqrt(4*ps0*ps0+ps1*ps1)	sqrt(ps0*ps0+ps1*ps1)	ps1	sqrt(ps0*ps0+ps1*ps1)	sqrt(4*ps0*ps0+ps1*ps1)
0	sqrt(ps0*ps0+4*ps1*ps1)	0	sqrt(ps0*ps0+4*ps1*ps1)	0

Setting `chamfer` to 0 and `metric` to an image with these values produces the same results as setting `chamfer` to 3 or 5.

The output image `distance` is comparable to the `out` image of `GreyWeightedDistanceTransform`, except that that function uses optimal chamfer distances whereas this one uses the (sub-optimal) true distance. In return, this function works on images of any dimensionality, allows for non-isotropic sampling, does not skip pixels close to the edge of the image, and can be used with a mask image to constrain the propagation. Note that the `seed` image in `GreyWeightedDistanceTransform` corresponds to the zero pixels of `in` for this function.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input binary or labelled image
`dip_Image`	`grey`	Input grey-value image
`dip_Image`	`mask`	Mask image
`dip_Image`	`out`	Output binary or labelled image
`dip_Image`	`distance`	Output distance image
`dip_FloatArray`	`pixelsize`	Pixel size
`dip_int`	`chamfer`	Chamfer distance
`dip_Image`	`metric`	Custom metric

LITERATURE

"3-D Texture characterized by Accessibility measurements, based on the grey weighted distance transform", K.C. Strasters, A.W.M. Smeulders, and H.T.M. van der Voort, BioImaging, vol 2, no. 1, 1994, p. 1-21.

"An efficient uniform cost algorithm applied to distance transforms", B.J.H. Verwer, P.W. Verbeek, and S.T. Dekker, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 4, 1989, 425-429.

SEE ALSO

`GrowRegions`, `GreyWeightedDistanceTransform`, `Label`

HartleyTransform

Computes the Hartley transform

SYNOPSIS

```
#include "dip_transform.h"
dip_Error dip_HartleyTransform ( in, out, trFlags, process )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function computes a Hartley transform on `in` and places the result in `out`.

Normalisation: 1/sqrt(dimension) for each dimension.

The main advantage of the Hartley transform over the Fourier transform is that is requires half the storage for real valued images. Note, that is also possible to directly reduce the storage requirements of the Fourier transform by just storing the right half plane, since for real valued images the left half plane can be derived from the right half using the symmetry properties of the Fourier transform.

Unfortunately there seem to be two definitions of the multi-dimensional Hartley transform (they are identical in the 1-D case). DIPlib implements the Bracewell (see below) variant, since this one is easy to implement and inherits the storage advantage from the 1-D case. The following are references which each use a different variant (all scaling factors have been dropped):

Bracewell, *"Discrete Hartley Transform"*, J. Opt. Soc. Am, vol. 73, no. 12, December 1983 :

```
DHT(u,v) = Sum Sum I(x,y) cas( ux ) cas( vy )
            y   x
```

Kenneth R. Castleman, *"Digital image processing"*, Prentice Hall, 1996 :

```
DHT(u,v) = Sum Sum I(x,y) cas( ux + vy )
            y   x
```

Using cas(a) = cos(a) + sin(a) :

```
cas(ux)cas(vy) = cos(ux)cos(vy)+cos(ux)sin(vy)+sin(ux)cos(vy)+sin(ux)sin(vy)
```

```
cas(ux+vy)     = cos(ux)cos(vy)+cos(ux)sin(vy)+sin(ux)cos(vy)-sin(ux)sin(vy)
                                                             ^^^
```

A subtle difference. The two definitions have very similar properties, for example the convolution property.

In implementation terms, Bracewell is equivalent to perform the one-dimensional Hartley transform along each dimension. The Castleman variant is equivalent to the definition: DHT = re(DFT) - im(DFT). On a final note, I've not noticed mention of the difference between the two variants, so the indications Bracewell's and Castleman's variant are not and should not be accepted "labels" to refer to the variants (For both variants I have selected the first reference I came across, not chronologically the first reference to use the variant).

Defaults: `process` may be zero, indicating that all dimensions should be processed.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dipf_FourierTransform`	`trFlags`	Transformation flags
`dip_BooleanArray`	`process (0)`	Dimensions to process

The `dipf_FourierTransform` enumeration consists of the following flags:

Name	Description
`DIP_TR_FORWARD`	Forward transformation
`DIP_TR_INVERSE`	Inverse transformation

SEE ALSO

`FourierTransform`

HasContiguousData

Determines whether an image has all data contiguous in memory

SYNOPSIS

`dip_Error dip_HasContiguousData( image, &answer )`

FUNCTION

Determines whether an image has all data contiguous in memory. This can potentially not be the case if the image is an ROI, for example, or if it was allocated with strides that cause unused gaps in the image's memory block. If `answer` is not zero, the verdict is passed in this variable. Otherwise, `HasContiguousData` returns an error in case `image` does not have contiguous data.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	The image under investigation
`dip_Boolean *`	`answer`	The verdict

SEE ALSO

The image structure

`HasNormalStride`, `ImageGetStride`, `IsScalar`

HasNormalStride

Determines whether an image has a normal stride

SYNOPSIS

`dip_Error dip_HasNormalStride( image, &answer )`

FUNCTION

Determines whether an image has a normal stride. Normal stride is defined as a stride of 1 in the first dimension, a stride of image width in the second dimension, etc. If `answer` is not zero, the verdict is passed in this variable. Otherwise, `HasNormalStride` returns an error in case `image` does not have a normal stride.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	The image under investigation
`dip_Boolean *`	`answer`	The verdict

SEE ALSO

The image structure

`HasContiguousData`, `ImageGetStride`, `IsScalar`

HysteresisThreshold

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_HysteresisThreshold ( in, out, low, high )
```

DATA TYPES

integer, **float**

FUNCTION

Performs hysteresis thresholding. From the binary image (in>low) only those regions are selected for which at least one location also has (in>high). The output image will be a binary image with foreground pixel 1 and background pixel 0;

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	low	Lower threshold
dip_float	high	Higher threshold

SEE ALSO

Threshold, RangeThreshold, IsodataThreshold

IDivergence

difference measure

SYNOPSIS

`dip_Error dip_IDivergence ( in1, in2, mask, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Calculates the I-divergence between each pixel value of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

The I-Divergence is defined as: `I(x,y) = x ln(x/y) - (x - y)` and is divied by the number of pixels. It is the -log of a possion distribution `p(x,y)=e^(-y)/x!-y^x` with the stirling approximation for `ln x!`. For x=0, the stirling approximation would fail, `y` is returned.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in1`	First input, Data:x
`dip_Image`	`in2`	Second input, Model:y
`dip_Image`	`mask`	Mask
`dip_Image`	`out`	Output

LITERATURE

Why Least Squares and Maximum Entropy? An axiomatic approach to inference for linear inverse problems , I. Csiszar, The Annals of Statistics, 19, 2032-2066, 1991.

SEE ALSO

`MeanError`, `MeanSquareError`, `RootMeanSquareError`, `MeanAbsoluteError`, `LnNormError`

ImageArrayFree

Array free function

SYNOPSIS

`dip_Error dip_ImageArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
`dip_ImageArray *`	`array`	Array

SEE ALSO

ImageArrayNew, ImageArrayFree

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

ImageArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_ImageArrayNew ( array, size, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_ImageArray` and sets the size of the array to `size`.

ARGUMENTS

Data type	Name	Description
dip_ImageArray *	array	Array
dip_int	size	Size
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ImageArrayNew, ImageArrayFree

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

ImageAssimilate

Inherit properties of another image

SYNOPSIS

`dip_Error dip_ImageAssimilate( example, target )`

FUNCTION

Give the target image the same properties (type, data type, etc...) as the example image. The example image may be either "raw" or "forged". The target image is forged.

If the target was forged before calling this function, and it exactly matches the example, nothing happens. If it doesn't match the example, it is stripped before the properties are copied.

ARGUMENTS

Data type	Name	Description
dip_Image	example	An example image
dip_Image	target	The target image

SEE ALSO

ImageCopyProperties, ChangeDataType, ChangeTo0d

ImageChainCode

Extracts all chain codes from a labeled image

SYNOPSIS

```
#include "dip_chaincode.h"
dip_Error dip_ImageChainCode ( objectIm, connectivity, objectID,
chaincodearray, resources )
```

DATA TYPES

integer

FUNCTION

Extracts the chain codes for the objects in `objectIm` (only 2D images supported) that are listed in `objectID`, assuming that each object is compact (i.e. it returns the chain code for only one border for each label ID in `objectID`). Chain codes are constructed according to `connectivity`, which can only be 1 or 2 (see The connectivity parameter). The output structure `chaincodearray` is allocated by this function and registered in `resources`.

The `dip_ChainCodeArray` structure, like all arrays in DIPlib, contains a `size` and an `array` element. Each element is of type `dip_ChainCode`, and accessed by `chaincodearray->array[ii]`, where `ii` is between `0` and `chaincodearray->size-1`. Data in the `dip_ChainCode` structures can only be accessed through the corresponding access functions, see `ChainCodeNew`.

ARGUMENTS

Data type	Name	Description
dip_Image	objectIm	Labeled input image
dip_int	connectivity	Pixel connectivity of the objects
dip_IntegerArray	objectID	Array containing object label values
dip_ChainCodeArray *	chaincodearray	Output chain codes
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

`ChainCodeNew`, `ChainCodeFree`, `ChainCodeArrayNew`, `ChainCodeArrayFree`, `ChainCodeGetSize`, `ChainCodeGetChains`, `ChainCodeGetStart`, `ChainCodeGetLabel`, ChainCodeGetConnectivity, ChainCodeGetLength, ChainCodeGetLongestRun, ChainCodeGetFeret

ImageCheckBooleanArray

Check a boolean array

SYNOPSIS

`dip_Error dip_ImageCheckBooleanArray ( im, array, answer)`

FUNCTION

This functions check whether the size of `array` is equal to the dimensionality of `im`. If `answer` is not zero, it will contain the result of the test, otherwise the `DIP_E_ARRAY_ILLEGAL_SIZE` will be set when the test has failed.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_BooleanArray	array	Array
dip_Boolean *	answer	Answer

SEE ALSO

ImageCheckIntegerArray, ImageCheckFloatArray, ImageCheckComplexArray, ImageCheckBoundaryArray

ImageCheckBoundaryArray

Check a boundary array

SYNOPSIS

`dip_Error dip_ImageCheckBoundaryArray ( im, array, answer)`

FUNCTION

This functions check whether the size of `array` is equal to the dimensionality of `im`. If `answer` is not zero, it will contain the result of the test, otherwise the `DIP_E_ARRAY_ILLEGAL_SIZE` will be set when the test has failed.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_BoundaryArray	array	Boundary conditions
dip_Boolean *	answer	Answer

SEE ALSO

ImageCheckIntegerArray, ImageCheckFloatArray, ImageCheckComplexArray, ImageCheckBoundaryArray

ImageCheckComplexArray

Check a complex array

SYNOPSIS

`dip_Error dip_ImageCheckComplexArray ( im, array, answer)`

FUNCTION

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_ComplexArray	array	Array
dip_Boolean *	answer	Answer

SEE ALSO

ImageCheckIntegerArray, ImageCheckFloatArray, ImageCheckComplexArray, ImageCheckBoundaryArray

ImageCheckFloatArray

Check a float array

SYNOPSIS

`dip_Error dip_ImageCheckFloatArray ( im, array, answer)`

FUNCTION

This functions check whether the size of `array` is equal to the dimensionality of `im`. If `answer` is not zero, it will contain the result of the test, otherwise the `DIP_E_ARRAY_ILLEGAL_SIZE` will be set when the test has failed.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_FloatArray	array	Array
dip_Boolean *	answer	Answer

SEE ALSO

ImageCheckIntegerArray, ImageCheckFloatArray, ImageCheckComplexArray, ImageCheckBoundaryArray

ImageCheckIntegerArray

Check an integer array

SYNOPSIS

dip_Error dip_ImageCheckIntegerArray (im, array, answer)

FUNCTION

This functions check whether the size of array is equal to the dimensionality of im. If answer is not zero, it will contain the result of the test, otherwise the DIP_E_ARRAY_ILLEGAL_SIZE will be set when the test has failed.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_IntegerArray	array	Integer rray
dip_Boolean *	answer	Answer

SEE ALSO

ImageCheckIntegerArray, ImageCheckFloatArray, ImageCheckComplexArray, ImageCheckBoundaryArray

ImageCopyProperties

Copy the properties of an image

SYNOPSIS

`dip_Error dip_ImageCopyProperties( example, target )`

FUNCTION

Give the target image the same properties (type, data type, etc...) as the example image. The example image may be either "raw" or "forged", whereas the target image must be "raw". See ImageAssimilate.

ARGUMENTS

Data type	Name	Description
dip_Image	example	An example image
dip_Image	target	The target image

SEE ALSO

The image structure

ImageFileGetInfo

Get information about image in file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageFileGetInfo ( imInfo, filename, format, addExtensions, recognised, resources )`

FUNCTION

This function opens an image file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` is allocated by this function. Use `ImageFileInformationFree` to free this structure, or set the `resources` parameter for automatic deallocation. If `format` is 0, all different ImageRead functions are called in sequence until the correct format has been found. If you know the format, get the correct format ID through the registry functions. See File formats recognized by dipIO for a list of currently supported formats.

The boolean `addExtensions` specifies whether `ImageFileGetInfo` should try to add file format extensions to `filename`, if the registered file format reader fails to recognise `filename` straight away. The extensions are provided by the registered file readers.

If `recognised` is not zero, ImageFileGetInfo will set it to `DIP_TRUE` when it has been able to read `filename`, and it will set it to `DIP_FALSE` when it is not able to read the file. No error will be generated in this case.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation *	imInfo	Output image file information. See `ImageFileInformationNew`
dip_String	filename	File name
dip_int	format	ID of file format
dip_Boolean	addExtensions	Add file format extensions to `filename`
dip_Boolean *	recognised	Pointer to boolean containing the file read status
dip_Resources	resources	Resources tracking structure. See `ResourcesNew`

SEE ALSO

ImageReadCSVInfo, ImageReadGIFInfo, ImageReadICSInfo, ImageReadLSMInfo, ImageReadPICInfo, ImageReadTIFFInfo, ImageReadJPEGInfo, ImageRead, ImageReadColour, ImageReadROI

ImageFileInformationFree

Free a Image File Information structure (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageFileInformationFree ( imInfo )`

FUNCTION

Frees a `dipio_ImageFileInformation` structure allocated through `ImageFileInformationNew` or by `ImageFileGetInfo`.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation *	imInfo	Structure to free

SEE ALSO

`ImageFileInformationNew`, `ImageFileGetInfo`

ImageFileInformationNew

Allocate an Image File Information structure (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageFileInformationNew ( newImInfo, name, filetype, datatype, dims, resources )`

FUNCTION

Allocates a `dipio_ImageFileInformation` structure. It must be freed through `ImageFileInformationFree`, unless a `resources` parameter is given, in which case it will be freed automatically when freeing the resources. This structure is usually allocated by `ImageFileGetInfo`.

This function will fill out some of the values in the structure with the values given on the command line. All of these can be 0.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation *	newImInfo	Output structure
dip_String	name	Initial value for `name`
dip_String	filetype	Initial value for `filetype`
dip_DataType	datatype	Initial value for `datatype`
dip_IntegerArray	dims	Initial value for `dimensions`
dip_Resources	resources	Resources tracking structure. See `ResourcesNew`

The structure `dipio_ImageFileInformation` contains the following elements:

Data type	Name	Description
dip_String	name	File name
dip_String	filetype	File format string
dip_DataType	datatype	Data type of image
dip_int	sigbits	Significant bits
dip_IntegerArray	dimensions	Dimensions of image
dipio_PhotometricInterpretation	photometric	Color space
dip_PhysicalDimensions	physDims	Physical dimensions structure. See PhysicalDimensionsNew
dip_int	numberOfImages	Number of images in a TIFF file. If filetype is not “TIFF”, this number is not set
dip_StringArray	history	History tags
dip_Resources	resources	Resource tracking; all elements within this structure are tracked here

The enumerator dipio_PhotometricInterpretation contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SEE ALSO

ImageFileInformationFree, ImageFileGetInfo, PhysicalDimensionsNew

ImageForge

Allocate pixel data for an image

SYNOPSIS

`dip_Error dip_ImageForge( image )`

FUNCTION

Allocates a block of memory to store pixel data for an image. The image must be "raw", and will be "forged" afterwards. The routine will fail if the image fields do not contain a valid combination of values for the image type.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	The image for which the pixel data must be allocated

SEE ALSO

The image structure

`ImageNew`, `ImageFree`, `ImageStrip`, `ImageCopyProperties`

ImageFree

Free an image

SYNOPSIS

`dip_Error dip_ImageFree( image )`

FUNCTION

Free any pixel data associated with the image and return all fields to their initial ("raw") state by calling `ImageStrip`. Then the image structure itself is freed. Notice that you must pass a pointer to the image instead of the image itself. This allows `ImageFree` to set your image variable to zero, preventing further use of the now freed image.

Because `ImageNew` accepts a resources structure to keep track of allocated images, direct calls to `ImageFree` should be unnecessary.

ARGUMENTS

Data type	Name	Description
dip_Image *	image	A pointer to the image to be freed

SEE ALSO

The image structure

`ImageNew`, `ImageForge`, `ImageStrip`, `ImageCopyProperties`

ImageGetData

Get the data pointers of a set of images

SYNOPSIS

`dip_Error dip_ImageGetData( in, idp, iflags, out, odp, oflags, flags, resources )`

FUNCTION

Get the data pointers of a set of images. This function should not be called before the clean up of the previous invocation (by `ResourcesFree`) has been performed. Currently no clean up is required by `ImageGetData`, but any data pointers obtained by a previous call to this function should be considered invalid when calling this function. The `iflags`, `oflags`, and `flags` parameters are not used in the current version. These fields should be set to zero. The resources parameter is mandatory. Any of the image arrays' elements may be set to zero, indicating that it is to be ignored.

No functions that will possibly modify an image should be called after the call to `ImageGetData` and before its clean up. The proper time to call `ImageGetPlane` and `ImageGetStride` is right after the call to `ImageGetData`.

ARGUMENTS

Data type	Name	Description
dip_ImageArray	in	Array of input images
dip_VoidPointerArray *	idp	Returns input data pointers
dipf_ImageGetDataArray	iflags	Flags for input images
dip_ImageArray	out	Array of output images
dip_VoidPointerArray *	odp	Returns output data pointers
dipf_ImageGetDataArray	oflags	Flags for output images
dipf_ImageGetData	flags	Flags for all images
dip_Resources	resources	Resources tracking structure. See `ResourcesNew`

SEE ALSO

The image structure

`ImageGetPlane`, `ImageGetStride`

ImageGetDataType

Read the data type field

SYNOPSIS

`dip_Error dip_ImageGetDataType( image, dataType )`

FUNCTION

Read the `dip_Image` data type field.

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_DataType *	dataType	Returns the data type field

SEE ALSO

The image structure

DIPlib's data types

ImageSetDataType

ImageGetDimensionality

Read the dimensionality field

SYNOPSIS

```
dip_Error dip_ImageGetDimensionality( image, dimensionality )
```

FUNCTION

Read the `dip_Image` dimensionality field.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	An image
`dip_int *`	`dimensionality`	Returns the dimensionality field

SEE ALSO

The image structure

`ImageGetDimensions`

ImageGetDimensions

Read the dimensions array

SYNOPSIS

`dip_Error dip_ImageGetDimensions( image, dimensions, resources )`

FUNCTION

Read the `dip_Image` dimensions Array. The array that is used to return the dimensions in, is allocated by this routine using `IntegerArrayNew`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_IntegerArray *	dimensions	Returns the dimensions Array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

The image structure

`ImageGetDimensionality`

ImageGetPlane

Read the plane number

SYNOPSIS

`dip_Error dip_ImageGetPlane( image, plane )`

FUNCTION

Read the `dip_Image` plane number. For binary images this is the number of the bit in which the data is stored. For other data types it is meaningless. The proper time to call this function is right after `ImageGetData`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_Int *	plane	Returns the plane number

SEE ALSO

The image structure

`ImageGetData`, `ImageGetStride`

ImageGetStride

Read the stride array

SYNOPSIS

`dip_Error dip_ImageGetStride( image, &stride, resources )`

FUNCTION

Read the `dip_Image` stride array. The array that is used to return the dimensions in, is allocated by this routine using `IntegerArrayNew`. The proper time to call this function is right after `ImageGetData`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_IntegerArray *	stride	Returns the stride array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

The image structure

`ImageGetData`, `ImageGetPlane`

ImageGetType

Read the type field

SYNOPSIS

`dip_Error dip_ImageGetType( image, type )`

FUNCTION

Read the `dip_Image` type field.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	An image
`dip_ImageType *`	`type`	Returns the type field

SEE ALSO

The image structure

`ImageSetType`

ImageIsGIF

Confirm that a file is a GIF file (in dipIO)

SYNOPSIS

```
#include "dipio_gif.h"
dip_Error dipio_ImageIsGIF ( filename, veredict )
```

FUNCTION

This function verifies that the file is an GIF file. `veredict` is set to `DIP_TRUE` if it is, and to `DIP_FALSE` if it isn't.

ARGUMENTS

Data type	Name	Description
`dip_String`	`filename`	File name
`dip_Boolean *`	`veredict`	Set to `DIP_TRUE` or `DIP_FALSE`

SOFTWARE

This function uses `GifLib` (version 4.1.0 or later), which supports GIF 87a & 98a.
Copyright (c)1997 Eric S. Raymond

SEE ALSO

ImageWriteGIF, ImageReadGIF

ImageIsICS

Confirm that a file is an ICS file (in dipIO)

SYNOPSIS

```
#include "dipio_ics.h"
dip_Error dipio_ImageIsICS ( filename, veredict )
```

FUNCTION

This function verifies that the file is an ICS file. `veredict` is set to `DIP_TRUE` if it is, and to `DIP_FALSE` if it isn't.

ARGUMENTS

Data type	Name	Description
dip_String	filename	File name
dip_Boolean *	veredict	Set to DIP_TRUE or DIP_FALSE

SOFTWARE

This function uses `libics` (version 1.3), which supports the ICS specification revision 2.0. Copyright (c)2000-2002 Cris L. Luengo Hendriks, Dr. Hans T.M. van der Voort and many others.

SEE ALSO

ImageWriteICS, ImageReadICS

ImageIsJPEG

Confirm that a file is a JPEG file (in dipIO)

SYNOPSIS

```
#include "dipio_jpeg.h"
dip_Error dipio_ImageIsJPEG ( filename, veredict )
```

FUNCTION

This function verifies that the file is a JPEG file. `veredict` is set to `DIP_TRUE` if it is, and to `DIP_FALSE` if it isn't.

ARGUMENTS

Data type	Name	Description
`dip_String`	`filename`	File name
`dip_Boolean *`	`veredict`	Set to `DIP_TRUE` or `DIP_FALSE`

SOFTWARE

This function uses `libjpeg` (version 6b or later). Copyright (c)1994-1998, Thomas G. Lane.

SEE ALSO

ImageWriteJPEG, ImageReadJPEG, ImageReadJPEGInfo

ImageIsLSM

Confirm that a file is a Zeiss LSM file (in dipIO)

SYNOPSIS

```
#include "dipio_ics.h"
dip_Error dipio_ImageIsLSM ( filename, veredict )
```

FUNCTION

This function verifies that the file is a Zeiss LSM file. `veredict` is set to `DIP_TRUE` if it is, and to `DIP_FALSE` if it isn't.

ARGUMENTS

Data type	Name	Description
dip_String	filename	File name
dip_Boolean *	veredict	Set to DIP_TRUE or DIP_FALSE

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

SEE ALSO

ImageReadLSM

ImageIsTIFF

Confirm that a file is a TIFF file (in dipIO)

SYNOPSIS

```
#include "dipio_tiff.h"
dip_Error dipio_ImageIsTIFF ( filename, veredict )
```

FUNCTION

This function verifies that the file is a TIFF file. `veredict` is set to `DIP_TRUE` if it is, and to `DIP_FALSE` if it isn't.

ARGUMENTS

Data type	Name	Description
dip_String	filename	File name
dip_Boolean *	veredict	Set to DIP_TRUE or DIP_FALSE

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

SEE ALSO

ImageWriteTIFF, ImageReadTIFF

ImageNew

Allocate a structure

SYNOPSIS

`dip_Error dip_ImageNew( image, resources )`

FUNCTION

Allocates a `dip_Image` structure and initializes all fields to their default values. The resulting image is in the "raw" state, see The image structure. By using `ImageCopyProperties` and the "ImageSet" access functions, the image fields can be set to their desired values. Pixel data for the image can be allocated using the `ImageForge` function, which will will put the image in the "forged" state.

ARGUMENTS

Data type	Name	Description
`dip_Image *`	`image`	Used to return the newly allocated image
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

The image structure

`ImageFree`, `ImageForge`, `ImageStrip`, `ImageCopyProperties`

ImageRead

Read grey-value image from file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageRead ( image, filename, format, addExtensions, recognised )`

FUNCTION

This function reads an image from a file and puts it in `image`. `image` must be allocated before calling this function. If `format` is 0, all different `ImageRead` functions are called in sequence until the correct format has been found. If you know the format, get the correct format ID through the registry functions. See File formats recognized by dipIO for a list of currently supported formats.

The boolean `addExtensions` specifies whether `ImageRead` should try to add file format extensions to `filename`, if the registered file format reader fails to recognise `filename` straight away. The extensions are provided by the registered file readers.

If `recognised` is not zero, `ImageRead` will set it to `DIP_TRUE` when it has been able to read `filename`, and it will set it to `DIP_FALSE` when it is not able to read the file. No error will be generated in this case.

If the file contains a colour image, `Colour2Gray` is called. That is, this function always returns a grey-value image.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dip_int`	`format`	ID of file format
`dip_Boolean`	`addExtensions`	Add file format extensions to `filename`
`dip_Boolean *`	`recognised`	Pointer to boolean containing the file read status

SEE ALSO

`ImageReadColour`, `ImageReadROI`, `ImageFileGetInfo`, `ImageReadCSV`, `ImageReadGIF`, `ImageReadICS`, `ImageReadLSM`, `ImageReadPIC`, `ImageReadTIFF`, `ImageReadJPEG`, `ImageWrite`, `Colour2Gray`

ImageReadColour

Read colour image from file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageReadColour ( image, filename, photometric, format, addExtensions, recognised )`

FUNCTION

This function reads an image from a file and puts it in `image`. `image` must be allocated before calling this function. It works the same as `ImageRead`, except that, if the file contains a colour image, `Colour2Gray` is not called. The returned image has an extra dimension with colours (always the last dimension), and `photometric` is set to the colour space.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dipio_PhotometricInterpretation*	photometric	Photometric interpretation (==colour space)
dip_int	format	ID of file format
dip_Boolean	addExtensions	Add file format extensions to `filename`
dip_Boolean *	recognised	Pointer to boolean containing the file read status

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as `DIPIO_PHM_GREYVALUE`
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as `DIPIO_PHM_CMYK`

Most file formats support only some of these.

SEE ALSO

`ImageRead`, `ImageReadROI`, `ImageFileGetInfo`, `ImageReadCSV`, `ImageReadGIF`, `ImageReadICS`, `ImageReadLSM`, `ImageReadPIC`, `ImageReadTIFF`, `ImageReadJPEG`, `ImageWrite`, `Colour2Gray`

ImageReadCSV

Read comma-separated values from file (in dipIO)

SYNOPSIS

```
#include "dipio_csv.h"
dip_Error dipio_ImageReadCSV ( image, filename, separator )
```

FUNCTION

This function reads the comma-separated values from a file and puts it in `image`. `image` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
char	separator	Separator character

SEE ALSO

ImageRead, ImageWriteCSV

ImageReadCSVInfo

Get information about image in comma-separated values file (in dipIO)

SYNOPSIS

```
#include "dipio_csv.h"
dip_Error dipio_ImageReadCSVInfo ( imInfo, filename )
```

FUNCTION

Opens a comma-separated values (CSV) file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SEE ALSO

ImageFileGetInfo, ImageReadCSV, ImageWriteCSV, ImageFileInformationNew

ImageReadGIF

Read a GIF image from file (in dipIO)

SYNOPSIS

```
#include "dipio_gif.h"
dip_Error dipio_ImageReadGIF ( image, filename, photometric )
```

FUNCTION

This function reads an image from a GIF file and puts it in `image`. `image` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dipio_PhotometricInterpretation *	photometric	Photometric interpretation

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `GifLib` (version 4.1.0 or later), which supports GIF 87a & 98a. Copyright (c)1997 Eric S. Raymond

SEE ALSO

ImageRead, ImageReadColour, ImageWriteGIF, ImageIsGIF

ImageReadGIFInfo

Get information about image in GIF file (in dipIO)

SYNOPSIS

`#include "dipio_gif.h"`

`dip_Error dipio_ImageReadGIFInfo ( imInfo, filename )`

FUNCTION

Opens a GIF file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SOFTWARE

This function uses `GifLib` (version 4.1.0 or later), which supports GIF 87a & 98a. Copyright (c)1997 Eric S. Raymond

SEE ALSO

ImageFileGetInfo, ImageIsGIF, ImageReadGIF, ImageWriteGIF, ImageFileInformationNew

ImageReadICS

Read ICS image from file (in dipIO)

SYNOPSIS

`#include "dipio_ics.h"`

`dip_Error dipio_ImageReadICS ( image, filename, photometric, offset, roisize, sampling )`

FUNCTION

This function reads the image in the ICS file and puts it in `image`. `image` must be allocated before calling this function. `photometric` is set to match the photometric interpretation of the data in the file, if it is recognised. The colour dimension is always the last dimension of the image (no matter how it was saved in the ICS file). `offset`, `roisize` and `sampling` define a ROI to read in. See the comments in ImageReadROI for more information on this.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dipio_PhotometricInterpretation*	photometric	Photometric interpretation
dip_IntegerArray	offset	ROI offset
dip_IntegerArray	roisize	ROI size
dip_IntegerArray	sampling	ROI sampling rate

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libics` (version 1.3 or later), which supports the ICS specification revision 2.0. Copyright (c)2000-2002 Cris L. Luengo Hendriks, Dr. Hans T.M. van der Voort and many others.

This function uses `zlib` (version 1.1.4 or later). Copyright (c)1995-2002 Jean-loup Gailly and Mark Adler

SEE ALSO

ImageRead, ImageReadColour, ImageReadROI, ImageWriteICS, ImageIsICS

ImageReadICSInfo

Get information about image in ICS file (in dipIO)

SYNOPSIS

```
#include "dipio_ics.h"
dip_Error dipio_ImageReadICSInfo ( imInfo, filename )
```

FUNCTION

Opens a ICS file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SOFTWARE

This function uses `libics` (version 1.3), which supports the ICS specification revision 2.0. Copyright (c)2000-2002 Cris L. Luengo Hendriks, Dr. Hans T.M. van der Voort and many others.

SEE ALSO

ImageFileGetInfo, ImageIsICS, ImageReadICS, ImageWriteICS, ImageFileInformationNew

ImageReadJPEG

Read JPEG image from file (in dipIO)

SYNOPSIS

```
#include "dipio_jpeg.h"
dip_Error dipio_ImageReadJPEG ( image, filename, imageNumber, photometric )
```

FUNCTION

This function reads an image from the JPEG file and puts it in `image`. `image` must be allocated before calling this function. `photometric` is set to either `DIPIO_PHM_RGB` or `DIPIO_PHM_GREYVALUE`. If `photometric` is 0, the image will be read in as grey-value, even if color information is present in the file. Color images are allocated as 3D images, with the different samples along the 3rd. dimension.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dipio_PhotometricInterpretation *`	`photometric`	Photometric interpretation

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libjpeg` (version 6b or later). Copyright (c)1994-1998, Thomas G. Lane.

SEE ALSO

ImageRead, ImageReadColour, ImageWriteJPEG, ImageIsJPEG, ImageReadJPEGInfo, Colour2Gray

ImageReadJPEGInfo

Get information about image in JPEG file (in dipIO)

SYNOPSIS

```
#include "dipio_jpeg.h"
dip_Error dipio_ImageReadJPEGInfo ( imInfo, filename, imageNumber )
```

FUNCTION

Opens a JPEG file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SOFTWARE

This function uses `libjpeg` (version 6b or later). Copyright (c)1994-1998, Thomas G. Lane.

SEE ALSO

ImageFileGetInfo, ImageIsJPEG, ImageReadJPEG, ImageWriteJPEG, ImageFileInformationNew

ImageReadLSM

Read Zeiss LSM image from file (in dipIO)

SYNOPSIS

```
#include "dipio_lsm.h"
```

```
dip_Error dipio_ImageReadLSM ( image, filename, offset, roisize, sampling,
imInfo, resources )
```

FUNCTION

This function reads the image in the Zeiss LSM file and puts it in `image`. `image` must be allocated before calling this function. Depending on the recording mode and the number of channels recorded, an image with 2 to 5 dimensions is returned. If multiple channels were recorded, they will be put along the last dimension (which can be either the third, fourth or fifth). The "stack", "time series plane" ans "time series z-scan" recording modes return a 3D image, the "time series stack" returns a 4D image, all other modes return a 2D image (including the "line" mode).

`imInfo->physDims` contains information on the distance between pixels. `resources` is only used to allocate the `imInfo` structure, so if `imInfo` is 0, `resources` can be 0 too.

`offset`, `roisize` and `sampling` define a region of interest to read in. See the comments in `ImageReadROI` for more information on this. Note that the channel dimension is part of this ROI.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_IntegerArray	offset	ROI offset
dip_IntegerArray	roisize	ROI size
dip_IntegerArray	sampling	ROI sampling rate
dipio_ImageFileInformation*	imInfo	Image file information structure
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon

Graphics, Inc.

This function uses `zlib` (version 1.1.4 or later). Copyright (c)1995-2002 Jean-loup Gailly and Mark Adler

SEE ALSO

ImageRead, ImageReadROI, ImageIsLSM

ImageReadLSMInfo

Get information about image in LSM file (in dipIO)

SYNOPSIS

```
#include "dipio_lsm.h"
dip_Error dipio_ImageReadLSMInfo ( imInfo, filename )
```

FUNCTION

Opens a LSM file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

SEE ALSO

ImageFileGetInfo, ImageIsLSM, ImageReadLSM, ImageFileInformationNew

ImageReadPIC

Read BioRad PIC image from file (in dipIO)

SYNOPSIS

```
#include "dipio_pic.h"
```

```
dip_Error dipio_ImageReadPIC ( image, filename, offset, roisize, sampling,
info, resources )
```

FUNCTION

This function reads the image in the BioRAD PIC file and puts it in `image`. `image` must be allocated before calling this function. The information stored in the file is put in `info`.

`offset` and `roisize` define a region of interest to be read in. The ROI is clipped to the actual image data, so it is safe to specify a ROI that is too large. `sampling` can be used to read in a subset of the pixels of the chosen ROI. Any or all of these three parameters can be `NULL`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_IntegerArray	offset	ROI offset
dip_IntegerArray	roisize	ROI size
dip_IntegerArray	sampling	ROI sampling rate
dipio_ImageFileInformation *	info	File information
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ImageRead, ImageReadROI

ImageReadPICInfo

Get information about image in BioRad PIC file (in dipIO)

SYNOPSIS

```
#include "dipio_pic.h"
dip_Error dipio_ImageReadPICInfo ( imInfo, filename )
```

FUNCTION

Opens a BioRAD PIC file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name

SEE ALSO

ImageFileGetInfo, ImageReadPIC, ImageFileInformationNew

ImageReadROI

Read a portion of a grey-value image from file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageReadROI ( image, filename, offset, roisize, sampling, format, addExtensions, recognised )`

FUNCTION

This function reads an image from a file and puts it in `image`. `image` must be allocated before calling this function. It works the same as `ImageRead`, except that the user is allowed to specify a region of the image to read. This is done through the `offset` and `roisize` parameters. The ROI is clipped to the image size, so it is safe to specify a ROI that is too large. `sampling` can be used to read in a subset of the pixels of the chosen ROI. Any or all of these three parameters can be `NULL`.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_IntegerArray	offset	ROI offset
dip_IntegerArray	roisize	ROI size
dip_IntegerArray	sampling	ROI sampling rate
dip_int	format	ID of file format
dip_Boolean	addExtensions	Add file format extensions to `filename`
dip_Boolean *	recognised	Pointer to boolean containing the file read status

SEE ALSO

`ImageRead`, `ImageReadColour`, `ImageFileGetInfo`, `ImageReadCSV`, `ImageReadGIF`, `ImageReadICS`, `ImageReadLSM`, `ImageReadPIC`, `ImageReadTIFF`, `ImageReadJPEG`, `ImageWrite`, `Colour2Gray`

ImageReadTIFF

Read TIFF image from file (in dipIO)

SYNOPSIS

```
#include "dipio_tiff.h"
dip_Error dipio_ImageReadTIFF ( image, filename, imageNumber, photometric )
```

FUNCTION

This function reads an image from the TIFF file and puts it in `image`. `image` must be allocated before calling this function. `imageNumber` indicates which image from the multi-page TIFF file to read. 0 is the first image. `photometric` is set to match the photometric interpretation of the TIFF file. Colour images and multi-sample images are allocated as 3D images, with the different samples along the 3rd dimension.

Multi-page TIFF files in which all pages contain an image of the same size and type, can be read as a 3D or 4D (Colour along the 4th dimension) image by setting `imageNumber` to -1. If the images are not of the same size and type, an error will be generated.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_int	imageNumber	Image number to read
dipio_PhotometricInterpretation *	photometric	Photometric interpretation

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

This function uses `zlib` (version 1.1.4 or later). Copyright (c)1995-2002 Jean-loup Gailly and Mark Adler

KNOWN BUGS

TIFF is a very flexible file format. We have to limit the types of images that can be read to the more common ones, and to the ones dipIO writes. These are the most obvious limitations:

Tiled images are not supported.

Only 1, 4, 8, 16 and 32 bits per pixel integer grayvalues are read, as well as 32-bit and 64-bit floating point.

Only 4 and 8 bits per pixel colourmapped images are read. Colourmapped images contain 16-bit gray-values: stretching of the display will be necessary.

Class Y images (YCbCr) and Log-compressed images (LogLuv or LogL) are not supported.

SEE ALSO

ImageRead, ImageReadColour, ImageWriteTIFF, ImageIsTIFF, Colour2Gray

ImageReadTIFFInfo

Get information about image in TIFF file (in dipIO)

SYNOPSIS

```
#include "dipio_tiff.h"
dip_Error dipio_ImageReadTIFFInfo ( imInfo, filename, imageNumber )
```

FUNCTION

Opens a TIFF file and fills a `dipio_ImageFileInformation` structure with the information from that file. `imInfo` must be allocated before calling this function. `imageNumber` indicates which image from the multi-page TIFF file to get info on. 0 is the first image. `imInfo->numberOfImages` gives the number of pages in the file.

ARGUMENTS

Data type	Name	Description
dipio_ImageFileInformation	imInfo	Output image file information. See ImageFileInformationNew
dip_String	filename	File name
dip_int	imageNumber	Image number to query

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

SEE ALSO

ImageFileGetInfo, ImageIsTIFF, ImageReadTIFF, ImageWriteTIFF, ImageFileInformationNew

ImagesCheck

Check properties of several images

SYNOPSIS

`dip_Error dip_ImagesCheck( images, imageType, dataType, compareFlag, checkFlag )`

FUNCTION

This function checks whether the image type and the data type of all the images in the array match with the `imageType` and `dataType` variables, and compares selected properties of the first image with those of the other images in the array. This comparison is done by calling `ImagesCompareTwo`. The `checkFlag` can be used to compare properties not supported by `ImagesCompare`. An error is returned by `ImagesCheck` if a check or comparison fails.

ARGUMENTS

Data type	Name	Description
dip_ImageArray *	images	Array of Images
dip_ImageType	imageType	Image type of the first Image
dip_DataTypeProperties	dataType	Data type of the first Image. See DataTypeGetInfo
dipf_ImagesCompare	compareFlag	Properties to compare. See ImagesCompare
dipf_ImagesCheck	checkFlag	Extra properties to be compared

dipf_ImagesCheck

Name	Description
DIP_CKIM_MAX_PRECISION_MATCH	Check whether data types match or match to the DIP_GTP_MAX_PRECISION DataType
DIP_CKIM_CASTING_TYPE_MATCH	Check whether data types match or match to the DIP_GTP_CAST_R2C or DIP_GTP_CAST_C2R types of the first image in image
DIP_CKIM_IGNORE_NULL_DIM_IMAGES	Ignore images with a zero dimensionality, this flag is usefull when 0d images are used as generic data containers of constants

SEE ALSO

ImagesCompareTwo, ImagesCompare, ImagesCheckTwo

ImagesCheckTwo

Check properties of two images

SYNOPSIS

`dip_Error dip_ImagesCheckTwo( image1, image2, imageType, dataType, compareFlag, checkFlag )`

FUNCTION

This function checks whether the image type and the data type of the two images match with the `imageType` and `dataType` variables, and compares selected properties of the two images. This comparison is done by calling `ImagesCompareTwo`. The `checkFlag` can be used to compare properties not supported by `ImagesCompare`. `ImagesCheckTwo` returns an error code if a check or comparison fails.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image1`	First Image
`dip_Image`	`image2`	Second Image
`dip_ImageType`	`imageType`	Image type of the first Image
`dip_DataTypeProperties`	`dataType`	Data type of the first Image. See `DataTypeGetInfo`
`dipf_ImagesCompare`	`compareFlag`	Properties to compare. See `ImagesCompare`
`dipf_ImagesCheck`	`checkFlag`	Extra properties to be compared

dipf_ImagesCheck

Name	Description
`DIP_CKIM_MAX_PRECISION_MATCH`	Check whether data types match or match to the `DIP_GTP_MAX_PRECISION` DataType
`DIP_CKIM_CASTING_TYPE_MATCH`	Check whether data types match or match to the `DIP_GTP_CAST_R2C` or `DIP_GTP_CAST_C2R` types
`DIP_CKIM_IGNORE_NULL_DIM_IMAGES`	Ignore images with a zero dimensionality, this flag is usefull when 0d images are used as generic data containers of constants

SEE ALSO

`ImagesCompareTwo`, `ImagesCompare`, `ImagesCheck`

ImagesCompare

Compare properties of several images

SYNOPSIS

`dip_Error dip_ImagesCompare( images, condition, result )`

FUNCTION

This function compares some standard fields of a number of Images or performs a full comparison. Only if the comparison result is true between each of the Images, will the final comparison result be true. The condition parameter specifies which properties should be tested. If 0, a full comparison of the Images is performed. Otherwise it should be a logical OR of the `dipf_ImagesCompare` flags. `DIP_CPIM_MATCH_ALL_STANDARD` is equivalent to all the flags OR'ed together. The difference between `DIP_CPIM_MATCH_ALL_STANDARD` and the full comparison specified by 0, is that the first will compare all the standard fields (type, data type, dimensions), whereas the other compares all fields relevant to a particular DIPlib Image type. This may exclude some of the standard fields and include some fields particular to the type of DIPlib Image in question. There are two modes of operation. If the result parameter is set, it is used to store the result of the comparison, a set of OR'ed `dipf_ImagesCompare` flags. If the result parameter is 0, an error is returned if the condition parameter and the resulting set of flags are not the same.

ARGUMENTS

Data type	**Name**	**Description**
`dip_ImageArray`	`images`	Array of Images
`dipf_ImagesCompare`	`condition`	Properties to compare. 0 indicates full comparison
`dipf_ImagesCompare *`	`result`	Result: flags to indicate if the properties were the same. 0 indicates that an error should be returned if the requested properties do not match

dipf_ImagesCompare

Name	**Description**
`DIP_CPIM_DIMENSIONALITIES_MATCH`	Dimensionalities match
`DIP_CPIM_DIMENSIONS_MATCH`	Dimensions match. The comparison is done up to the lower of the of the two dimensionalities
`DIP_CPIM_SIZE_MATCH`	Combination of `DIP_CPIM_DIMENSIONALITIES_MATCH` and `DIP_CPIM_DIMENSIONS_MATCH`
`DIP_CPIM_TYPES_MATCH`	Types match
`DIP_CPIM_DATA_TYPES_MATCH`	Data types match
`DIP_CPIM_MATCH_ALL_STANDARD`	All flags above OR'ed together
`DIP_CPIM_STRIDES_MATCH`	Strides match
`DIP_CPIM_FULL_MATCH`	Full match. Returned in `result`. To test for a full match use 0. **Note:** This is NOT equivalent to the other flags OR'ed together, and it cannot be used as condition

SEE ALSO

ImagesCompareTwo, ImagesCheckTwo, ImagesCheck

ImagesCompareTwo

Compare properties of two images

SYNOPSIS

`dip_Error dip_ImagesCompareTwo( image1, image2, condition, result )`

FUNCTION

This function compares some standard fields of two Images or performs a full comparison. The condition parameter specifies which properties should be tested. See ImagesCompare for more information. There are two modes of operation. If the result parameter is set, it is used to store the result of the comparison, a set of OR'ed `dipf_ImagesCompare` flags. If the result parameter is 0, an error is returned if the condition parameter and the resulting set of flags are not the same.

ARGUMENTS

Data type	Name	Description
dip_Image	image1	First Image
dip_Image	image2	Second Image
dipf_ImagesCompare	condition	Properties to compare. See ImagesCompare
dipf_ImagesCompare*	result	Result: flags to indicate if the properties were the same. 0 indicates that an error should be returned if the requested properties do not match

SEE ALSO

ImagesCompare, ImagesCheckTwo, ImagesCheck

ImageSetDataType

Set the data type field

SYNOPSIS

`dip_Error dip_ImageSetDataType( image, dataType )`

FUNCTION

Set the `dip_Image` data type field. The image must be "raw".

ARGUMENTS

Data type	Name	Description
dip_DataType	image	An image
dip_DataType	type	The image data type

SEE ALSO

DIPlib's data types

The image structure

`ImageGetDataType`

ImageSetDimensions

Set the dimensions array

SYNOPSIS

`dip_Error dip_ImageSetDimensions( image, dimensions )`

FUNCTION

Set the `dip_Image` dimensions array. The image must be "raw".

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_IntegerArray	dimensions	The image dimensions

SEE ALSO

The image structure

ImageGetDimensions, ChangeDimensions

ImageSetType

Set the image type field

SYNOPSIS

`dip_Error dip_ImageSetType( image, type )`

FUNCTION

Set the `dip_Image` type field. The image must be “raw”.

ARGUMENTS

Data type	Name	Description
dip_Image	image	An image
dip_ImageType	type	The image type

SEE ALSO

The image structure `ImageGetType`

ImageSort

Sort image data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_ImageSort ( in, out, algorithm )
```

FUNCTION

Produces an image (out) with the sorted pixel values of in.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Sort	algorithm	Sort algorithm

The sortType parameter is one of:

Name	Description
DIP_SORT_DEFAULT	Default sort algorithm
DIP_SORT_QUICK_SORT	Quick sort
DIP_SORT_DISTRIBUTION_SORT	Distribution sort
DIP_SORT_INSERTION_SORT	Insertion sort

SEE ALSO

General information about sorting

DistributionSort, InsertionSort, QuickSort, Sort, SortIndices, SortIndices16, ImageSortIndices

ImageSortIndices

Sort indices to image data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_ImageSortIndices ( in, indices, algorithm, flags )
```

FUNCTION

Sorts a list of indices rather than the data itself using the algorithm specified by `algorithm`. Unless the `DIP_ISI_USE_INDICES`, the `indices` image will be initialised with one index for each pixel in the image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	indices	Indices
dip_Sort	algorithm	Sort algorithm
dipf_ImageSortIndices	flags	Flags

The `sortType` parameter is one of:

Name	Description
DIP_SORT_DEFAULT	Default sort algorithm
DIP_SORT_QUICK_SORT	Quick sort
DIP_SORT_DISTRIBUTION_SORT	Distribution sort
DIP_SORT_INSERTION_SORT	Insertion sort

The `dipf_ImageSortIndices` enumeration consists of the following flags:

Name	Description
DIP_ISI_USE_INDICES	Use the indices as given in the indices image

SEE ALSO

General information about sorting

DistributionSort, InsertionSort, QuickSort, Sort, ImageSort, SortIndices, SortIndices16

ImagesSeparate

Take care of in-place operations

SYNOPSIS

`dip_Error dip_ImagesSeparate( in, out, newOut, saved, resources )`

FUNCTION

First the list of output images is checked to see if any output image is used more than once. If this is the case an error is returned. Then the input and output images are examined. If any of the output images is also used as an input image, the function allocates a new image. This image is returned through the `newOut` array. For each output image a corresponding image is returned in this array. Either the original output image itself, or either a new image as discussed above. After the call to `dip_ImagesSeparate`, the images in the `newOut` array should be used instead of the original output images. After you are done processing the images, a call to `ResourcesFree` will perform the necessary post-processing. The post-processing consists of copying the data from the temporary output images to the original output images and freeing the temporary images. Because the post-processing is called through `ResourcesFree`, the resources parameter is mandatory. Any of the image arrays' elements may be set to zero, indicating that it is to be ignored.

The boolean `saved` array can be used to indicate that an input image has been stored in a safe place. In this case `dip_ImagesSeparate` will not have to allocate a temporary image if the input image is also used as an output image. The `saved` parameter may either be zero, which indicates that none of the input images has been saved, or it must be an array containing booleans corresponding each of the input images. `DIP_TRUE` indicates that the image has been saved.

ARGUMENTS

Data type	Name	Description
`dip_ImageArray`	`in`	An array of input images
`dip_ImageArray`	`out`	An array of output images
`dip_ImageArray *`	`newOut`	Returns an array containing the replacement output images
`dip_BooleanArray`	`saved`	An array of booleans indicating which input images are safely stored
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`. May not be zero

SEE ALSO

ImageGetData

ImageStrip

Restore an image to its initial ("raw") state

SYNOPSIS

```
dip_Error dip_ImageStrip( image )
```

FUNCTION

Free any pixel data associated with the image and return all fields to their initial ("raw") state. Essentially the image is returned to the state it was in right after it was allocated with ImageNew.

ARGUMENTS

Data type	Name	Description
dip_Image	image	The image to be stripped

SEE ALSO

The image structure

ImageNew, ImageForge, ImageFree, ImageCopyProperties

ImageWrite

Write grey-value image to file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageWrite ( image, filename, physDims, format, compression )`

FUNCTION

This function writes a grey-vlaue image to a file, overwriting any other file with the same name. `physDims` gives physical dimensions of the image, and can be set to 0 for default values. Not all file formats are able to store physical dimensions. Get the format ID through the registry functions. See File formats recognized by dipIO for a list of currently supported formats. If `format` is 0, ICSv2 is used.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_PhysicalDimensions	physDims	Physical dimensions structure. See PhysicalDimensionsNew
dip_int	format	ID of file format
dipio_Compression	compression	Compression method and level. See Compression methods for image files

SEE ALSO

ImageWriteColour, ImageWriteCSV, ImageWriteEPS, ImageWriteFLD, ImageWriteGIF, ImageWriteICS, ImageWritePS, ImageWriteTIFF, ImageWriteJPEG, ImageRead

ImageWriteColour

Write colour image to file (in dipIO)

SYNOPSIS

`dip_Error dipio_ImageWriteColour ( image, filename, photometric, physDims, format, compression )`

FUNCTION

This function writes a colour image to a file, overwriting any other file with the same name. `photometric` must be set to the correct value. Not all file formats support all photometric values, and some don't support colour at all. `physDims` gives physical dimensions of the image, and can be set to 0 for default values. Not all file formats are able to store physical dimensions. Get the format ID through the registry functions. See File formats recognized by dipIO for a list of currently supported formats. If `format` is 0, ICSv2 is used.

ARGUMENTS

Data type	**Name**	**Description**
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dipio_PhotometricInterpretation`	`photometric`	Photometric interpretation (==colour space)
`dip_PhysicalDimensions`	`physDims`	Physical dimensions structure. See `PhysicalDimensionsNew`
`dip_int`	`format`	ID of file format
`dipio_Compression`	`compression`	Compression method and level. See Compression methods for image files

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	**Description**
`DIPIO_PHM_GREYVALUE`	No colour information present; it's a grey-value image.
`DIPIO_PHM_RGB`	RGB image (the first three planes are red, green and blue)
`DIPIO_PHM_RGB_NONLINEAR`	Non-linear R'G'B' image (RGB channels to the power of 0.4)
`DIPIO_PHM_CMY`	CMY image (the first three planes are cyan, magenta and yellow)
`DIPIO_PHM_CMYK`	CMYK image (the first four planes are cyan, magenta, yellow and black)
`DIPIO_PHM_CIELUV`	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
`DIPIO_PHM_CIELAB`	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
`DIPIO_PHM_CIEXYZ`	CIE XYZ (the first three planes are X, Y and Z)
`DIPIO_PHM_CIEYXY`	CIE Yxy (the first three planes are Y, x and y)
`DIPIO_PHM_HCV`	HCV image (the first three planes are hue, chroma and value)
`DIPIO_PHM_HSV`	HSV image (the first three planes are hue, saturation and value)
`DIPIO_PHM_DEFAULT`	Same as `DIPIO_PHM_GREYVALUE`
`DIPIO_PHM_GENERIC`	Anything can be coded in the channels; the same as `DIPIO_PHM_CMYK`

Most file formats support only some of these.

SEE ALSO

`ImageWrite`, `ImageWriteCSV`, `ImageWriteEPS`, `ImageWriteFLD`, `ImageWriteGIF`, `ImageWriteICS`, `ImageWritePS`, `ImageWriteTIFF`, `ImageWriteJPEG`, `ImageRead`, `Colour2Gray`

ImageWriteCSV

Write image to a comma-separated-value file (in dipIO)

SYNOPSIS

`#include "dipio_csv.h"`

`dip_Error dipio_ImageWriteCSV ( image, filename, separator )`

dip_Error dipio_ImageWriteCSV (dip_Image, dip_String, char);

FUNCTION

This function writes the image to a comma-separated-values file, overwriting any other file with the same name. Optionally, an other separator than the comma can be specified using `separator`. Sometimes a space, a tab or a colon are used instead. Each line of image data is ended by a newline.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`char`	`separator`	Optional alternative separator character

SEE ALSO

ImageWrite, ImageReadCSV

ImageWriteEPS

Write image to Encapsulated PostScript file (in dipIO)

SYNOPSIS

```
#include "dipio_ps.h"
dip_Error dipio_ImageWriteEPS ( image, filename, photometric, xcm, ycm, border )
```

FUNCTION

This function writes the image to an Encapsulated PostScript file, overwriting any other file with the same name. Set the image size in `xcm` and `ycm`. `border` sets the size of the border around the image. If `border` is 0, no border is drawn. For colour images, set `photometric` (supported are RGB and CMYK) and write the colour channels along the third image dimension.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dipio_PhotometricInterpretation	photometric	Photometric interpretation
dip_float	xcm	X-size of image in cm.
dip_float	ycm	Y-size of image in cm.
dip_int	border	Thickness of border, zero is no border

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SEE ALSO

ImageWrite, ImageWriteColour, ImageWritePS

ImageWriteFLD

Write image to AVS field file (in dipIO)

SYNOPSIS

```
#include "dipio_fld.h"
dip_Error dipio_ImageWriteFLD ( image, filename )
```

FUNCTION

This function writes the image to an AVS Field file.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name

SEE ALSO

ImageWrite

ImageWriteGIF

Write image to a GIF file (in dipIO)

SYNOPSIS

```
#include "dipio_gif.h"
dip_Error dipio_ImageWriteGIF ( image, filename, labelImage )
```

FUNCTION

This function writes the gray-value image to a GIF file, overwriting any other file with the same name. Optionally, an integer-typed image can be identified as a labeled image using `labelImage`. In that case a colour GIF image will be saved.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dip_Boolean	labelImage	Regard an integer image as a labeled image

SOFTWARE

This function uses `GifLib` (version 4.1.0 or later), which supports GIF 87a & 98a.
Copyright (c)1997 Eric S. Raymond

SEE ALSO

ImageWrite, ImageReadGIF, ImageIsGIF

ImageWriteICS

Write ICS image to file (in dipIO)

SYNOPSIS

```
#include "dipio_ics.h"
dip_Error dipio_ImageWriteICS ( image, filename, photometric, physDims,
history, sigbits, version, compression )
```

FUNCTION

This function writes the image to an ICS file, overwriting any other file with the same name. `version` can set to 1 to use the ICS v.1.0 file format (the 2-file version), instead of ICS v.2.0. For colour images, set `photometric` and write the colour channels along the last image dimension. Set `sigbits` only if the number of significant bits is different from the full range of the data type of `image` (use 0 otherwise). `physDims` can be set to 0 to fill out default values. `history` can be 0 if you do not want to bother.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dipio_PhotometricInterpretation`	`photometric`	Photometric interpretation
`dip_PhysicalDimensions`	`physDims`	Physical dimensions structure. See `PhysicalDimensionsNew`
`dip_StringArray`	`history`	Tags that are written to the history in the ICS header
`dip_int`	`sigbits`	Number of significant bits.
`dip_int`	`version`	ICS version
`dipio_Compression`	`compression`	Compression method and level. See Compression methods for image files

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libics` (version 1.3 or later), which supports the ICS specification revision 2.0. Copyright (c)2000-2002 Cris L. Luengo Hendriks, Dr. Hans T.M. van der Voort and many others.

This function uses `zlib` (version 1.1.4 or later). Copyright (c)1995-2002 Jean-loup Gailly and Mark Adler

SEE ALSO

`ImageWrite`, `ImageWriteColour`, `ImageReadICS`, `ImageIsICS`

ImageWriteJPEG

Write JPEG image to file (in dipIO)

SYNOPSIS

`#include "dipio_jpeg.h"`

`dip_Error ImageWriteJPEG ( image, filename, photometric, physDims, complevel )`

FUNCTION

This function writes the image to a JPEG file, overwriting any other file with the same name. `photometric` can set to let the function know how to write the JPEG image (supported colour space is RGB).

If `photometric` is not `DIPIO_PHM_GRAYVALUE`, a 3D image is expected, in which the different planes are stored along the 3rd dimension.

`physDims` gives physical dimensions of the image, which will be used to set the dots per inch property of the JPEG file. It can be set to 0 for default values (300 dpi). If the `physDims->dimensionUnits` is not given, meters are assumed.

`complevel` is a number between 1 (worst quality, smallest files) and 100 (best quality, largest files). Setting `complevel` to 0 uses the default compression level, which is 90.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dipio_PhotometricInterpretation`	`photometric`	Photometric interpretation
`dip_PhysicalDimensions`	`physDims`	Physical dimensions structure. See `PhysicalDimensionsNew`
`dipio_uint`	`complevel`	Compression level

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libjpeg` (version 6b or later). Copyright (c)1994-1998, Thomas G. Lane.

SEE ALSO

ImageWrite, ImageWriteColour, ImageReadJPEG, ImageIsJPEG, ImageReadJPEGInfo

ImageWritePS

Write image to PostScript file (in dipIO)

SYNOPSIS

```
#include "dipio_ps.h"
```

```
dip_Error dipio_ImageWritePS ( image, filename, photometric, caption, xcm,
ycm, border )
```

FUNCTION

This function writes the image to a PostScript file, overwriting any other file with the same name. Set the image size in `xcm` and `ycm`. `border` sets the size of the border around the image. If `border` is 0, no border is drawn. You can give the page a title through `caption`. For colour images, set `photometric` (supported are RGB and CMYK) and write the colour channels along the third image dimension.

ARGUMENTS

Data type	**Name**	**Description**
`dip_Image`	`image`	Output image
`dip_String`	`filename`	File name
`dipio_PhotometricInterpretation`	`photometric`	Photometric interpretation
`dip_String`	`caption`	Title for page
`dip_float`	`xcm`	X-size of image on page, in cm.
`dip_float`	`ycm`	Y-size of image on page, in cm.
`dip_int`	`border`	Thickness of border, zero is no border

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SEE ALSO

ImageWrite, ImageWriteColour, ImageWriteEPS

ImageWriteTIFF

Write TIFF image to file (in dipIO)

SYNOPSIS

```
#include "dipio_tiff.h"
dip_Error ImageWriteTIFF ( image, filename, photometric, physDims,
compression )
```

FUNCTION

This function writes the image to a TIFF file, overwriting any other file with the same name. `photometric` can set to let the function know how to write the TIFF image (supported colour spaces are RGB, CIE Lab and CMYK).

If `photometric` is not `DIPIO_PHM_GRAYVALUE`, a 3D image is expected, in which the different planes are stored along the 3rd dimension.

`physDims` gives physical dimensions of the image, which will be used to set the dots per inch property of the TIFF file. It can be set to 0 for default values (300 dpi). If the `physDims->dimensionUnits` is not given, meters are assumed.

ARGUMENTS

Data type	Name	Description
dip_Image	image	Output image
dip_String	filename	File name
dipio_PhotometricInterpretation	photometric	Photometric interpretation
dip_PhysicalDimensions	physDims	Physical dimensions structure. See PhysicalDimensionsNew
dipio_Compression	compression	Compression method and level. See Compression methods for image files

The enumerator `dipio_PhotometricInterpretation` contains the following constants:

Name	Description
DIPIO_PHM_GREYVALUE	No colour information present; it's a grey-value image.
DIPIO_PHM_RGB	RGB image (the first three planes are red, green and blue)
DIPIO_PHM_RGB_NONLINEAR	Non-linear R'G'B' image (RGB channels to the power of 0.4)
DIPIO_PHM_CMY	CMY image (the first three planes are cyan, magenta and yellow)
DIPIO_PHM_CMYK	CMYK image (the first four planes are cyan, magenta, yellow and black)
DIPIO_PHM_CIELUV	CIE L*u'v' image (the first three planes are luminosity, u* and v*)
DIPIO_PHM_CIELAB	CIE L*a*b* image (the first three planes are luminosity, a* and b*)
DIPIO_PHM_CIEXYZ	CIE XYZ (the first three planes are X, Y and Z)
DIPIO_PHM_CIEYXY	CIE Yxy (the first three planes are Y, x and y)
DIPIO_PHM_HCV	HCV image (the first three planes are hue, chroma and value)
DIPIO_PHM_HSV	HSV image (the first three planes are hue, saturation and value)
DIPIO_PHM_DEFAULT	Same as DIPIO_PHM_GREYVALUE
DIPIO_PHM_GENERIC	Anything can be coded in the channels; the same as DIPIO_PHM_CMYK

Most file formats support only some of these.

SOFTWARE

This function uses `libtiff` (version 3.6.1 or later), which supports the TIFF specification revision 6.0. Copyright (c)1988-1997 Sam Leffler and Copyright (c)1991-1997 Silicon Graphics, Inc.

This function uses `zlib` (version 1.1.4 or later). Copyright (c)1995-2002 Jean-loup Gailly and Mark Adler

SEE ALSO

`ImageWrite`, `ImageWriteColour`, `ImageReadTIFF`, `ImageIsTIFF`

Imaginary

Arithmetic function

SYNOPSIS

`dip_Error dip_Imaginary ( in, out )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

Computes the imaginary part of the input image values, and outputs a float typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Modulus, Phase, Real

IncoherentOTF

Generates an incoherent OTF

SYNOPSIS

```
#include "dip_microscopy.h"
dip_Error dip_IncoherentOTF ( out, defocus, xNyquist, amplitude, otf )
```

DATA TYPES

Output: sfloat

FUNCTION

This function implements the formulae for a (defocused) incoherent OTF as described by Castleman. When `defocus` is unequal to zero, either the Stokseth approximation or the Hopkins approximation is used. The `defocus` is defined a the maximum defocus path length error divided by the wave length (See Castleman for details). The summation over the Bessel functions in the Hopkins formluation, is stopped when the change is smaller than `DIP_MICROSCOPY_HOPKINS_OTF_CUTOFF`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output
dip_float	defocus	Defocus
dip_float	xNyquist	Oversampling
dip_float	amplitude	Amplitude
dipf_IncoherentOTF	otf	Otf approximation

The `dipf_IncoherentOTF` enumeration supports the following flags:

Name	Description
DIP_MICROSCOPY_OTF_STOKSETH	Stokseth OTF approximation
DIP_MICROSCOPY_OTF_HOPKINS	Hopkins OTF approximation

LITERATURE

K.R. Castleman, *"Digital image processing, second edition"*, Prentice Hall, Englewood Cliffs, 1996.

SEE ALSO

IncoherentPSF

IncoherentPSF

Generates an incoherent PSF

SYNOPSIS

```
#include "dip_microscopy.h"
dip_Error dip_IncoherentPSF ( output, xNyquist, amplitude )
```

DATA TYPES

Output: sfloat

FUNCTION

This function generates an incoherent in-focus point spread function of a diffraction limited objective.

ARGUMENTS

Data type	Name	Description
dip_Image	output	Output Image
dip_float	xNyquist	Oversampling Factor
dip_float	amplitude	Amplitude

LITERATURE

K.R. Castleman, *"Digital image processing, second edition"*, Prentice Hall, Englewood Cliffs, 1996.

SEE ALSO

IncoherentOTF

IndexToCoordinate

Convert pixel index to coordinate

SYNOPSIS

```
#include "dip_coordsindx.h"
dip_Error dip_IndexToCoordinate ( index, coordinate, stride )
```

FUNCTION

This function is identical to IndexToCoordinateWithSingletons, but does not handle images with singleton dimensions (dimensions where the size is 1). Please use the other function instead, this one is provided for backwards compatability only.

ARGUMENTS

Data type	Name	Description
dip_int	index	lineair index
dip_IntegerArray	coordinate	output coordinates
dip_IntegerArray	stride	stride array

SEE ALSO

IndexToCoordinateWithSingletons, CoordinateToIndex

IndexToCoordinateWithSingletons

Convert pixel index to coordinate

SYNOPSIS

```
#include "dip_coordsindx.h"
dip_Error dip_IndexToCoordinateWithSingletons ( index, coordinate, size,
stride )
```

FUNCTION

This function converts an pixel `index` of an image to a `coordinate` array. The conversion is done by calculating the modulus of the index with the `stride` and `size` arrays obtained from the image. `coordinate` has to be an allocated integer array with its size equal to the size of `stride` and `size`.

A set of macros can be used instead of this function to avoid some overhead when repeatedly converting linear indices to coordinates for the same image:

```
DIP_FNR_DECLARE;   /* Declares  dip_Registry rg */

dip_Image image;
dip_int index;
dip_IntegerArray coordinates;

dip_IntegerArray size;
dip_IntegerArray stride;

DIP_INDEX_TO_COORDINATE_DECL( ix ); /* This macro declares variable "ix", name it whatever

DIP_FNR_INITIALISE;

/* ... */

DIPXJ( dip_ImageGetDimensions( image, &size, rg ));
DIPXJ( dip_ImageGetStride( image, &stride, rg ));

DIP_INDEX_TO_COORDINATE_INIT( size, stride, ix, rg ); /* This macro initialises variable "

DIPXJ( dip_IntegerArrayNew( &coordinates, stride->size, 0, rg ));

/* Now, every time you need to obtain the coordinates for an index, do: */
```

```
DIP_INDEX_TO_COORDINATE( index, coordinates, stride, ix );
```

ARGUMENTS

Data type	Name	Description
dip_int	index	lineair index
dip_IntegerArray	coordinate	output coordinates
dip_IntegerArray	size	image size array
dip_IntegerArray	stride	stride array

SEE ALSO

IndexToCoordinate, CoordinateToIndex

Initialise

Initialise DIPlib

SYNOPSIS

```
dip_Error dip_Initialise( void )
dip_Error dipio_Initialise( void )
```

FUNCTION

Initialise the DIPlib library. Must be called before using any of the other DIPlib functions. This function can be invoked more than once; all but the first invocation are ignored.

SEE ALSO

Exit

InsertionSort

Sort a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_InsertionSort ( data, size, dataType )
```

FUNCTION

Sorts a block of data (of size `size` and data type `dataType`) using the insertion sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

InsertionSortIndices, InsertionSortIndices16, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

InsertionSortIndices

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_InsertionSortIndices ( data, indices, size, dataType )
```

FUNCTION

Sorts a list of indices rather than the data itself using the insertion sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint32 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type, See DIPlib's data types

SEE ALSO

General information about sorting

InsertionSort, InsertionSortIndices16, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

InsertionSortIndices16

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_InsertionSortIndices16 ( data, indices, size, dataType )
```

FUNCTION

Sorts a list of (16 bit) indices rather than the data itself using the insertion sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint16 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

InsertionSort, InsertionSortIndices, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

IntegerArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_IntegerArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the integer array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	dest	Destination array
dip_IntegerArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

IntegerArrayNew, IntegerArrayFree, IntegerArrayCopy, IntegerArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

IntegerArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_IntegerArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and "returns" its index in the array. If `found` is zero, `IntegerArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
dip_IntegerArray	array	Array to find value in
dip_int	value	Value to find
dip_int *	index	Index of the found value
dip_Boolean *	found	Value found or not

SEE ALSO

IntegerArrayNew, IntegerArrayFree, IntegerArrayCopy, IntegerArrayFind

IntegerArrayFind, FloatArrayFind, ComplexArrayFind, DataTypeArrayFind, BooleanArrayFind, VoidPointerArrayFind

IntegerArrayFree

Array free function

SYNOPSIS

`dip_Error dip_IntegerArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	array	Array

SEE ALSO

IntegerArrayNew, IntegerArrayFree, IntegerArrayCopy, IntegerArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

IntegerArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_IntegerArrayNew ( array, size, value, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_IntegerArray` and sets the size of the array to `size`. Each array element is initialized with `value`.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	array	Array
dip_int	size	Size
dip_int	value	Initial value
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

IntegerArrayNew, IntegerArrayFree, IntegerArrayCopy, IntegerArrayFind

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

Invert

logic operation

SYNOPSIS

`dip_Error dip_Invert ( in, out )`

DATA TYPES

binary, **integer**

FUNCTION

The function `Invert` inverts the pixel value in `in1` and stores the result in `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Binary input image
dip_Image	out	Output image

SEE ALSO

And, Xor, Or

IsodataThreshold

Point operation

SYNOPSIS

```
#include "dip_analysis.h"
dip_Error dip_IsodataThreshold ( in, out, mask, numbthresholds, values )
```

DATA TYPES

integer, **float**

FUNCTION

Thresholds `in` with the isodata method. Several threholds can be supplied, their value is returned in `values`. The different regions are label in `out` with different grey-values. A maks image `mask` can be given to compute the isodata only there.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	mask	Mask image
dip_int	numbthresholds	Number of Thresholds
dip_FloatArray	values	Values

SEE ALSO

Threshold, RangeThreshold, HysteresisThreshold

IsScalar

Determines whether an image is a scalar

SYNOPSIS

```
dip_Error dip_IsScalar( image, answer )
```

FUNCTION

Determines whether an image is of the `DIP_IMTP_SCALAR` type. If `answer` is not zero, the verdict is passed in this variable. Otherwise, `dip_IsScalar` returns an error in case `image` fails to be a scalar.

ARGUMENTS

Data type	Name	Description
dip_Image	image	The image under investigation
dip_Boolean *	answer	The verdict

Kuwahara

Edge perserving smoothing filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_Kuwahara ( in, out, se, boundary, param, shape )
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function implements the kuwahara edge-preserving smoothing function. See section 9.4, “Smoothing operations”, in Fundamentals of Image Processing for a description of the algorithm. However, this function does not implement the classical kuwahara filter, which only compares the variance of four regions in the filter window. Instead, it compares the variance of every region specified by the filter shape and size centered within the filter window.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The “on” pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use se as filter window, can be any size

SEE ALSO

GeneralisedKuwahara, KuwaharaImproved, GeneralisedKuwaharaImproved, VarianceFilter, Uniform

KuwaharaImproved

Edge perserving smoothing filter

SYNOPSIS

```
#include "dip_filtering.h"
```

```
dip_Error dip_KuwaharaImproved ( in, out, se, boundary, param, shape,
threshold )
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function implements an improved version of `Kuwahara`, see that function's description for more information. This function adds a `threshold` parameter that avoids false edges in uniform regions. If the difference between maximal and minimal variance within the filter window is smaller or equal to `threshold`, the centre pixel is taken, instead of the minimum. Setting `threshold` to zero yields the same result as `Kuwahara`.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape
dip_float	threshold	Minimal variance difference within window

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

Kuwahara, GeneralisedKuwahara, GeneralisedKuwaharaImproved, VarianceFilter, Uniform

Label

Label a binary image

SYNOPSIS

```
#include "dip_regions.h"
dip_Error dip_Label ( in, out, connectivity, flags, minsize, maxsize, nol,
boundary )
```

DATA TYPES

binary

FUNCTION

The output is an integer image. Each object (respecting the connectivity, see The connectivity parameter) in the input image receives a unique number. This number ranges from 1 to the number of objects in the image. The pixels in the output image corresponding to a given object are set to this number (label). The remaining pixels in the output image are set to 0. The `minsize` and `maxsize` set limits on the size of the objects, if the flag `DIP_LB_THRESHOLD_ON_SIZE` is set: Objects smaller than `minsize` or larger than `maxsize` do not receive a label and the corresponding pixels in the output image are set to zero. Setting `minsize` to zero implies that there is no check with respect to the minimum size of the object, and the same holds for `maxsize` and the maximum size of the object. If the flag `DIP_LB_LABEL_IS_SIZE` is set, the objects' labels are set to the objects' sizes. The boundary conditions are generally ignored (labeling stops at the boundary). The exception is `DIP_BC_PERIODIC`, which is the only one that makes sense for this algorithm.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input binary image
dip_Image	out	Output label image
dip_int	connectivity	Connectivity
dip_int	flags	0, or a logical OR of the flags described above
dip_int	minsize	Minimum size of the objects (0=do not check)
dip_int	maxsize	Maximum size of the objects (0=do not check)
dip_int *	nol	Pointer to dip_int. Used for returning the number of objects. May be set to 0.
dip_BoundaryArray	boundary	Boundary conditions

Laplace

Second order derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_Laplace ( in, out, boundary, ps, sigmas, tc, flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes the Laplace of an image using the `Derivative` function.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_BooleanArray`	`ps`	Dimensions to process
`dip_FloatArray`	`sigmas`	Sigma of Gaussian
`dip_float`	`tc`	Truncation of Gaussian, see `GlobalGaussianTruncationGet`
`dip_DerivativeFlavour`	`flavour`	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
`DIP_DF_DEFAULT`	Default derivative flavour (==`DIP_DF_FIRGAUSS`)
`DIP_DF_FIRGAUSS`	Gaussian family, FIR implementation, `Gauss`
`DIP_DF_IIRGAUSS`	Gaussian family, IIR implementation, `GaussIIR`
`DIP_DF_FTGAUSS`	Gaussian family, FT implementation, `GaussFT`
`DIP_DF_FINITEDIFF`	Finite difference implementation, `FiniteDifferenceEx`

SEE ALSO

See section 9.5, "Derivative-based operations", in Fundamentals of Image Processing.

`Derivative`, `GradientMagnitude`, `GradientDirection2D`, `Dgg`, `LaplacePlusDgg`, `LaplaceMinDgg`

LaplaceMinDgg

Second order derivative filter

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_LaplaceMinDgg ( in, out, boundary, ps, sigmas, tc, flavour )
```

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes Laplace - Dgg. For two-dimensional images this is equivalent to the second order derivative in the direction perpendicular to the gradient direction.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_float	tc	Truncation of Gaussian, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

`Derivative`, `GradientMagnitude`, `GradientDirection2D`, `Laplace`, `Dgg`, `LaplacePlusDgg`

LaplacePlusDgg

Second order derivative filter

SYNOPSIS

`#include "dip_derivatives.h"`

`dip_Error dip_LaplacePlusDgg ( in, out, boundary, ps, sigmas, tc, flavour )`

DATA TYPES

Depends on the underlying implementation, but expect:

binary, integer, **float**

FUNCTION

Computes the laplace and the second derivative in gradient direction of an image using the `Derivative` function and adds the results. The zero-crossings of the result correspond to the edges in the image, just as for the individual Laplace and Dgg operators. The localization is improved by an order of magnitude with respect to the individual operators.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	ps	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_float	tc	Truncation of Gaussian, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative flavour

The enumerator `flavour` parameter is one of:

Name	Description
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

LITERATURE

Lucas J. van Vliet, *"Grey-Scale Measurements in Multi-Dimensional Digitized Images"*, Delft University of Technology, 1993

P.W. Verbeek and L.J. van Vliet, *"On the location error of curved edges in low-pass filtered 2-D and 3-D images"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 7, 1994, 726-733.

SEE ALSO

`Derivative`, `GradientMagnitude`, `GradientDirection2D`, `Laplace`, `Dgg`, `LaplaceMinDgg`

Lee

Morphological edge detector

SYNOPSIS

`#include "dip_morphology.h"`

`dip_Error dip_Lee ( in, out, se, boundary, param, shape, edgeType, flags )`

DATA TYPES

integer, **float**

FUNCTION

Implements a morphological edge detector based on the minimum of two complementary morphological operations. These can be chosen through the `edgeType` parameter.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dip_MphEdgeType	edgeType	Edge type
dipf_LeeSign	flags	Lee sign flag

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

The enumerator `dip_MphEdgeType` contains the following constants:

Name	Description
DIP_MPH_TEXTURE	Response is limited to edges in texture
DIP_MPH_OBJECT	Response is limited to object edges
DIP_MPH_BOTH	All edges produce equal response

The enumerator `dipf_LeeSign` contains the following constants:

Name	Description
DIP_LEE_UNSIGNED	Absolute edge strength
DIP_LEE_SIGNED	Signed edge strength

SEE ALSO

`MorphologicalGradientMagnitude`, `MorphologicalRange`, `MultiScaleMorphologicalGradient`, `Tophat`

Lesser

Compare grey values in two images

SYNOPSIS

`dip_Error dip_Lesser ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to "true" when for corresponding pixels `in1` < `in2`. This is the same as `Compare` with the `DIP_SELECT_LESSER` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to a functionality similar to that of `Threshold`.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

`Compare`, `Threshold`, `Equal`, `Greater`, `NotEqual`, `NotGreater`, `NotLesser`, `SelectValue`, `NotZero`

Ln

arithmetic function

SYNOPSIS

`dip_Error dip_Ln ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the natural logarithm of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp, Exp2, Exp10, Log2, Log10

LnGamma

mathematical function

SYNOPSIS

`dip_Error dip_LnGamma ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the natural logarithm of the gamma function of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselY1, BesselYN, Erf, Erfc, Sinc

LnNormError

difference measure

SYNOPSIS

`dip_Error dip_LnNormError ( in1, in2, mask, out, order )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Calculates the `order` norm difference between each pixel value of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	mask	Mask
dip_Image	out	Output
dip_float	order	Order

SEE ALSO

MeanError, MeanSquareError, RootMeanSquareError, MeanAbsoluteError, IDivergence

LocalMinima

Marks local minima (or regional minima)

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_LocalMinima ( in, mask, out, connectivity, max_depth, max_size,
binaryOutput )
```

DATA TYPES

integer, **float**

FUNCTION

The binary output image is true on all pixels belonging to the minima of a region (as defined by the watershed). To find local maxima, use the inverse of the image as input to this function (see `Invert`). If `binaryOutput` is `DIP_FALSE`, the output is a labelled image instead of a binary one. In this case, pixels belonging to the same local minimum are assigned the same value.

The algorithm is based on the watershed transform, see `Watershed` for information on the parameters.

`Minima` is a different algorithm to obtain local minima; `Maxima` yields the local maxima.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Mask
dip_Image	out	Output (binary)
dip_int	connectivity	Connectivity
dip_float	max_depth	Maximum depth of a region that can be merged
dip_int	max_size	Maximum size of a region that can be merged
dip_Boolean	binaryOutput	DIP_FALSE if the output should be a labelled image

SEE ALSO

`Watershed`, `SeededWatershed`, `UpperEnvelope`, `Minima`, `Maxima`

Log10

arithmetic function

SYNOPSIS

`dip_Error dip_Log10 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the base ten logarithm of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp, Exp2, Exp10, Ln, Log2

Log2

arithmetic function

SYNOPSIS

`dip_Error dip_Log2 ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the base two logarithm of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sqrt, Exp, Exp2, Exp10, Ln, Log10

macros.h

Various macros

DESCRIPTION

The include files `dip_macros.h` contains a number of useful macros.

Math macros

`DIP_ABS( x )`	Absolute value of `x`
`DIP_MAX( x, y )`	Maximum of `x` and `y`
`DIP_MIN( x, y )`	Minimum of `x` and `y`
`DIP_FUNC( funcName, suffix )`	Attaches the suffix to the function name, and puts and underscore in between.
`DIP_SWAP( x, y, z )`	Swaps variables `x` and `y`, using temporary variable `z`. Must be followed by a trailing ";"

Macros for handling complex numbers:

`DIP_REAL( x )`	Real part of complex number `x`
`DIP_IMAGINARY( x )`	Imaginary part of complex number `x`
`DIP_SQUARE_MODULUS( x )`	Square modulus of complex number `x`
`DIP_MODULUS( x )`	Modulus of complex number `x`
`DIP_PHASE( x )`	Phase of complex number `x`

Binary I/O macros

`DIP_BINARY_MASK( mask, plane )`	Computes a binary `mask` from the `plane` value
`DIP_BINARY_READ( in, mask )`	Returns the binary value from `in`
`DIP_BINARY_WRITE( out, val, mask )`	Writes the value of `val` to `out`

Random access I/O macros:

```
DIP_PIXEL_GET( ip, pos, stride, value )
DIP_PIXEL_SET( ip, pos, stride, value )
```

get/set the `value` of the pixel at position `pos` from data pointer `ip` with strides `stride`. Both `pos` and `stride` are `dip_IntegerArrays`.

```
DIP_PIXEL_ADD( ip, pos, stride, value )
DIP_PIXEL_SUB( ip, pos, stride, value )
DIP_PIXEL_MUL( ip, pos, stride, value )
DIP_PIXEL_DIV( ip, pos, stride, value )
```

add/subtract/multlipy/divide the `value` with the pixel-value at position `pos` from data pointer `ip` with strides `stride`. Both `pos` and `stride` are `dip_IntegerArrays`.

Map

Remaps an image

SYNOPSIS

`#include "dip_manipulation.h"`

`dip_Error dip_Map ( in, out, map, mirror )`

DATA TYPES

binary, integer, float, complex

FUNCTION

This function maps the dimensions of the output image to (different) dimensions of the input image. The array index of `map` specifies the dimension of the output image, the value of the array element of `map` specifies to which dimension in the input image it corresponds. Optionally, the dimensions can be mirrored, when the value of the corresponding array element in `mirror` is set to `DIP_TRUE`. The mirror operation is performed after the mapping operation.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_IntegerArray`	`map`	Map array
`dip_BooleanArray`	`mirror`	Mirror array

SEE ALSO

Mirror

Max

arithmetic function

SYNOPSIS

`dip_Error dip_Max ( in1, in2, out )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = max (`in1` , `in2`) on a pixel by pixel basis. The data types of the `in1` and `in2` image may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

MaxFloat, Min, MinFloat

MaxFloat

arithmetic function

SYNOPSIS

`dip_Error dip_MaxFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = max(`in` , `constant`) on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

Max, Min, MinFloat

Maxima

Detects local maxima

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_Maxima ( in, mask, out, connectivity, booleanOutput )
```

DATA TYPES

integer, float

FUNCTION

This function detects local maxima.

The algorithm finds a connected set of pixels with identical value, an no neighbours with higher value. This set is a local maximum and its pixels are set to 1 in the output image. If `booleanOutput` is false, the output image is a labelled image.

For images that have large plateaus (regions of constant value) that are not local maxima, this function can be quite slow. For example, an image that is zero everywhere except for a small peak somewhere. For such an image it is recommended to use the `mask` input, for example with the output of a threshold operation.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	mask	Mask image
dip_Image	out	Binary output image
dip_int	connectivity	Connectivity
dip_Boolean	booleanOutput	Give a binary output image?

NOTE

If you are looking for the old version of `Maxima`, it is still available through the following combination of commands:

```
dip_Dilation( in, out, se, boundary, param, shape );
dip_Equal( in, out, out );
```

SEE ALSO

Minima, SubpixelMaxima, LocalMinima, SeededWatershed, GrowRegions

Maximum

statistics function

SYNOPSIS

`dip_Error dip_Maximum ( in, mask, out, ps )`

DATA TYPES

binary, integer, float

FUNCTION

Calculates the maximum of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Minimum, Median, Percentile

mBesselJ0

mathematical function

SYNOPSIS

`dip_float dipm_BesselJ0 ( x )`

FUNCTION

Computes the Bessel function J0 of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mBesselJ1

mathematical function

SYNOPSIS

`dip_float dipm_BesselJ1 ( x )`

FUNCTION

Computes the Bessel function J1 of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mBesselJN

mathematical function

SYNOPSIS

`dip_float dipm_BesselJN ( x, n )`

FUNCTION

Computes the Bessel function J of the order `n` of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value
dip_int	n	Order of Bessel function

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mBesselY0

mathematical function

SYNOPSIS

`dip_float dipm_BesselY0 ( x )`

FUNCTION

Computes the Bessel function Y0 of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mBesselY1

mathematical function

SYNOPSIS

`dip_float dipm_BesselY1 ( x )`

FUNCTION

Computes the Bessel function Y1 of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mBesselYN

mathematical function

SYNOPSIS

`dip_float dipm_BesselYN ( x, n )`

FUNCTION

Computes the Bessel function Y of the order `n` of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value
dip_int	n	Order of Bessel function

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

Mean

statistics function

SYNOPSIS

`dip_Error dip_Mean ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the mean of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

MeanAbsoluteError

difference measure

SYNOPSIS

`dip_Error dip_MeanAbsoluteError ( in1, in2, mask, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Calculates the mean absolute error difference between each pixel value of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	mask	Mask
dip_Image	out	Output

SEE ALSO

MeanError, MeanSquareError, RootMeanSquareError, LnNormError, IDivergence

MeanError

difference measure

SYNOPSIS

`dip_Error dip_MeanError ( in1, in2, mask, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Calculates the mean error difference between all pixel values of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	mask	Mask
dip_Image	out	Output

SEE ALSO

MeanSquareError, RootMeanSquareError, MeanAbsoluteError, LnNormError, IDivergence

MeanModulus

statistics function

SYNOPSIS

`dip_Error dip_MeanModulus ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the mean modulus of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, StandardDeviation, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

MeanSquareError

difference measure

SYNOPSIS

`dip_Error dip_MeanSquareError ( in1, in2, mask, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Calculates the mean square error difference between all pixel values of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	mask	Mask
dip_Image	out	Output

SEE ALSO

MeanError, RootMeanSquareError, MeanAbsoluteError, LnNormError, IDivergence

MeanSquareModulus

statistics function

SYNOPSIS

`dip_Error dip_MeanSquareModulus ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the mean square modulus of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`mask (0)`	Mask
`dip_Image`	`out`	Output
`dip_BooleanArray`	`ps (0)`	Dimensions to project

SEE ALSO

From images to scalars

`Sum`, `Mean`, `Variance`, `StandardDeviation`, `MeanModulus`, `SumModulus`, `Maximum`, `Minimum`, `Median`, `Percentile`

Measure

Measure object features

SYNOPSIS

`#include "dip_measurement.h"`

`dip_Error dip_Measure ( measurement, featureID, featureParams, objectID, objectIm, intensityIm, connectivity, physDims )`

DATA TYPES

`objectIm`: **integer**

`intensityIm`: integer, **float**

FUNCTION

The `Measure` function is the top-level function of DIPlib's measurement library. This function performs measurements of the objects in the specified `objectIm` image. The measurements to be performed are specified by the `featureID` array of measurement function IDs. If `featureParams` is non-zero, its size should equal that of `featureID`. Although the current implementation of `Measure` does not make use of this argument, future versions will pass the data pointers of the `featureParams` to the corresponding measurement functions. `featureParams` should be set to zero for now.

The list of object IDs on which the measurements have to be performed is specified by `objectID`. If it is zero, `Measure` will call `GetObjectLabels` to obtain a list of all non-zero values in `objectIm`. The objectID values should be unequal to zero.

The state of `measurement` should be raw (see `MeasurementNew`), since `Measure` will forge the measurement data structure by calling `MeasurementForge`.

The `intensityIm` image defines the pixel intensity of the objects, whose shape is defined by `objectIm`. If none of the measurements specified in `featureID` require the grey-value image, it can be set to `NULL`.

The `physDims` parameter defines the physical dimensions of the pixel sizes and pixel intensity. See `PhysicalDimensionsNew` for more information.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_IntegerArray	featureID	Array of measurement function IDs
dip_VoidPointerArray	featureParams (0)	Set to zero
dip_IntegerArray	objectID (0)	Array of Object IDs
dip_Image	objectIm	Image containing object IDs, i.e. object labels
dip_Image	intensityIm	Intensity image
dip_int	connectivity	Connectivity of object's contour pixels, see The connectivity parameter
dip_PhysicalDimensions	physDims	Structure specifying the physical dimensions of the image pixels

SEE ALSO

Label, ObjectToMeasurement, MeasurementToImage, MeasurementToHistogram, MeasurementWrite, MeasurementNew, MeasurementFree, MeasurementIsValid

FeatureAnisotropy2D, FeatureBendingEnergy, FeatureCenter, FeatureChainCodeBendingEnergy, FeatureConvexArea, FeatureConvexPerimeter, FeatureConvexity, FeatureDimension, FeatureExcessKurtosis, FeatureFeret, FeatureGinertia, FeatureGmu, FeatureGravity, FeatureInertia, FeatureLongestChaincodeRun, FeatureMass, FeatureMaxVal, FeatureMaximum, FeatureMean, FeatureMinVal, FeatureMinimum, FeatureMu, FeatureOrientation2D, FeatureP2A, FeaturePerimeter, FeatureRadius, FeatureShape, FeatureSize, FeatureSkewness, FeatureStdDev, FeatureSum, FeatureSurfaceArea

MeasurementFeatureConvert

Convert the data of a measurement feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureConvert ( in, featureID, inID, out, outID,
resources )
```

FUNCTION

This function convert the data of object `inID` in measurement `in` measured by feature `featureID` to object `outID` in `out`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	in	Input measurement data structure
dip_int	featureID	ID of the measurement feature
dip_int	inID	ID of the object in in
dip_Measurement	out	Output measurement data structure
dip_int	outID	ID of the object in out
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementNew

MeasurementFeatureDescription

Measurement Description access function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureDescription ( measurement, featureID,
description, resources )
```

FUNCTION

The MeasurementObjectData, MeasurementObjectValue and MeasurementFeatureDescription functions provide access to the functions that are registered by each measurement function. See also MeasurementFeatureRegister.

This function gives access to a structure containing the name, a short description of the measurement feature, as well as the labels and units of the data measured by the feature specified with `featureID`. Use the functions FeatureDescriptionGetName, FeatureDescriptionGetDescription, FeatureDescriptionGetLabels and FeatureDescriptionGetUnits to access the values in the `description` structure.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement feature ID
dip_FeatureDescription *	description	Pointer to a dip_FeatureDescription structure containing a description of the specified feature
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementNew, MeasurementFeatures, MeasurementFeatureValid, MeasurementObjectData, MeasurementObjectValue, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage FeatureDescriptionNew, FeatureDescriptionFree, FeatureDescriptionSetName, FeatureDescriptionGetName, FeatureDescriptionSetDescription, FeatureDescriptionGetDescription, FeatureDescriptionSetLabels, FeatureDescriptionGetLabels, FeatureDescriptionSetLabel, FeatureDescriptionSetDimensionLabels, FeatureDescriptionSetUnits, FeatureDescriptionGetUnits, FeatureDescriptionSetUnit

MeasurementFeatureFormat

Feature data format convenience function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureFormat ( measurement, featureID, format )
```

FUNCTION

This function is a convenience function on top of `MeasurementObjectValue`, providing an easy access to the data format of the measurement values of the `featureID` measurement function.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dipf_MeasurementValueFormat *	format	Pointer to measurement value data format

SEE ALSO

`MeasurementObjectValue`, `MeasurementNew`

MeasurementFeatureRegister

Register a measurement function

SYNOPSIS

`#include "dip_measurement.h"`

`dip_Error dip_MeasurementFeatureRegister ( registry )`

FUNCTION

This function registers a measurement function, specified by `registry`. Once a function is registered, it can be used through the Measure function by specifying `registry.id.rtid` as the measurement ID. `registry` contains pointers to a series of functions related to making the measurement, and contains information on how these functions should be called. See below for more information.

ARGUMENTS

Data type	Name	Description
dip_MeasurementFeatureRegistry	registry	Registry

THE REGISTRY STRUCTURE

The `dip_MeasurementFeatureRegistry` structure contains the following fields:

Data type	Name	Description
dip_Identifier	id	Unique identifier
dipf_FeatureMeasureFunction	type	Type of function measure points to
dip_FeatureCreateFunction	create	Function pointer, see FeatureCreateFunction
dip_FeatureComposeFunction	compose	Function pointer, see FeatureComposeFunction
dip_FeatureMeasureFunction	measure	Union of function pointers
dip_FeatureValueFunction	value	Function pointer, see FeatureValueFunction
dip_FeatureDescriptionFunction	description	Function pointer, see FeatureDescriptionFunction
dip_FeatureConvertFunction	convert	Function pointer, see FeatureConvertFunction
dip_int	iterations	Currently ignored (set to 1)
dip_Boolean	needIntensityIm	Whether or not a grey-value image is needed

dip_Identifier is a struct with two values: uuid and rtid. rtid is of type dip_int, and needs to be set to a unique number (use GetUniqueNumber for that). uuid is curently ignored, but should be set to a universally unique number by using the time, date and processor ID at the time of writing the code. The UNIX command uuidgen should be used for this.

measure points to the main measuring function, and can be of four different types, based on how it does the measuring. measure is a union with the following fields:

Data type	Name	Description
dip_FeatureLineFunction	line	Takes one image line at the time, see FeatureLineFunction
dip_FeatureImageFunction	image	Takes the whole image at once, see FeatureImageFunction
dip_FeatureChainCodeFunction	chaincode	Takes one chain code at the time, see FeatureChainCodeFunction
dip_FeatureConvHullFunction	convhull	Takes one convex polygon at the time, see FeatureConvHullFunction
dip_FeatureCompositeFunction	composite	Combines the results of various other measurements, see FeatureCompositeFunction

The type flag should match the function type pointed to, and can be one of the following:

Name	Description
DIP_MSR_FUNCTION_LINE_BASED	measure.line is set
DIP_MSR_FUNCTION_IMAGE_BASED	measure.image is set
DIP_MSR_FUNCTION_CHAINCODE_BASED	measure.chaincode is set
DIP_MSR_FUNCTION_CONVHULL_BASED	measure.convhull is set
DIP_MSR_FUNCTION_COMPOSITE	measure.composite is set

`create` points to a function that allocates and initialises any data before the measurement can start. `value` points to a function that returns the measurement result (called by `MeasurementObjectValue`). `description` points to a function that returns information on the measurement performed (called by `MeasurementFeatureDescription`). `convert` points to a function that copies the collected measurement data to a second measurement object (called `MeasurementFeatureConvert`). Finally, the `compose` element points to a function that returns the list of measurement IDs that the `DIP_MSR_FUNCTION_COMPOSITE` function depends on. This value is ignored for other types of measurement functions.

`needIntensityIm` should be set if the measurement function expects a grey-value input as well as the labeled image.

SEE ALSO

`Measure`, `MeasurementFeatureRegistryList`, `MeasurementFeatureRegistryGet`, `MeasurementFeatureRegistryFeatureDescription`, `MeasurementFeatureRegistryFeatureNeedsIntensityImage`, `MeasurementFeatureValid`, `MeasurementFeatureDescription`, `MeasurementObjectData`, `MeasurementObjectValue`, `MeasurementFeatureConvert`, `FeatureLineFunction`, `FeatureImageFunction`, `FeatureChainCodeFunction`, `FeatureConvHullFunction`, `FeatureCompositeFunction`, `FeatureCreateFunction`, `FeatureComposeFunction`, `FeatureValueFunction`, `FeatureConvertFunction`, `FeatureDescriptionFunction`

MeasurementFeatureRegistryFeatureDescription

Get the feature description of a registered measurement feature

SYNOPSIS

```
#include "dip_measurement.h"
```

```
dip_Error dip_MeasurementFeatureRegistryFeatureDescription ( featureID,
description, resources )
```

FUNCTION

This function obtains the feature description information of the measurement feature specified by `featureID`.

ARGUMENTS

Data type	Name	Description
`dip_int`	`featureID`	Measurement feature ID
`dip_FeatureDescription *`	`description`	pointer to a dip_FeatureDescription structure containing descriptive information of the measurement feature. This data can be accessed with `MeasurementFeatureDescription`
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`Measure`, `MeasurementFeatureRegister`, `MeasurementFeatureRegistryList`, `MeasurementFeatureRegistryGet`, `MeasurementFeatureRegistryFeatureNeedsIntensityImage`, `MeasurementFeatureValid`, `MeasurementFeatureDescription`, `MeasurementObjectData`, `MeasurementObjectValue`

MeasurementFeatureRegistryFeatureNeedsIntensityImage

Checks whether the measurement function needs an intensity image

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureRegistryFeatureNeedsIntensityImage (
featureID, veredict )
```

FUNCTION

This function sets `veredict` to `DIP_TRUE` if the measurement feature specified by `featureID` requires a grey-value image, or `DIP_FALSE` otherwise.

ARGUMENTS

Data type	Name	Description
dip_int	featureID	Measurement feature ID
dip_Boolean *	veredict	Return value

SEE ALSO

Measure, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureValid, MeasurementFeatureDescription, MeasurementObjectData, MeasurementObjectValue

MeasurementFeatureRegistryGet

Get the registry information of a measurement feature

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureRegistryGet ( featureID, registry )
```

FUNCTION

This function obtains (a copy of) the registry structure of the measurement feature function specified by `featureID`.

ARGUMENTS

Data type	Name	Description
dip_int	featureID	Measurement function ID
dip_MeasurementFeatureRegistry *	registry	Pointer to a measurement feature registry structure

SEE ALSO

Measure, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage, MeasurementFeatureValid, MeasurementFeatureDescription, MeasurementObjectData, MeasurementObjectValue

MeasurementFeatureRegistryList

Obtain a list of the registered measurement features

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureRegistryList ( featureID, resources )
```

FUNCTION

This functions obtains an array of registered measurement feature IDs.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	featureID	Pointer to an array of measurement feature IDs
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementFeatureRegister, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage, MeasurementFeatureValid, MeasurementFeatureDescription, MeasurementObjectData, MeasurementObjectValue

MeasurementFeatures

Get the measurement ID array

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatures ( measurement, featureID, resources )
```

FUNCTION

This function obtains an array of measurement function IDs in the measurement structure. See MeasurementForge for a (brief) explination of the measurement data structure.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_IntegerArray *	featureID	pointer to an array of measurement function IDs
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementNew, MeasurementNumberOfFeatures, MeasurementFeatureValid, MeasurementFeatureDescription

MeasurementFeatureSize

Feature data convenience function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureSize ( measurement, featureID, size )
```

FUNCTION

This function is a convenience function on top of `MeasurementObjectValue`, providing an easy access to the number of the measurement values of the `featureID` measurement function.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	ID of the measurement feature
dip_int *	size	Number of measurement values

SEE ALSO

`MeasurementObjectValue`, `MeasurementNew`

MeasurementFeatureValid

Verify a measurement feature ID

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFeatureValid ( measurement, featureID, verdict )
```

FUNCTION

This function determines whether `featureID` is a valid measurement feature, by verifying whether `featureID` equals the ID of one of the registered measurement features. If `verdict` is not zero, the result (`DIP_TRUE` or `DIP_FALSE`) is stored in `verdict`, otherwise an error is returned in case the verification fails.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_int`	`featureID`	Measurement feature ID to validated
`dip_Boolean *`	`verdict`	Pointer to the boolean verdict

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementFeatures`, `MeasurementFeatureDescription`, `MeasurementFeatureRegister`, `MeasurementFeatureRegistryList`, `MeasurementFeatureRegistryGet`, `MeasurementFeatureRegistryFeatureDescription`, `MeasurementFeatureRegistryFeatureNeedsIntensityImage`

MeasurementForge

Allocate the data of a measurement data structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementForge ( measurement, featureID, objectID )
```

FUNCTION

This function forges a measurement data structure, that has been created with MeasurementNew. The featureID array should contain the IDs of the features to be performed. The vadility of these IDs is checked by comparing them with the IDs of registered measurement functions (see MeasurementFeatureRegister). The objectID array contains the IDs (i.e. labels) of the objects on which the features are to be performed. (For example, the Measure function accepts as one of its arguments a label image, of which the intensity of each individual pixel represents the ID of the object to which that pixel belongs. These label values should in that case correspond to the values of objectID.)

The measurement structure can be regarded as a matrix spanned by the number of features along one axis, and the number of objects along the other. MeasurementForge allocates and initialises the internal structures to contain this matrix and the data required for each conbimation of measurement and object ID.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement structure
dip_IntegerArray	featureID	Array of measurement function IDs
dip_IntegerArray	objectID	Array of Object IDs

SEE ALSO

Measure, MeasurementNew, MeasurementFree, MeasurementIsValid, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage

MeasurementFree

Free a measurement data structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementFree ( measurement )
```

FUNCTION

This function frees a Measurement data structure. After the Measurement has been freed, the pointer `measurement` is set to zero.

ARGUMENTS

Data type	Name	Description
dip_Measurement *	measurement	pointer to the measurement structure to be freed

SEE ALSO

Measure, MeasurementNew, MeasurementForge, MeasurementIsValid

MeasurementGetName

Get the name of a Measurement structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementGetName ( measurement, name, resources )
```

FUNCTION

This function gets the name of a measurement structure

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement
dip_String *	name	Name
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementNew, MeasurementID, MeasurementSetName, MeasurementGetPhysicalDimensions, MeasurementSetPhysicalDimensions

MeasurementGetPhysicalDimensions

Get the physical dimensions info of a measurement

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementGetPhysicalDimensions ( measurement, physDims,
resources )
```

FUNCTION

This function obtains a copy of the physical dimensions information associated with the `measurement` data structure. The physical dimensions data structure informs measurement features about the physical sizes and position of the pixels of the measured image.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_PhysicalDimensions *`	`physDims`	Pointer to a Physical Dimensions data structure
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementID`, `MeasurementSetName`, `MeasurementGetName`, `MeasurementSetPhysicalDimensions`

MeasurementID

Get the ID of a Measurement structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementID ( measurement, id )
```

FUNCTION

This function obtains the ID of a the `measurement` structure. The ID is a DIPlib wide unique number (see `GetUniqueNumber`).

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement structure
dip_int *	id	Pointer to the id

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementSetName`, `MeasurementGetName`, `MeasurementGetPhysicalDimensions`, `MeasurementSetPhysicalDimensions`

MeasurementIsValid

Checks whether a measurement is valid

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementIsValid ( measurement, verdict )
```

FUNCTION

This function determines whether `measurement` is forged. If `verdict` is not zero, the result (`DIP_TRUE` or `DIP_FALSE`) is stored in `verdict`, otherwise an error is returned in case the verification fails.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_Boolean *	verdict	The validation verdict

SEE ALSO

Measure, MeasurementNew, MeasurementFree, MeasurementForge

MeasurementNew

Create new measurement data structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementNew ( measurement, resources )
```

FUNCTION

This function creates, by allocating and initialising it, a new Measurement data structure. After this function has been used to create a new measurement structure, the state of it is raw. It needs to be passed through MeasurementForge before it can be used to store measurement results.

ARGUMENTS

Data type	Name	Description
dip_Measurement *	measurement	pointer to the measurement structure to be created
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Measure, MeasurementFree, MeasurementForge, MeasurementIsValid, MeasurementID, MeasurementSetName, MeasurementGetName, MeasurementGetPhysicalDimensions, MeasurementSetPhysicalDimensions, MeasurementNumberOfFeatures, MeasurementFeatures, MeasurementFeatureValid, MeasurementFeatureDescription, MeasurementNumberOfObjects, MeasurementObjects, MeasurementObjectValid, MeasurementObjectData, MeasurementObjectValue, MeasurementFeatureConvert, MeasurementWrite

MeasurementNumberOfFeatures

Get the number of measurement feature IDs

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementNumberOfFeatures ( measurement, features )
```

FUNCTION

This function obtains the number of measurement feature IDs in the measurement structure. See MeasurementForge for a (brief) explination of the measurement data structure.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int *	features	pointer to the number of measurement feature IDs

SEE ALSO

Measure, MeasurementNew, MeasurementFeatures

MeasurementNumberOfObjects

Get the number of object IDs

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementNumberOfObjects ( measurement, objects )
```

FUNCTION

This function obtains the number of object IDs belonging to the `featureID` measurement function ID in the measurement structure. See `MeasurementForge` for a (brief) explination of the measurement data structure.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_int *`	`objects`	Pointer to an integer containing the number of object IDs

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementObjects`

MeasurementObjectData

Object data access function

SYNOPSIS

```
#include "dip_measurement.h"
```

```
dip_Error dip_MeasurementObjectData ( measurement, featureID, objectID, data,
verdict )
```

FUNCTION

The MeasurementObjectData, MeasurementObjectValue and MeasurementFeatureDescription functions provide access to the functions that are registered by each measurement function. See also MeasurementFeatureRegister.

The Object data is the data allocated by a measurement function for internal purposes, for example to store intermediate results. Its format is free. Therefore, the use of this function is only meaningful for a particular measurement function itself. To access the measurement values of a measurement function, use MeasurementObjectValue.

The `verdict` parameter provides a means to test whether `featureID` or `objectID` are valid within the context of `measurement`. If one of them is invalid, and `verdict` is not zero, `*verdict` is set to `DIP_FALSE`, otherwise its value is `DIP_TRUE`. If `verdict` is zero, `MeasurementObjectData` produces an error when either `featureID` or `objectID` is invalid.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	Object ID
void **	data	Pointer to the internal measurement data pointer
dip_Boolean *	verdict	Pointer to a boolean containing validation information

SEE ALSO

Measure, MeasurementNew, MeasurementObjects, MeasurementObjectData, MeasurementObjectValue, MeasurementFeatureDescription, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage

MeasurementObjects

Get an object ID array

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementObjects ( measurement, featureID, objectID,
resources )
```

FUNCTION

This function obtains an array of object IDs belonging to the `featureID` measurement function in the measurement structure. See `MeasurementForge` for a (brief) explination of the measurement data structure.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_int`	`featureID`	Measurement function ID
`dip_IntegerArray *`	`objectID`	Pointer to an object ID array
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementNumberOfObjects`, `MeasurementObjectValid`, `MeasurementObjectData`, `MeasurementObjectValue`

MeasurementObjectValid

Verify an object ID

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementObjectValid ( measurement, featureID, objectID,
verdict )
```

FUNCTION

This function determines whether the object ID `objectID`, belonging to the measurement function ID `featureID`, is a valid object ID, by comparing `objectID` to the object IDs belonging to the `featureID` in `measurement`. If `verdict` is not zero, the result (`DIP_TRUE` or `DIP_FALSE`) is stored in `verdict`, otherwise an error is returned in case the verification fails.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_int`	`featureID`	Measurement function ID
`dip_int`	`objectID`	Object ID
`dip_Boolean *`	`verdict`	Pointer to a boolean contaning the validation verdict

SEE ALSO

`Measure`, `MeasurementNew`, `MeasurementObjects`, `MeasurementObjectData`, `MeasurementObjectValue`

MeasurementObjectValue

Object value access function

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementObjectValue ( measurement, featureID, objectID,
data, format, resources )
```

FUNCTION

The MeasurementObjectData, MeasurementObjectValue and MeasurementFeatureDescription functions provide access to the functions that are registered by each measurement function. See also MeasurementFeatureRegister.

The MeasurementObjectValue function provides access to the measurement values produced by the featureID measurement function measured on the objectID labeled object. The format of data is specified by format.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_int	featureID	Measurement function ID
dip_int	objectID	Object ID
void **	data	Pointer to data pointer
dipf_MeasurementValueFormat *	format	Pointer to the data format label
dip_Resources	resources	Resources tracking structure. See ResourcesNew

Measurement data formats

Name	Description
DIP_MSR_VALUE_FORMAT_INTEGER	Integer scalar data format
DIP_MSR_VALUE_FORMAT_FLOAT	Float scalar data format
DIP_MSR_VALUE_FORMAT_INTEGER_ARRAY	Integer array data format
DIP_MSR_VALUE_FORMAT_FLOAT_ARRAY	Float array data format
DIP_MSR_VALUE_FORMAT_IMAGE	Data is formatted as an dip_Image

SEE ALSO

Measure, MeasurementNew, MeasurementObjects, MeasurementObjectData, MeasurementObjectValue, MeasurementFeatureDescription, MeasurementFeatureFormat, MeasurementFeatureSize, MeasurementFeatureRegister, MeasurementFeatureRegistryList, MeasurementFeatureRegistryGet, MeasurementFeatureRegistryFeatureDescription, MeasurementFeatureRegistryFeatureNeedsIntensityImage

MeasurementRead

Read measurement results from a file

SYNOPSIS

```
#include "dipio_measurement.h"
```

```
dip_Error dipio_MeasurementRead ( measurement, filename, format,
addExtensions, recognised )
```

FUNCTION

This function reads measurement data from a file and puts it in `measurement`. `measurement` must be allocated before calling this function. If `format` is 0, all different `MeasurementRead` functions are called in sequence until the correct format has been found. If you know the format, get the correct format ID through the registry functions.

The boolean `addExtensions` specifies whether `MeasurementRead` should try to add file format extensions to `filename`, if the registered file format reader fails to recognise `filename` straight away. The extensions are provided by the registered file readers.

If `recognised` is not zero, `MeasurementRead` will set it to `DIP_TRUE` when it has been able to read `filename`, and it will set it to `DIP_FALSE` when it is not able to read the file. No error will be generated in this case.

NOTE

There are currently no measurement reading functions, so this function will always fail.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`dip_String`	`filename`	File name to read from
`dip_int`	`format`	ID of file format
`dip_Boolean`	`addExtensions`	Add extensions when looking for the file
`dip_Boolean *`	`recognised`	Set to `DIP_TRUE` if the file was found

SEE ALSO

`Measure`, `MeasurementWrite`

MeasurementSetName

Set the name of a measurement structure

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementSetName ( measurement, name )
```

FUNCTION

This function sets the name of `measurement` to `name`.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement
dip_String	name	Name

SEE ALSO

Measure, MeasurementNew, MeasurementID, MeasurementGetName, MeasurementGetPhysicalDimensions, MeasurementSetPhysicalDimensions

MeasurementSetPhysicalDimensions

Set the physical dimensions info of the measurement

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementSetPhysicalDimensions ( measurement, physDims )
```

FUNCTION

This function sets the physical dimensions information for the `measurement` data structure. The physical dimensions data structure informs measurement features about the physical sizes and position of the pixels of the measured image.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_PhysicalDimensions	physDims	Physical Dimensions data structure

SEE ALSO

Measure, MeasurementNew, MeasurementID, MeasurementSetName, MeasurementGetName, MeasurementGetPhysicalDimensions

MeasurementToHistogram

Creats a histogram for a measurement

SYNOPSIS

`#include "dip_measurement.h" #include "dip_distribution.h"`

`dip_Error dip_MeasurementToHistogram ( histogram, measurement, featureID, binSize, maximum, minimum, percentage, addMeasurement )`

DATA TYPES

integer

FUNCTION

This function creates a (possibly multi-dimensional) histogram with the measurement results of one feature. If `addMeasurement` is `DIP_TRUE`, new data points are added to the existing histogram, and `binSize`, `maximum`, `minimum` and `percentage` input arguments are ignored. Otherwise, `histogram` is destroyed and recreated according to the chosen values for `binSize`, `maximum`, `minimum` and `percentage`. If `percentage` is `DIP_TRUE`, `maximum` and `minimum` represent a percentage of the data range, otherwise they represent absolute values. If `maximum` or `minimum` are `NULL`, the maximum or minimum of the data is used.

ARGUMENTS

Data type	Name	Description
dip_Distribution	histogram	Output histogram
dip_Measurement	measurement	Measurement data
dip_IntegerArray	featureID	List of feature IDs to use
dip_FloatArray	binSize	Size of the histogram bins
dip_FloatArray	maximum	Maximum value represented in the histogram
dip_FloatArray	minimum	Minimum value represented in the histogram
dip_Boolean	percentage	Whether `maximum` and `minimum` are percentages
dip_Boolean	addMeasurement	Whether to add data to `histogram` or create a new one

SEE ALSO

Measure, MeasurementToImage, ObjectToMeasurement

MeasurementToImage

Exports the data in a measurement structure to an image

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_MeasurementToImage ( measurement, out, measurementIDs, objects
)
```

DATA TYPES

float

FUNCTION

This function creates an image and writes the measurement data to it as if it were a table, measurements along the first dimension, objects along the second dimension.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Input measurement
dip_Image	out	Output image
dip_IntegerArray	measurementIDs	List of measurement IDs to export
dip_IntegerArray	objects	List of object IDs to export

SEE ALSO

Measure, ObjectToMeasurement, MeasurementToHistogram

MeasurementWrite

Write measurement results to a file

SYNOPSIS

```
#include "dipio_measurement.h"
dip_Error dipio_MeasurementWrite ( measurement, filename, format, labels )
```

FUNCTION

This function writes measurement data to a file, overwriting any other file with the same name. Get the format ID through the registry functions. If `format` is 0, CSV is used.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_String	filename	File name to write to
dip_int	format	ID of file format
dip_Boolean	labels	DIP_TRUE to write labels to file

SEE ALSO

Measure, MeasurementRead, MeasurementWriteCSV, MeasurementWriteHTML, MeasurementWriteText

MeasurementWriteCSV

Write measurement results to a CSV file

SYNOPSIS

```
#include "dipio_measurement.h"
dip_Error dipio_MeasurementWriteCSV (measurement, filename, separator,
labels)
```

FUNCTION

This function writes the measurement results to a comma separated values (CSV) file, overwriting any other file with the same name.

This function calls MeasurementWriteText with the proper settings.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_String	filename	File name to write to
char *	separator	Characters to separate values
dip_Boolean	labels	DIP_TRUE to write labels to file

SEE ALSO

Measure, MeasurementWrite, MeasurementWriteText

MeasurementWriteHTML

Write measurement results to an HTML file

SYNOPSIS

```
#include "dipio_measurement.h"
dip_Error dipio_MeasurementWriteHTML (measurement, filename, separator,
labels)
```

FUNCTION

This function writes the measurement results to a formatted HTML file, overwriting any other file with the same name.

ARGUMENTS

Data type	Name	Description
dip_Measurement	measurement	Measurement data structure
dip_String	filename	File name to write to
char *	separator	Characters to separate values
dip_Boolean	labels	DIP_TRUE to write labels to file

SEE ALSO

Measure, MeasurementWrite

MeasurementWriteText

Write measurement results as readable text

SYNOPSIS

```
#include "dipio_measurement.h"
dip_Error dipio_MeasurementWriteText ( measurement, fp, options )
```

FUNCTION

This function saves/prints the results of a measurement stored in the `measurement` data structure. Since it will save the results to the `fp` FILE pointer (which has to be opened before this function is called, and closed afterwards), the results can be printed to a screen (specify stdout as `fp`) or to a file.

The results are saved in a matrix, with a column for each measurement, and a row for each object. The first column contains the object ID. The `options` structure provides a means to adjust the formatting of the measurement data. Its `separator` variable specifies the column separator character, the rows are separated by a newline. If the `labelAlign` variable is `DIP_TRUE`, the `separator` is repeated such that the columns are aligned. If the `labels` variable is `DIP_TRUE`, the first row contains measurement labels, and `info` specifies whether or not the short description of each measurement function should be printed before the result matrix. If `results` is `DIP_FALSE`, the measurement values are not printed.

ARGUMENTS

Data type	Name	Description
`dip_Measurement`	`measurement`	Measurement data structure
`FILE *`	`fp`	FILE pointer to which the results are saved
`dipio_WriteTextFormat`	`options`	Text formatting options

The structure `dipio_WriteTextFormat` contains the following elements:

Data type	Name	Description
`char *`	`separator`	Column separator character
`dip_Boolean`	`info`	Write descriptio
`dip_Boolean`	`labels`	Write labels
`dip_Boolean`	`results`	Write values
`dip_Boolean`	`labelAlign`	Align columns

SEE ALSO

`Measure`, `MeasurementWrite`

Median

statistics function

SYNOPSIS

`dip_Error dip_Median ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the median of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`mask (0)`	Mask
`dip_Image`	`out`	Output
`dip_BooleanArray`	`ps (0)`	Dimensions to project

SEE ALSO

From images to scalars

`Sum`, `Mean`, `Variance`, `StandardDeviation`, `MeanModulus`, `SumModulus`, `MeanSquareModulus`, `Maximum`, `Minimum`, `Percentile`

MedianFilter

Non-linear smoothing filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MedianFilter ( in, out, se, boundary, param, shape )
```

DATA TYPES

integer, **float**

FUNCTION

Median filter with different filter shapes.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_Image`	`se`	Custom filter shape (binary)
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_FloatArray`	`param`	Filter parameters
`dip_FilterShape`	`shape`	Filter shape

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

PercentileFilter, Uniform, Sigma

MemoryCopy

Copy memory blocks

SYNOPSIS

`void dip_MemoryCopy( in, out, number )`

FUNCTION

Copy a memory block

ARGUMENTS

Data type	Name	Description
`void *`	`in`	pointer to memory source block
`void *`	`out`	pointer to memory destination block
`dip_int`	`number`	number of bytes to be copied

NOTE

The behaviour of this function is undefined when the `in` and `out` blocks overlap.

MemoryFree

Free a chunk of memory

SYNOPSIS

`dip_Error dip_MemoryFree( pointer )`

FUNCTION

Frees a chunk of memory.

ARGUMENTS

Data type	Name	Description
void *	pointer	pointer to an allocated chunk of memory

SEE ALSO

MemoryNew, MemoryReallocate, MemoryFunctionsSet

MemoryFunctionsSet

Sets memory allocation functions

SYNOPSIS

`dip_Error dip_MemoryFunctionsSet( MemoryNewFunction, MemoryReallocateFunction, MemoryFreeFunction )`

FUNCTION

Sets the memory allocation functions used by DIPlib.

ARGUMENTS

Data type	Name	Description
dip_MemoryNewFunction	MemoryNewFunction	pointer to a memory allocation function
dip_MemoryReallocateFunction	MemoryReallocateFunction	pointer to a memory reallocation function
dip_MemoryFreeFunction	MemoryFreeFunction	pointer to a memory freeing function

NOTE

The three allocation functions are defined as follows:

```
typedef void* (*dip_MemoryNewFunction)(size_t size)
typedef void* (*dip_MemoryReallocateFunction)(void *ptr, size_t size)
typedef void (*dip_MemoryFreeFunction)(void *ptr)
```

And are by default set to `malloc`, `realloc` and `free`.

SEE ALSO

MemoryNew, MemoryReallocate, MemoryFree

MemoryNew

Allocate and track memory

SYNOPSIS

`dip_Error dip_MemoryNew( pointer, size, resources )`

FUNCTION

Allocates a chunk of memory, and adds a reference to the chunk to the list of tracked resources.

ARGUMENTS

Data type	Name	Description
void **	pointer	pointer to the memory chunk pointer
size_t	size	size of the memory chunk in bytes
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MemoryReallocate, MemoryFree, MemoryFunctionsSet

MemoryReallocate

Reallocate a chunk of memory

SYNOPSIS

`dip_Error dip_MemoryReallocate ( pointer, newsize, resources )`

FUNCTION

Reallocates a chunk of memory, to change its size. `resources` must be the `dip_Resources` structure used in the call to `MemoryNew` when `pointer` was allocated.

ARGUMENTS

Data type	Name	Description
void **	pointer	pointer to the memory chunk pointer
size_t	newsize	size of the memory chunk in bytes
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

MemoryNew, MemoryFree, MemoryFunctionsSet

mErf

mathematical function

SYNOPSIS

`dip_float dipm_Erf ( x )`

FUNCTION

Computes the error function of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErfc, mGammaP, mGammaQ

mErfc

mathematical function

SYNOPSIS

`dip_float dipm_Erfc ( x )`

FUNCTION

Computes the complementary error function of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mGammaP, mGammaQ

mExp10

mathematical function

SYNOPSIS

`dip_float dipm_Exp10 ( x )`

FUNCTION

Computes the base ten exponent of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	value	Value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mExp2

mathematical function

SYNOPSIS

`dip_float dipm_Exp2 ( x )`

FUNCTION

Computes the base two exponent of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	value	Value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mFraction

mathematical function

SYNOPSIS

`dip_float dipm_Fraction ( x )`

FUNCTION

Computes the fraction of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mGammaP

mathematical function

SYNOPSIS

`dip_float dipm_GammaP ( a, x )`

FUNCTION

Computes the incomplete gamma function of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	a	A
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaQ

mGammaQ

mathematical function

SYNOPSIS

`dip_float dipm_GammaQ ( a, x )`

FUNCTION

Computes the complementary incomplete gamma function of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	a	A
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP

Min

arithmetic function

SYNOPSIS

`dip_Error dip_Min ( in1, in2, out )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = min(`in1` , `in2`) on a pixel by pixel basis. The data types of the `in1` and `in2` image may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

Max, MaxFloat, MinFloat

MinFloat

arithmetic function

SYNOPSIS

`dip_Error dip_MinFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function computes `out` = min(`in` , `constant`) on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

Max, MaxFloat, Min

Minima

Detects local minima

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_Minima ( in, mask, out, connectivity, booleanOutput )
```

DATA TYPES

integer, float

FUNCTION

This function detects local minima.

The algorithm finds a connected set of pixels with identical value, an no neighbours with lower value. This set is a local minimum and its pixels are set to 1 in the output image. If `booleanOutput` is false, the output image is a labelled image.

This function differs from `LocalMinima` in that it marks every minimum. `LocalMinima` is able to filter out unimportant minima.

For images that have large plateaus (regions of constant value) that are not local minima, this function can be quite slow. For example, an image that is one everywhere except for a small valley somewhere. For such an image it is recommended to use the `mask` input, for example with the output of a threshold operation.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	mask	Mask image
dip_Image	out	Binary output image
dip_int	connectivity	Connectivity
dip_Boolean	booleanOutput	Give a binary output image?

NOTE

If you are looking for the old version of `Minima`, it is still available through the following combination of commands:

```
dip_Erosion( in, out, se, boundary, param, shape );
dip_Equal( in, out, out );
```

SEE ALSO

Maxima, SubpixelMinima, LocalMinima, SeededWatershed, GrowRegions

Minimum

statistics function

SYNOPSIS

`dip_Error dip_Minimum ( in, mask, out, ps )`

DATA TYPES

binary, integer, float

FUNCTION

Calculates the minimum of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Median, Percentile

Mirror

Mirrors an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Mirror ( in, out, mirror )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

This function mirrors the pixels in those dimensions of image as specified by `mirror`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_BooleanArray	mirror	Mirror flags

SEE ALSO

Map

mLnGamma

mathematical function

SYNOPSIS

`dip_float dipm_LnGamma ( x )`

FUNCTION

Computes the natural logarithm of the gamma function of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	value	Value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mErf, mErfc, mGammaP, mGammaQ

mLog2

mathematical function

SYNOPSIS

`dip_float dipm_Log2 ( x )`

FUNCTION

Computes the base two logarithm of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	value	Value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mNearestInt

mathematical function

SYNOPSIS

`dip_float dipm_NearestInt ( x )`

FUNCTION

Computes the nearest int of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

Modulo

Arithmetic function

SYNOPSIS

`dip_Error dip_Modulo ( in, out, period )`

DATA TYPES

integer

FUNCTION

Computes the modulo of the input image values, by computing the remainder of the the division of the input image values with period.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	period	Period

SEE ALSO

Div, DivFloat, DivComplex, Reciprocal

Modulus

Arithmetic function

SYNOPSIS

`dip_Error dip_Modulus ( in, out )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

Computes the modulus of the input image values, and outputs a float typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Phase, Real, Imaginary

MonadicFrameWork

FrameWork for monadic operations

SYNOPSIS

`dip_Error dip_MonadicFrameWork ( in, out, processBoundary, processBorder, process )`

FUNCTION

This function is a frontend on the SeparableFrameWork. It provides an easier interface for filters that only need to scan an image once. The dimension in which the image should be scanned can be specified or left to `MonadicFrameWork` by specifiying the dimension with `DIP_MONADIC_OPTIMAL_DIMENSION`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Boundary	processBoundary	ProcessBoundary
dip_int	processBorder	ProcessBorder
dip_FrameWorkProcess	process	Process

SEE ALSO

SeparableFrameWork, SingleOutputFrameWork

MorphologicalGradientMagnitude

Morphological edge detector

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MorphologicalGradientMagnitude ( in, out, se, boundary, param,
shape, edgeType )
```

DATA TYPES

integer, **float**

FUNCTION

The same as MorphologicalRange.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dip_MphEdgeType	edgeType	edgeType

SEE ALSO

MorphologicalRange, Lee, MultiScaleMorphologicalGradient, Tophat

MorphologicalRange

Morphological edge detector

SYNOPSIS

#include "dip_morphology.h"

dip_Error dip_MorphologicalRange (in, out, se, boundary, param, shape, edgeType)

DATA TYPES

integer, **float**

FUNCTION

Implements a morphological edge detector based on the difference of two complementary morphological operations. These can be chosen through the `edgeType` parameter.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dip_MphEdgeType	edgeType	edgeType

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

The enumerator `dip_MphEdgeType` contains the following constants:

Name	Description
DIP_MPH_TEXTURE	Response is limited to edges in texture
DIP_MPH_OBJECT	Response is limited to object edges
DIP_MPH_BOTH	All edges produce equal response

SEE ALSO

`MorphologicalGradientMagnitude`, `Lee`, `MultiScaleMorphologicalGradient`, `Tophat`

MorphologicalReconstruction

Morphological filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MorphologicalReconstruction ( marker, mask, out, connectivity )
```

DATA TYPES

integer, **float**

FUNCTION

Dilation of the image `marker`, constrained by the image `mask`. `out` will be smaller or equal to `mask`. The image is grown according to the `connectivity` parameter. See The connectivity parameter for more information.

ARGUMENTS

Data type	Name	Description
dip_Image	marker	Marker input image
dip_Image	mask	Mask input image
dip_Image	out	Output image
dip_int	connectivity	Connectivity

KNOWN LIMITATIONS

The image `marker` is converted to the same data type as `mask`. If `mask` is an unsigned integer, and `marker` has negative values, then it is possible that these negative values will be turned into large positive values, yielding an unexpected output. The solution is to make sure that `mask` and `marker` are in compatible data types.

LITERATURE

K. Robinson and P.F. Whelan, Efficient morphological reconstruction: a downhill filter, Pattern Recognition Letters 25(15):1759-1767, 2004.

SEE ALSO

`Dilation`, `BinaryPropagation`, `AreaOpening`

MorphologicalSmoothing

Morphological smoothing filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MorphologicalSmoothing ( in, out, se, boundary, param, shape, flags )
```

DATA TYPES

integer, **float**

FUNCTION

Implements a morphological smoothing based on the sequence of two complementary morphological operations. These can be chosen through the `dipf_MphSmoothing` parameter.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dipf_MphSmoothing	flags	flags

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

The enumerator `dipf_MphSmoothing` contains the following constants:

Name	Description
DIP_MPH_OPEN_CLOSE	First the opening, then the closing
DIP_MPH_CLOSE_OPEN	First the closing, then the opening
DIP_MPH_AVERAGE	The average of the result of the two above

SEE ALSO

MorphologicalThreshold, Tophat

MorphologicalThreshold

Morphological smoothing filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MorphologicalThreshold ( in, out, se, boundary, param, shape,
edgeType )
```

DATA TYPES

integer, **float**

FUNCTION

Implements a morphological smoothing based on the average of two complementary morphological operations. These can be chosen through the `edgeType` parameter.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dip_MphEdgeType	edgeType	edgeType

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

The enumerator `dip_MphEdgeType` contains the following constants:

Name	Description
DIP_MPH_TEXTURE	Response is limited to edges in texture
DIP_MPH_OBJECT	Response is limited to object edges
DIP_MPH_BOTH	All edges produce equal response

SEE ALSO

`MorphologicalSmoothing`, `Tophat`

mReciprocal

mathematical function

SYNOPSIS

`dip_float dipm_Reciprocal ( x )`

FUNCTION

Computes the reciprocal (1/x) of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mSign

mathematical function

SYNOPSIS

`dip_float dipm_Sign ( x )`

FUNCTION

Computes the sign of the input value. The sign of zero is defined as zero.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mSinc

mathematical function

SYNOPSIS

`dip_float dipm_Sinc ( x )`

FUNCTION

Computes the sinc (sin(x)/x) of the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mTruncate, mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

mTruncate

mathematical function

SYNOPSIS

`dip_float dipm_Truncate ( x )`

FUNCTION

Truncates the input value.

ARGUMENTS

Data type	Name	Description
dip_float	x	Input value

SEE ALSO

mFraction, mNearestInt, mSign, mExp2, mExp10, mLog2, mSinc, mReciprocal, mBesselJ0, mBesselJ1, mBesselJN, mBesselY0, mBesselY1, mBesselYN, mLnGamma, mErf, mErfc, mGammaP, mGammaQ

Mul

arithmetic function

SYNOPSIS

`dip_Error dip_Mul ( in1, in2, out )`

Calls `Arith` `( in1, in2, out, DIP_ARITHOP_MUL, DIP_DT_MINIMUM )`

MulComplex

arithmetic function

SYNOPSIS

`dip_Error dip_MulComplex ( in, out, constant )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

This function computes `out` = `in` * `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_complex	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, MulInteger, MulFloat, AddComplex, SubComplex, MulConjugateComplex, DivComplex

MulConjugate

arithmetic function

SYNOPSIS

`dip_Error dip_MulConjugate ( in1, in2, out )`

Calls `Arith ( in1, in2, out, DIP_ARITHOP_MUL_CONJUGATE, DIP_DT_MINIMUM )`

MulConjugateComplex

arithmetic function

SYNOPSIS

`dip_Error dip_MulConjugateComplex ( in, out, constant )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

This function computes `out` = `in` * conj(`constant`) on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_complex	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, AddComplex, SubComplex, MulComplex, DivComplex

MulFloat

arithmetic function

SYNOPSIS

`dip_Error dip_MulFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` * `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, MulInteger, MulComplex, AddFloat, SubFloat, DivFloat

MulInteger

arithmetic function

SYNOPSIS

`dip_Error dip_MulInteger ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in` * `constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, MulFloat, MulComplex, AddInteger, SubInteger, DivInteger

MultiScaleMorphologicalGradient

Morphological edge detector

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_MultiScaleMorphologicalGradient ( in, out, se, boundary,
upperSize, lowerSize, shape )
```

DATA TYPES

integer, **float**

FUNCTION

This function computes the average morphological gradient over a range of scales bounded by `upperSize` and `lowerSize`. The morphological gradient is computed as the difference of the dilation and erosion of the input image at a particular scale, eroded by an erosion of one size smaller. At the lowest scale, the size of the structuring element is `2 * upperSize + 1`.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_int	upperSize	Upper size of structuring element
dip_int	lowerSize	Lower size of structuring element
dip_FilterShape	shape	Structuring element

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

LITERATURE

D. Wang, Pattern Recognition, **30**(12), pp. 2043-2052, 1997

SEE ALSO

`Lee`, `MorphologicalGradientMagnitude`, `MorphologicalRange`, `Tophat`

NearestInt

Arithmetic function

SYNOPSIS

`dip_Error dip_NearestInt ( in, out )`

DATA TYPES

binary, integer, float, **complex** binary, integer, **float**

FUNCTION

Computes the nearest integer value of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Ceil, Floor, Sign, Truncate, Fraction

NeighbourIndicesListMake

Get indices to direct neighbours

SYNOPSIS

```
#include "dip_neighbourlist.h"
```

```
dip_Error dip_NeighbourIndicesListMake ( stride, connectivity, indices,
resources )
```

FUNCTION

A list `indices` is created with the relative indices to the direct neighbours of a pixel in an image whose strides are given by `stride`. How many direct neighbours are returned is controlled by `connectivity`, see The connectivity parameter for the available values and their meaning.

indices is allocated and tracked in `resources`.

ARGUMENTS

Data type	Name	Description
`dip_IntegerArray`	`stride`	Stride array
`dip_int`	`connectivity`	Connectivity
`dip_IntegerArray *`	`indices`	Output neighbour indices
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`NeighbourListMake`, `NeighbourListMakeChamfer`, `NeighbourListMakeImage`, `NeighbourListToIndices`, `NeighbourIndicesListMake`

NeighbourListMake

Get list of direct neighbours

SYNOPSIS

`#include "dip_neighbourlist.h"`

`dip_Error dip_NeighbourListMake ( ndims, connectivity, coords, resources )`

FUNCTION

A list `coords` is created with the relative coordinates to the direct neighbours of a pixel in an `ndims`-dimensional image. How many direct neighbours are returned is controlled by `connectivity`, see The connectivity parameter for the available values and their meaning.

`coords` is allocated and tracked in `resources`.

ARGUMENTS

Data type	Name	Description
dip_int	ndims	Image dimensionality
dip_int	connectivity	Connectivity
dip_CoordinateArray *	coords	Output neighbour coordinates
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

NeighbourListMake, NeighbourListMakeChamfer, NeighbourListMakeImage, NeighbourListToIndices, NeighbourIndicesListMake

NeighbourListMakeChamfer

Get list of neighbours based on Chamfer metric

SYNOPSIS

```
#include "dip_neighbourlist.h"
```

```
dip_Error dip_NeighbourListMakeChamfer ( pixelsize, maxdistance, coords,
distance, resources )
```

FUNCTION

A list `coords` is created with the relative coordinates to the neighbours of a pixel in an `pixelsize->size`-dimensional image. Here, neighbours are all pixels within a `maxdistance` distance. `pixelsize` gives the size of a pixel, and hence controls the size of the neighbourhood with `maxdistance`. Anisotropic pixel grids are supported. `distance` contains the distance to each of the neighbours in `coords`.

Distances between two pixels are taken to be the Euclidean distance. There are better metrics described in the literature for small neighbourhoods, that yield a more isotropic measure when compounded over longer distances. These are not implemented in this function.

`coords` and `distance` are allocated and tracked in `resources`.

ARGUMENTS

Data type	Name	Description
`dip_FloatArray`	`pixelsize`	Physical dimensions of the pixels
`dip_int`	`maxdistance`	Maximum distance to which select neighbours
`dip_CoordinateArray *`	`coords`	Output neighbour coordinates
`dip_FloatArray *`	`distance`	Output distances to neighbours
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`NeighbourListMake`, `NeighbourListMakeChamfer`, `NeighbourListMakeImage`, `NeighbourListToIndices`, `NeighbourIndicesListMake`

NeighbourListMakeImage

Get list of neighbours based on metric in image

SYNOPSIS

```
#include "dip_neighbourlist.h"
dip_Error dip_NeighbourListMakeImage ( metric, coords, distance, resources )
```

FUNCTION

A list `coords` is created with the relative coordinates to the neighbours of a pixel in an image, with dimensionality as in `metric`. `metric` is an image that specifies the distance to each of the neighbours. This image must be odd in size, the centre pixel is the origin of the neighbourhood. Any pixel with value 0 is not considered part of the neighbourhood. `distance` contains the distance to each of the neighbours in `coords`.

Distances between two pixels are taken to be the Euclidean distance. There are better metrics described in the literature for small neighbourhoods, that yield a more isotropic measure when compounded over longer distances. These are not implemented in this function.

`coords` and `distance` are allocated and tracked in `resources`.

ARGUMENTS

Data type	Name	Description
dip_Image	metric	Image whose pixel values indicate the neighbour distance
dip_CoordinateArray *	coords	Output neighbour coordinates
dip_FloatArray *	distance	Output distances to neighbours
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

NeighbourListMake, NeighbourListMakeChamfer, NeighbourListMakeImage, NeighbourListToIndices, NeighbourIndicesListMake

NeighbourListToIndices

Get indices to neighbours

SYNOPSIS

```
#include "dip_neighbourlist.h"
dip_Error dip_NeighbourListToIndices ( coords, stride, indices, resources )
```

FUNCTION

This function translates the relative coordinates in `coords` into relative indices into an image with strides given by `stride`.

indices is allocated and tracked in `resources`.

ARGUMENTS

Data type	Name	Description
dip_CoordinateArray	coords	Input neighbour coordinates
dip_IntegerArray	stride	Stride array
dip_IntegerArray *	indices	Output neighbour indices
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

NeighbourListMake, NeighbourListMakeChamfer, NeighbourListMakeImage, NeighbourListToIndices, NeighbourIndicesListMake

NormaliseSum

Normalise the sum of the pixel values

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_NormaliseSum ( in, out, newSum )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function normalizes the sum of the pixel values in `in` to `newSum`, and puts the result in `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	newSum	New sum

NotEqual

Compare grey values in two images

SYNOPSIS

`dip_Error dip_NotEqual ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to "true" when corresponding pixels in `in1` and `in2` are different. This is the same as `Compare` with the `DIP_SELECT_NOT_EQUAL` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to the functionality of `NotZero`, but with more options.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

`Compare`, `Threshold`, `Equal`, `Greater`, `Lesser`, `NotGreater`, `NotLesser`, `SelectValue`, `NotZero`

NotGreater

Compare grey values in two images

SYNOPSIS

`dip_Error dip_NotGreater ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to “true” when for corresponding pixels `in1` <= `in2`. This is the same as Compare with the `DIP_SELECT_LESSER_EQUAL` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to a functionality similar to that of Threshold.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

Compare, Threshold, Equal, Greater, Lesser, NotEqual, NotLesser, SelectValue, NotZero

NotLesser

Compare grey values in two images

SYNOPSIS

`dip_Error dip_NotLesser ( in1, in2, out )`

DATA TYPES

binary, integer, float

FUNCTION

This function sets each pixel in `out` to "true" when for corresponding pixels `in1` >= `in2`. This is the same as Compare with the `DIP_SELECT_GREATER_EQUAL` selector flag.

`in2` can be a 0D image for comparison of pixel values with a single scalar value. This leads to a functionality similar to that of Threshold.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output

SEE ALSO

Compare, Threshold, Equal, Greater, Lesser, NotEqual, NotGreater, SelectValue, NotZero

NotZero

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_NotZero ( in, out )
```

DATA TYPES

integer, float

FUNCTION

This function returns a binary image with value 1 where `in != 0` and value 0 elsewhere. The opposite can be accomplished with SelectValue: `dip_SelectValue(in,out,0);`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image

SEE ALSO

Threshold, SelectValue, Compare, RangeThreshold

ObjectToMeasurement

Convert object label value to measurement value

SYNOPSIS

```
#include "dip_measurement.h"
```

```
dip_Error dip_ObjectToMeasurement ( object, intensity, out, connectivity,
objectID, featureID, measurementDim )
```

DATA TYPES

`objectIm`: **integer** `intensityIm`: integer, **float**

FUNCTION

This function produces an output image which pixel intensities are equal to the measurement value that the `featureID` measurement function measured on the object who label is defined by the pixel intensity of the corresponding pixel in `object`. This function is therefore useful to select (i.e. threshold) objects on basis of a measurement perfomed on the object. `intensity` provides pixel intensity information for measurements that require pixel intensity information of the objects, whose shape is defined by `object`.

The list of object IDs on which the measurements have to be performed is specified by `objectID`. If it is zero, `ObjectToMeasurement` will call `GetObjectLabels` to obtain a list of all non-zero values in `objectIm`.

If the `featureID` measurement function produces an array of measurement values, `measurementDim` will be used to select the desired array element.

ARGUMENTS

Data type	**Name**	**Description**
`dip_Image`	`object`	Object label image
`dip_Image`	`intensity`	Object intensity image
`dip_Image`	`out`	Output image
`dip_int`	`connectivity`	Connectivity of object's contour pixels, see The connectivity parameter
`dip_IntegerArray`	`objectID`	Array of Object IDs
`dip_int`	`featureID`	Measurement function ID
`dip_int`	`measurementDim`	Measurement results array index

SEE ALSO

Measure, SmallObjectsRemove, MeasurementToImage, MeasurementToHistogram

OneDimensionalSearch

Numerical algorithm

SYNOPSIS

```
#include "dip_numerical.h"
dip_Error dip_OneDimensionalSearch ( result, min, max, tol, func, dfunc,
data, searchfor )
```

FUNCTION

This function implements a numerical line-search for either the minimum or zero-crossing of a function. The obejctive is searched for in the range specified by `min` and `man` with a tolerance of `tol`. The search methods are based on Brent's algorithm. The `dfunc` parameter is preparation for support of search algorithms using derivative information. This is not supported in the current implementation, and `dfunc` should be set to zero.

ARGUMENTS

Data type	Name	Description
dip_float *	result	Result value
dip_float	min	Minimum value of search domain
dip_float	max	Maximum value of search domain
dip_float	tol	Tolerance
dip_OneDimensionalSearchFunction	func	Function
dip_OneDimensionalSearchFunction	dfunc	Derivative function
void *	data	User-supplied Data passed to `func` and `dfunc`
dipf_OneDimensionSearch	searchfor	Search for minimum of zero-crossing

Opening

Morphological opening operation

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Opening ( in, out, se, boundary, param, shape )
```

DATA TYPES

integer, **float**, **binary**

FUNCTION

Grey-value opening with different structuring elements.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element

The enumerator dip_FilterShape contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use se as filter window, can be any size

SEE ALSO

Closing, Dilation, Erosion

Or

logic operation

SYNOPSIS

`dip_Error dip_Or ( in1, in2, out )`

DATA TYPES

binary

FUNCTION

The function `Or` performs the logic OR operation between the corresponding pixels in `in1` and `in2`, and stores the result in `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First binary input image
dip_Image	in2	Second binary input image
dip_Image	out	Output image

SEE ALSO

Arith, And, Xor, Invert

ovl.h

Call an overloaded function

SYNOPSIS

```
dip_DataType ovlDataType = [current data type];
[#define DIP_OVL_ASSIGN [assignment target]]
#define DIP_OVL_FUNC  [function base name]
#define DIP_OVL_ARGS  [argument list]
#define DIP_OVL_ALLOW [type identifiers]
#include "dip_ovl.h"
```

FUNCTION

Call a type specific function based on the data type stored in the `ovlDataType` variable. The base name of the function is passed to `dip_ovl.h` by defining `DIP_OVL_FUNC`. The argument list is passed by defining `DIP_OVL_ARGS`. By defining `DIP_OVL_ALLOW` the list of data types for which overloading is possible can be controlled. If `DIP_OVL_ALLOW` is not defined, all data types are allowed. The list is specified by a logical OR of identifier and identifier group flags, see the table at DIPlib's data types. The code executed by `dip_ovl.h` is the following:

```
/* if ovlDataType is in the list specified by DIP_OVL_ALLOW */
DIPXJ( DIP_FUNC(DIP_OVL_FUNC,ovlDataType's extension) DIP_OVL_ARGS );
/* if ovlDataType is not in the list specified by DIP_OVL_ALLOW */
DIPSJ( DIP_E_DATA_TYPE_NOT_SUPPORTED );
```

`DIP_FUNC` is described in `macros.h`. Note that there are no brackets around `DIP_OVL_ARGS`, so they must be included in `DIP_OVL_ARGS` itself. If `ovlDataType` is one of the binary types, `DIP_OVL_BINARY_ARGS` can be defined to override `DIP_OVL_ARGS`.

If `DIP_OVL_ASSIGN` is defined, the following code will be executed by `dip_ovl.h` instead of the code shown above:

```
DIP_OVL_ASSIGN DIP_FUNC(DIP_OVL_FUNC,ovlDataType's extension) DIP_OVL_ARGS;
```

Note that to actually perform an assignment the "=" operator must be included in the definition of `DIP_OVL_ASSIGN` itself. `DIP_OVL_BINARY_ASSIGN` overrides `DIP_OVL_ASSIGN` if `ovlDataType` is one of the binary data types.

SEE ALSO

DIPlib's data types

`DataTypeGetInfo`, `tpi.h`

PaintBox

Paint a box

SYNOPSIS

```
#include "dip_paint.h"
dip_Error dip_PaintBox ( im, length, orign, amplitude )
```

DATA TYPES

binary, integer, **float**, complex

FUNCTION

Paints an box object in the image by replacing the values of the pixels in `im` that lie within the box (as specified by `length` and `origin`) with `amplitude`, and leaving the other pixel values untouched.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_FloatArray	length	Length array
dip_FloatArray	origin	Origin array
dip_float	amplitude	Pixel value of the painted ellips

SEE ALSO

PaintEllipsoid, PaintDiamond

PaintDiamond

Paint a diamond-shaped object

SYNOPSIS

```
#include "dip_paint.h"
dip_Error dip_PaintDiamond ( im, length, orign, amplitude )
```

DATA TYPES

binary, integer, **float**, complex

FUNCTION

Paints a diamond-shaped object in the image by replacing the values of the pixels in `im` that lie within the diamond (as specified by `length` and `origin`) with `amplitude`, and leaving the other pixel values untouched.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_FloatArray	length	Length array
dip_FloatArray	origin	Origin array
dip_float	amplitude	Pixel value of the painted ellips

SEE ALSO

PaintEllipsoid, PaintBox

PaintEllipsoid

Paint an ellipsoid

SYNOPSIS

```
#include "dip_paint.h"
dip_Error dip_PaintEllipsoid ( im, radius, orign, scale, amplitude )
```

DATA TYPES

binary, integer, **float**, complex

FUNCTION

Paints an elliptical object in the image by replacing the values of the pixels in `im` that lie within the ellips (as specified by `diameter` and `origin`) with `amplitude`, and leaving the other pixel values untouched.

ARGUMENTS

Data type	Name	Description
dip_Image	im	Image
dip_FloatArray	radius	Diameter array
dip_FloatArray	origin	Origin array
dip_float	amplitude	Pixel value of the painted ellips

SEE ALSO

PaintDiamond, PaintBox

PairCorrelation

Compute the pair correlation function

SYNOPSIS

```
#include "dip_analysis.h"
dip_Error dip_PairCorrelation ( object, mask, dist, probes, length, sampling,
covariance )
```

DATA TYPES

binary, **integer**

FUNCTION

This function computes the pair correlation function of the different phases in `object`. If `object` is a binary image, the image is a regarded as a two phase image. In case `object` is of the integer type, the image is regarded as a labeled image, with each integer value encoding a phase. Optionally a `mask` image can be provided to select which pixels in `object` should be used to compute the pair correlation. The `probes` variable specifies how many random point pairs should be drawn to compute the function. `Length` specifies the maximum correlation length. The correlation function can be computed using a random (`DIP_CORRELATION_ESTIMATOR_RANDOM`) or grid method (`DIP_CORRELATION_ESTIMATOR_GRID`), as specified by `sampling`. Finally `covariance` determines whether only the correlations (`DIP_FALSE`) or the covarianances (`DIP_TRUE`) have to be computed.

ARGUMENTS

Data type	Name	Description
dip_Image	object	Object image
dip_Image	mask	Mask image
dip_Distribution	dist	Ouput distribution
dip_int	probes	Number of probes
dip_int	length	Maximum correlation Length
dipf_CorrelationEstimator	sampling	Samplings method
dip_Boolean	covariance	Compute covariance

SEE ALSO

`ChordLength`, `ProbabilisticPairCorrelation`

PathOpening

Morphological filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error PathOpening ( grey, mask, out, length, closing, constrained )
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function applies DirectedPathOpening in all possible directions and takes the supremum of all results. That is, it is the supremum of all possible openings with approximately linear structuring elements of length `length`.

The number of times that DirectedPathOpening is applied is given by `((3^D)-1)/2`, with `D` the number of image dimensions. For example, in 2D there are 4 possible values for `param`: `[length,0]`, `[0,length]`, `[length,length]` and `[length,-length]` (note that, for example, `[-length,0]` produces the same result as `[length,0]`).

See DirectedPathOpening for more information.

ARGUMENTS

Data type	Name	Description
dip_Image	grey	Grey-value input image
dip_Image	mask	Mask image for ROI processing
dip_Image	out	Output image
dip_int	length	Length of structuring element (number of pixels)
dip_Boolean	closing	DIP_FALSE for path opening, DIP_TRUE for path closing
dip_Boolean	constrained	DIP_TRUE for constrained paths, DIP_FALSE for the original path opening algorithm

SEE ALSO

DirectedPathOpening, Opening, Closing, AreaOpening

Percentile

statistics function

SYNOPSIS

`dip_Error dip_Percentile ( in, mask, out, percentile, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the `perc` percentile of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_float	perc	Percentile
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median

PercentileFilter

Rank-order filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_PercentileFilter ( in, out, se, boundary, param, shape,
percentile )
```

DATA TYPES

integer, **float**

FUNCTION

Rank-order or percentile filter with different filter shapes.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	se	Custom filter shape (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Filter shape
dip_float	percentile	Percentile (%)

The enumerator `dip_FilterShape` contains the following constants:

Name	**Description**
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

MedianFilter

Phase

Arithmetic function

SYNOPSIS

`dip_Error dip_Phase ( in, out )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

Computes the phase of the input image values, and outputs a float typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Modulus, Real, Imaginary

PhysicalDimensionsCopy

Copy a Physical Dimensions

SYNOPSIS

`dip_Error dip_PhysicalDimensionsCopy ( newPhysDims, src, resources )`

FUNCTION

This function makes a copy of a PhysicalDimensions data structure.

ARGUMENTS

Data type	Name	Description
dip_PhysicalDimensions *	newPhysDims	New Physical Dimensions data structure
dip_PhysicalDimensions	src	source data structure
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

PhysicalDimensionsNew, PhysicalDimensionsFree, PhysicalDimensionsIsIsotropic

PhysicalDimensionsFree

Free a Physical Dimensions data structure

SYNOPSIS

`dip_Error dip_PhysicalDimensionsFree ( physDims )`

FUNCTION

This function free the Physical Dimensions `physDims` structure.

ARGUMENTS

Data type	Name	Description
`dip_PhysicalDimensions *`	`physDims`	Physical Dimensions data structure

SEE ALSO

PhysicalDimensionsNew, PhysicalDimensionsCopy, PhysicalDimensionsIsIsotropic

PhysicalDimensionsIsIsotropic

Checks if the Physical Dimensions are isotropic

SYNOPSIS

`dip_Error dip_PhysicalDimensionsIsIsotropic ( physDims, verdict )`

FUNCTION

This function checks whether the physical dimensions `physDims` are isotropic by checking that all its pixel dimensions and dimension units are equal to each other. If `verdict` is not zero, the result (`DIP_TRUE` or `DIP_FALSE`) is stored in `verdict`, otherwise an error is returned in case the verification fails.

ARGUMENTS

Data type	Name	Description
dip_PhysicalDimensions	physDims	Physical Dimensions data structure
dip_Boolean *	verdict	Verdict of the test

SEE ALSO

PhysicalDimensionsNew, PhysicalDimensionsFree, PhysicalDimensionsCopy

PhysicalDimensionsNew

Allocates a new Physical Dimensions structure

SYNOPSIS

`dip_Error dip_PhysicalDimensionsNew ( newPhysDims, dimensionality, dims, orig, dimUnit, intensity, offset, intensUnit, resources )`

FUNCTION

This function allocates a new Physical Dimensions structure.

A physical dimensions structure contains information about the physical dimensions of the data (of `dimensionality` dimension) in an image. It describes the position (`orig`) and size (`dims`) of a pixel in world coordinates and physical units (`dimUnits`), as well as the scaling (`intensity`) and offset (`offset`) of the pixel intensity in physical units (`intensUnit`).

Note that the initial values assigned by this function assume an isotropic pixel size. These values can be changed within the structure generated if this is not the case.

ARGUMENTS

Data type	Name	Description
`dip_PhysicalDimensions *`	`newPhysDims`	Pointer to a new Physical Dimensions data structure
`dip_int`	`dimensionality`	Dimensionality of the image
`dip_float`	`dims`	Initial value for `dimensions` along all axes
`dip_float`	`orig`	Initial value for `origin` along all axes
`char *`	`dimUnit`	Initial value for `dimensionUnits` along all axes
`dip_float`	`intensity`	Initial value for `intensity`
`dip_float`	`offset`	Initial value for `offset`
`char *`	`intensUnit`	Initial value for `intensityUnit`
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

The structure `dip_PhysicalDimensions` contains the following elements:

Data type	**Name**	**Description**
`dip_FloatArray`	`dimensions`	Dimensions of a pixel along each grid axis
`dip_FloatArray`	`origin`	Coordinates of the origin in physical units
`dip_StringArray`	`dimensionUnits`	Units for `dimensions` and `origin`
`dip_float`	`intensity`	Intensity scaling in physical units
`dip_float`	`offset`	Offset for intensity in physical units
`dip_String`	`intensityUnit`	Units for `intensity` and `offset`
`dip_Resources`	`resources`	Resource tracking; all elements within this structure are tracked here

SEE ALSO

`PhysicalDimensionsFree`, `PhysicalDimensionsCopy`, `PhysicalDimensionsIsIsotropic`

PixelHeapFree

Destroy heap structure

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelHeapFree ( heap )
```

FUNCTION

Frees all data associated to `heap` and sets `heap` to 0.

ARGUMENTS

Data type	Name	Description
`dip_PixelHeap *`	`heap`	The heap structure

SEE ALSO

`PixelHeapNew`, `StablePixelHeapNew`, `PixelQueueNew`, `PixelHeapPush`, `PixelHeapPop`, `PixelHeapIsEmpty`

PixelHeapIsEmpty

Query heap

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelHeapIsEmpty ( heap, result )
```

FUNCTION

Checks to see if there are any items on the heap. See PixelHeapNew for information on the heap data structure.

ARGUMENTS

Data type	Name	Description
dip_PixelHeap	heap	The heap structure
dip_Boolean *	result	Set to true if there are no items in the heap

SEE ALSO

PixelHeapNew, StablePixelHeapNew, PixelQueueNew, PixelHeapFree, PixelHeapPush, PixelHeapPop

PixelHeapNew

Create a new heap structure

SYNOPSIS

`#include "dip_pixelqueue.h"`

`dip_Error dip_PixelHeapNew ( heap, ndims, blocksize, order, resources )`

FUNCTION

This function allocates space for a new `dip_PixelHeap` structure. Memory allocated is tracked in `resources`. The heap is dimensioned to hold pixels from an `ndims`-dimensional image, and initially enough space is allocated for `blocksize` elements. The heap will be expanded as necessary when used.

The heap stores the coordinates, the value and the pointer to a pixel in an image. Note that the value does not need to equal the data pointed to by the pointer. `ndims` can be set to zero, in which case no coordinates are stored; this does not affect the function of the value and the pointer.

A heap is a priority queue data structure. Just like a queue, items can be added (pushed) and subtracted (popped). However, in the priority queue the item popped is always the higherst priority one: either the one with the highest-valued item (`order` is `DIP_GVSO_HIGH_FIRST`) or lowest-valued item (`order` is `DIP_GVSO_LOW_FIRST`). However, identically-valued items are stored on the heap in unpredictable order. If this order is important (such as for the GrowRegions algorithm with integer-valued pixels, use a `dip_StablePixelHeap` instead. See StablePixelHeapNew for information on the stable heap structure.

ARGUMENTS

Data type	Name	Description
`dip_PixelHeap *`	`heap`	The newly allocated heap structure
`dip_int`	`ndims`	Image dimensionality
`dip_int`	`blocksize`	Size of each allocation block
`dipf_GreyValueSortOrder`	`order`	Determines the heap's sort order
`dip_Resources`	`resources`	Resources tracking structure. See ResourcesNew

The `dipf_GreyValueSortOrder` enumeration consists of the following values:

Name	Description
`DIP_GVSO_HIGH_FIRST`	Process the pixels from high grey-value to low grey-value.
`DIP_GVSO_LOW_FIRST`	Process the pixels from low grey-value to high grey-value.

IMPLEMENTATION

When the heap grows beyond its initial size, its capacity is doubled in size by reallocating the data blocks. However, when removing pixels from the queue, the heap is not shrunk. It is assumed that the `dip_PixelHeap` structure will be destroyed as soon as the algorithm using it terminates. Reducing the memory footprint of the heap therefore has no benefit.

SEE ALSO

`StablePixelHeapNew`, `PixelQueueNew`, `PixelHeapFree`, `PixelHeapPush`, `PixelHeapPop`, `PixelHeapIsEmpty`

PixelHeapPop

Pop item onto heap

SYNOPSIS

`#include "dip_pixelqueue.h"`

`dip_Error dip_PixelHeapPop ( heap, coords, pointer, value )`

FUNCTION

Pops the next pixel from the heap. See PixelHeapNew for information on the heap data structure. `coords` is a pointer to an array of `dip_int`s, such as that obtained with `dip_IntegerArray->array`. It should have as many elements as the image dimensionality. If the stack was created with `ndims` set to 0, the `coords` pointer is ignored. `coords`, `pointer` and `value` can be `NULL` if you are not interested in either those values.

ARGUMENTS

Data type	Name	Description
dip_PixelHeap	heap	The heap structure
dip_int *	coords	Receives the coordinates of the popped item
void **	pointer	Receives the pointer of the popped item
dip_sfloat *	value	Receives the value of the popped item

SEE ALSO

PixelHeapNew, StablePixelHeapNew, PixelQueueNew, PixelHeapFree, PixelHeapPush, PixelHeapIsEmpty

PixelHeapPush

Push item onto heap

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelHeapPush ( heap, coords, pointer, value )
```

FUNCTION

Pushes a pixel onto the heap. See PixelHeapNew for information on the heap data structure. All 3 values **coords**, **pointer** and **value** are stored, except if the heap was created with **ndims** set to 0, in which case the **coords** pointer is ignored.

coords is a pointer to an array of **dip_ints**, such as that obtained with **dip_IntegerArray->array**. It should have as many elements as the image dimensionality. **pointer** is a pointer to any memory location, and **value** is the value to be used when sorting.

ARGUMENTS

Data type	Name	Description
dip_PixelHeap	heap	The heap structure
dip_int *	coords	Coordinates to be pushed
void *	pointer	Pointer to be pushed
dip_sfloat	value	Value to be pushed

SEE ALSO

PixelHeapNew, StablePixelHeapNew, PixelQueueNew, PixelHeapFree, PixelHeapPop, PixelHeapIsEmpty

PixelQueueFree

Destroy queue structure

SYNOPSIS

`#include "dip_pixelqueue.h"`

`dip_Error dip_PixelQueueFree ( queue )`

FUNCTION

Frees all data associated to `queue` and sets `queue` to 0.

ARGUMENTS

Data type	Name	Description
dip_PixelQueue *	queue	The queue structure

SEE ALSO

PixelQueueNew, PixelHeapNew, StablePixelHeapNew, PixelQueuePush, PixelQueuePop, PixelQueueIsEmpty

PixelQueueIsEmpty

Query queue

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelQueueIsEmpty ( queue, result )
```

FUNCTION

Checks to see if there are any items on the queue. See PixelQueueNew for information on the queue data structure.

ARGUMENTS

Data type	Name	Description
dip_PixelQueue	queue	The queue structure
dip_Boolean *	result	Set to true if there are no items in the queue

SEE ALSO

PixelQueueNew, PixelHeapNew, StablePixelHeapNew, PixelQueueFree, PixelQueuePush, PixelQueuePop

PixelQueueNew

Create a new queue structure

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelQueueNew ( queue, ndims, blocksize, resources )
```

FUNCTION

This function allocates space for a new `dip_PixelQueue` structure. Memory allocated is tracked in `resources`. The queue is dimensioned to hold pixels from an `ndims`-dimensional image, and initially enough space is allocated for `blocksize` elements. The queue will be expanded as necessary when used.

The queue stores the coordinates, and the pointer to a pixel in an image. `ndims` can be set to zero, in which case no coordinates are stored; this does not affect the function of the pointer.

A queue is a data structure to which items can be added (pushed) to the back, and subtracted (popped) from the front (FIFO - First In, First Out).

ARGUMENTS

Data type	Name	Description
`dip_PixelQueue *`	`queue`	The newly allocated queue structure
`dip_int`	`ndims`	Image dimensionality
`dip_int`	`blocksize`	Size of each allocation block
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

IMPLEMENTATION

The queue is stored in an array whose size can be incresed at will. This is accomplished by a linked list of blocks, each one holds `blocksize` elements. When more space is needed, a new block is simply allocated. No data needs to be moved as would be necessary when using `realloc` to change the size of the array. Blocks on the front of the queue that become empty are freed.

SEE ALSO

PixelHeapNew, StablePixelHeapNew, PixelQueueFree, PixelQueuePush, PixelQueuePop, PixelQueueIsEmpty

PixelQueuePop

Pop item from queue

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelQueuePop ( queue, coords, pointer, newiteration )
```

FUNCTION

Pops the next pixel from the queue. See PixelQueueNew for information on the queue data structure. `coords` is a pointer to an array of `dip_ints`, such as that obtained with `dip_IntegerArray->array`. It should have as many elements as the image dimensionality. If the queue was created with `ndims` set to 0, the `coords` pointer is ignored. `coords` and `pointer` can be `NULL` if you are not interested in either those values.

`newiteration`, when not `NULL`, will be set to `DIP_TRUE` if the pixel being popped is the first one in an iteration, or `DIP_FALSE` otherwise. When a new iteration starts, all pixels pushed onto the queue afterwards belong to the next iteration. This is useful in routines that use the queue for propagating boundaries, such as GrowRegions. First all boundary pixels are pushed onto the queue. The first iteration will need to process only these pixels, while at the same time push new pixels onto the queue for the second iteration. So after pushing all the initial boundary pixels onto the queue, the first iteration is started by popping the first pixel. All pixels pushed while processing this and the rest of the pixels will be pushed behind the "new iteration" marker. When the first pixel after this marker is popped, the `newiteration` boolean is set, so the program knows that the second iteration is starting. Also, the "new iteration" marker is moved to the end of the queue, so that pixels pushed subsequently will belong to iteration number 3.

ARGUMENTS

Data type	Name	Description
dip_PixelQueue	queue	The queue structure
dip_int *	coords	Receives the coordinates of the popped item
void **	pointer	Receives the pointer of the popped item
dip_Boolean *	newiteration	Set to true when the first item from an iteration is popped

SEE ALSO

PixelQueueNew, PixelHeapNew, StablePixelHeapNew, PixelQueueFree, PixelQueuePush, PixelQueueIsEmpty

PixelQueuePush

Push item onto queue

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_PixelQueuePush ( queue, coords, pointer )
```

FUNCTION

Pushes a pixel onto the queue. See PixelQueueNew for information on the queue data structure. Both `coords` and `pointer` are stored, except if the stack was created with `ndims` set to 0, in which case the `coords` values are ignored.

`coords` is a pointer to an array of `dip_ints`, such as that obtained with `dip_IntegerArray->array`. It should have as many elements as the image dimensionality. `pointer` is a pointer to any memory location.

ARGUMENTS

Data type	Name	Description
dip_PixelQueue	queue	The queue structure
dip_int *	coords	Coordinates to be pushed
void *	pointer	Pointer to be pushed

SEE ALSO

PixelQueueNew, PixelHeapNew, StablePixelHeapNew, PixelQueueFree, PixelQueuePop, PixelQueueIsEmpty

PixelTableAddRun

Add a new run to a pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableAddRun ( table, coordinate, length )
```

FUNCTION

Adds a new run to a pixel table. The new run is appended to the existing runs in the pixel table.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_IntegerArray	coordinate	Coordinate of the run
dip_int	length	Length of the run

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableCreateFilter

Create a pixel table from a filter shape

SYNOPSIS

`#include "dip_pixel_table.h"`

`dip_Error dip_PixelTableCreateFilter ( table, param, shape, se, resources )`

FUNCTION

This function allocates and creates a new pixel table data structure. The shape and dimensionality of the pixel table is specified by the `param`, `shape` and `se` parameters.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
`dip_PixelTable *`	`table`	Pointer to a pixel table
`dip_FloatArray`	`param`	Filter size
`dip_FilterShape`	`shape`	Filter shape
`dip_Image`	`se`	Structuring element
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
`DIP_FLT_SHAPE_DEFAULT`	Default filter window, same as `DIP_FLT_SHAPE_RECTANGULAR`
`DIP_FLT_SHAPE_RECTANGULAR`	Rectangular filter window, can be even in size
`DIP_FLT_SHAPE_ELLIPTIC`	Elliptic filter window, always odd in size
`DIP_FLT_SHAPE_DIAMOND`	Diamond-shaped filter window, always odd in size
`DIP_FLT_SHAPE_PARABOLIC`	Parabolic filter window (morphology only)
`DIP_FLT_SHAPE_DISCRETE_LINE`	Rotated line structuring element (morphology only)
`DIP_FLT_SHAPE_INTERPOLATED_LINE`	Rotated line structuring element, through interpolation (morphology only)
`DIP_FLT_SHAPE_PERIODIC_LINE`	(not implemented)
`DIP_FLT_SHAPE_STRUCTURING_ELEMENT`	Use `se` as filter window, can be any size

SEE ALSO

Description of DIPlib's pixel tables

`BinaryImageToPixelTable`, `GreyValuesInPixelTable`, `PixelTableToBinaryImage`

PixelTableFrameWork

FrameWork for PixelTable filters

SYNOPSIS

```
#include "dip_tprunlength.h"
dip_Error dip_PixelTableFrameWork ( in, out, boundary, process, table )
```

FUNCTION

This function provides a framework for filters that code the shape of their filter in a pixel table (run lengths). See SeparableFrameWork for details.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_BoundaryArray	boundary	Boundary conditions
dip_FrameWorkProcess	process	Process
dip_PixelTable	table	Pixel table

SEE ALSO

SeparableFrameWork

PixelTableGetDimensionality

Get the dimensionality of a pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetDimensionality ( table, dimension )
```

FUNCTION

Gets the dimensionality of the binary object that is encoded by the pixel table `table`.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	pixel table
dip_int *	dimension	pointer to a dimensionality variable

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableGetDimensions

Get the dimemsions of a pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetDimensions ( table, dimensions, resources )
```

FUNCTION

This functions gets the dimensions of the bounding box of the binary object that is encoded by the pixel table `table`.

ARGUMENTS

Data type	Name	Description
dip_PixelTable *	table	Pixel table
dip_IntegerArray *	dimensions	Pointer to a dimensions array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableGetOffsetAndLength

Converts the pixel table's runs

SYNOPSIS

`#include "dip_pixel_table.h"`

`dip_Error dip_PixelTableGetOffsetAndLength ( table, stride, offset, length, resources )`

FUNCTION

This functions converts the linked-list of runs in the pixel table `table` to two arrays of offsets and lengths. The length of these arrays equals the number of runs. The offsets are calculated by multiplying each coordinate of a run with the `stride` of that dimension. This function is useful when an image needs to be filtered with a filter that is encoded by a pixel table. Before processing the image. See also `PixelTableFrameWork`.

ARGUMENTS

Data type	Name	Description
`dip_PixelTable`	`table`	Pixel table
`dip_IntegerArray`	`stride`	Stride array
`dip_IntegerArray *`	`offset`	Pointer to offset array
`dip_IntegerArray *`	`length`	Pointer to length array
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

Description of DIPlib's pixel tables

`PixelTableNew`, `PixelTableGetOffsetAndLength`, `PixelTableCreateFilter`, `PixelTableFrameWork`

PixelTableGetOrigin

Get the origin of the pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetOrigin ( table, origin, resources )
```

FUNCTION

This function gets the origin of the pixel table `table`. All coordinates of the pixel table runs are defined relative to this origin. The origin is the pixel with coordinates (0,0), relative to the top left pixel.

Unless `PixelTableShiftOrigin` has been called, the origin is equal to the bounding box divided by 2 (integer division), meaning it is the middle pixel if the bounding box is odd in size, or the pixel to the right of the middle if it is even in size.

ARGUMENTS

Data type	Name	Description
dip_PixelTable *	table	Pixel table
dip_IntegerArray *	origin	Pointer to a origin array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableGetPixelCount

Get the number of pixels encoded in the pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetPixelCount ( table, count )
```

FUNCTION

Gets the total number of pixels of the binary object that is encoded by the Pixel table `table`.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_int *	count	pointer to count

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetOffsetAndLength

PixelTableGetRun

Get the contents of a pixel table run

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetRun ( table, run, coordinate, length )
```

FUNCTION

This functions get the the coordinate and length parameters of the `run`th run of the pixel table `table`.

ARGUMENTS

Data type	Name	Description
`dip_PixelTable`	`table`	Pixel table
`dip_int`	`run`	The run to be initialised
`dip_IntegerArray`	`coordinate`	Coordinate of the run
`dip_int *`	`length`	Length of the run

SEE ALSO

Description of DIPlib's pixel tables

`PixelTableNew`, `PixelTableSetRun`, `PixelTableAddRun`, `PixelTableGetRuns`, `PixelTableGetDimensionality`, `PixelTableGetDimensions`, `PixelTableGetOrigin`, `PixelTableGetSize`, `PixelTableGetPixelCount`, `PixelTableGetOffsetAndLength`

PixelTableGetRuns

Get the number of runs in a pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetRuns ( table, numberOfRuns )
```

FUNCTION

Gets the number of runs in a pixel table.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_int *	numberOfRuns	Point to the NumberOfRuns

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableGetSize

The number of pixels in the pixel table's bounding box

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableGetSize ( table, size )
```

FUNCTION

Gets the number of pixels in the bounding box of the pixel table `table`.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_int *	size	pointer to size

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableNew

Allocate a new pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableNew ( table, size, runs, resource s)
```

FUNCTION

Allocates a new pixel table `table`. The `size` array specifies the dimensionality of the coordinates in each run, and the sizes of the bounding box of the pixel table. The `runs` parameter specifies the number of runs in the pixel table.

ARGUMENTS

Data type	Name	Description
dip_PixelTable *	table	Pixel table
dip_IntegerArray	size	Size
dip_int	runs	Number of pixel table runs
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Description of DIPlib's pixel tables

PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetOrigin, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableSetRun

Initialises a pixel table run

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableSetRun ( table, run, coordinate, length )
```

FUNCTION

This function initialises the `run`th run of the pixel table `table`, by setting the run's coordinate to `coordinate` and its length to `length`. The pixel table must at least consist of `run` number of runs ans has to be allocated (using `PixelTableNew`).

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_int	run	The run to be initialised
dip_IntegerArray	coordinate	Coordinate of the run
dip_int	length	Length of the run

SEE ALSO

Description of DIPlib's pixel tables

`PixelTableNew`, `PixelTableGetRun`, `PixelTableAddRun`, `PixelTableGetRuns`, `PixelTableGetDimensionality`, `PixelTableGetDimensions`, `PixelTableGetOrigin`, `PixelTableGetSize`, `PixelTableGetPixelCount`, `PixelTableGetOffsetAndLength`

PixelTableShiftOrigin

Changes the origin of the pixel table

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableShiftOrigin ( table, offset )
```

FUNCTION

This function changes the origin of the pixel table `table`. By default, the origin is equal to the bounding box divided by 2 (integer division), meaning it is the middle pixel if the bounding box is odd in size, or the pixel to the right of the middle if it is even in size. After calling this function, the origin is equal to the previous origin plus the `offset`.

ARGUMENTS

Data type	Name	Description
dip_PixelTable	table	Pixel table
dip_IntegerArray	offset	An offset array, to be added to the origin

SEE ALSO

Description of DIPlib's pixel tables

PixelTableNew, PixelTableSetRun, PixelTableGetRun, PixelTableAddRun, PixelTableGetRuns, PixelTableGetDimensionality, PixelTableGetDimensions, PixelTableGetSize, PixelTableGetPixelCount, PixelTableGetOffsetAndLength

PixelTableToBinaryImage

Convert a pixel table to a binary image

SYNOPSIS

```
#include "dip_pixel_table.h"
dip_Error dip_PixelTableToBinaryImage ( table, im )
```

DATA TYPES

binary

FUNCTION

Converts the pixel table `table` to a binary image. The size of the image is set to the size of the bounding box of the pixel table.

ARGUMENTS

Data type	Name	Description
dip_PixelTable *	table	Pixel table
dip_Image	im	Binary image

SEE ALSO

Description of DIPlib's pixel tables

BinaryImageToPixelTable, PixelTableCreateFilter

PoissonNoise

Generate an image disturbed by Poisson noise

SYNOPSIS

```
#include "dip_noise.h"
dip_Error dip_PoissonNoise ( in, out, conversion, random )
```

DATA TYPES

integer, **float**

FUNCTION

Generate a Poisson noise disturbed image. The intensities of the input image divided by the conversion variable are used as mean value for the Poisson distribution. The conversion factor can be used to relate the pixel values with the number of counts. For example, the simulate a photon limited image acquired by a CCD camera, the conversion factor specifies the relation between the number of photons recorded and the pixel value it is represented by.

See PoissonRandomVariable for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	conversion	Conversion factor (photon/ADU)
dip_Random *	random	random

EXAMPLE

Get a Poisson disturbed image as follows:

```
dip_Image in, out;
dip_float conversion;
dip_Random random;

conversion = 2.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_PoissonNoise( in, out, conversion, &random ));
```

SEE ALSO

`PoissonRandomVariable`, `RandomVariable`, `RandomSeed`, `RandomSeedVector`, `UniformNoise`, `GaussianNoise`, `BinaryNoise`

PoissonRandomVariable

Poisson random variable generator

SYNOPSIS

```
#include "dip_noise.h"
dip_Error dip_PoissonRandomVariable ( random, mean, value )
```

FUNCTION

PoissonRandomVariable uses the algorithm as described in “Numerical Recipes in C, 2nd edition”, section 7.3. For values of mean larger or equal to 32 the rejection method is used.

See RandomVariable for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_float	mean	Mean value for the distribution
dip_float *	value	Poisson distributed output value

EXAMPLE

Get a Poisson random variable as follows:

```
dip_Random random;
dip_float mean, value;

mean = 23.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_PoissonRandomVariable( &random, mean, &value ));
```

LITERATURE

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. *Numerical Recipes in C, 2nd edition*, Cambridge University Press, Cambridge, 1992.

SEE ALSO

RandomVariable, RandomSeed, RandomSeedVector, UniformRandomVariable, GaussianRandomVariable, BinaryRandomVariable

ProbabilisticPairCorrelation

Compute the probabilistic pair correlation function

SYNOPSIS

```
#include "dip_analysis.h"
dip_Error dip_ProbabilisticPairCorrelation ( phases, mask, dist, probes,
length, sampling, covariance )
```

DATA TYPES

float

FUNCTION

This function computes the probabilistic pair correlation function of the different phases in `phases`. Each image in the image array `phases` is treated as a separate phase. The function assumes, but does not check, that the values in these images are with the [0 1] range. Optionally a `mask` image can be provided to select which pixels in `object` should be used to compute the pair correlation. The `probes` variable specifies how many random point pairs should be drawn to compute the function. `Length` specifies the maximum correlation length. The correlation function can be computed using a random (`DIP_CORRELATION_ESTIMATOR_RANDOM`) or grid method (`DIP_CORRELATION_ESTIMATOR_GRID`), as specified by `sampling`. Finally `covariance` determines whether only the correlations (`DIP_FALSE`) or the covarianances (`DIP_TRUE`) have to be computed.

ARGUMENTS

Data type	Name	Description
dip_ImageArray	phases	Phase image array
dip_Image	mask	Mask image
dip_Distribution	dist	Ouput distribution
dip_int	probes	Number of probes
dip_int	length	Maximum chord length
dipf_CorrelationEstimator	sampling	Samplings method
dip_Boolean	covariance	Compute covariance

SEE ALSO

`PairCorrelation`, `ChordLength`

PseudoInverse

Image restoration filter

SYNOPSIS

```
#include "dip_restoration.h"
dip_Error dip_PseudoInverse ( in, psf, out, threshold, flags )
```

FUNCTION

This function performs a basic, very noise sensitive image restoration operation by inverse filtering the image with a clipped point spread function. Each frequency in the output for which the response of the PSF is smaller than `threshold` is set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Point spread function image
dip_Image	out	Output image
dip_float	threshold	Threshold value
dipf_Restoration	flags	Restoration flags

LITERATURE

G.M.P. van Kempen, *Image Restoration in FLuorescence Microscopy*, Ph.D. Thesis, Delft University of Technology, 1999

SEE ALSO

Wiener, TikhonovMiller

PutLine

Put a line in an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_PutLine ( in, out, cor, dimension )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

Put a line in an image. Put a line orthogonally in an image. The position of the line in the image is specified by the coordinates at which its left most pixel (`cor`) is be placed and on which dimension of the image, the dimension of the line maps (`dimension`). If `in` has a different type than `out`, it will be converted to the type of `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input Line Image
dip_Image	out	Output Image
dip_IntegerArray	cor	Coordinate in the image of the left most pixel of the line
dip_int	dimension	Dimension of the image on which the line's dimension maps

SEE ALSO

GetSlice, PutSlice, GetLine

PutSlice

Put a slice in an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_PutSlice ( in, out, cor, dim1, dim2 )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

Put a slice orthogonally in an image. The position of the slice in the image is specified by the coordinates at which its upper left corner (`cor`) should be placed and on which dimensions of the image, the dimensions of the slice map (`dim1`, `dim2`). If `in` has a different type than `out`, it will be converted to the type of `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	2D Input Image
dip_Image	out	3D Output Image
dip_IntegerArray	cor	Coordinate in `out` where the upper left corner of `in` is placed
dip_int	dim1	Dimension of `in` on which the slice's first dimension maps
dip_int	dim2	Dimension of `in` on which the slice's second dimensionmaps

SEE ALSO

PutSlice, GetLine, PutLine

QuickSort

Sort a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_QuickSort ( data, size, dataType )
```

FUNCTION

Sorts a block of data (of size `size` and data type `dataType`) using the quick sort algorithm.

ARGUMENTS

Data type	Name	Description
`void *`	`data`	Data
`dip_int`	`size`	Size
`dip_DataType`	`dataType`	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

`QuickSortIndices`, `QuickSortIndices16`, `Sort`, `ImageSort`, `SortIndices`, `SortIndices16`, `ImageSortIndices`

QuickSortAnything

Sort data of any type

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_QuickSortAnything ( data, size, compareFunction, swapFunction,
tmpData )
```

FUNCTION

Sorts a block of data (of size `size`) using the quick sort algorithm. This routine requires the user to write two functions in order to fully implement the sorting procedure. These are `SortCompareFunction` and `SortSwapFunction`.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_int	size	Size
dip_SortCompareFunction	compareFunction	Function for comparing two data points
dip_SortSwapFunction	swapFunction	Function for swapping two data points, or copying one to the other
void *	tmpData	Pointer to a variable of the same type as the data, used as temporary space by some of the algorithms

SEE ALSO

General information about sorting

`SortAnything`, `SortCompareFunction`, `SortSwapFunction`

QuickSortIndices

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_QuickSortIndices ( data, indices, size, dataType )
```

FUNCTION

Sort a list of indices rather than the data itself using the quick sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint32 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

QuickSort, QuickSortIndices16, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

QuickSortIndices16

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_QuickSortIndices16 ( data, indices, size, dataType )
```

FUNCTION

Sorts a list of (16 bit) indices rather than the data itself using the quick sort algorithm.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint16 *	indices	Indices
dip_int	size	Size
dip_DataType	dataType	Data type. See DIPlib's data types

SEE ALSO

General information about sorting

QuickSort, QuickSortIndices, Sort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

RadialMaximum

statistics function

SYNOPSIS

`dip_Error dip_RadialMaximum ( in, mask, out, ps, binSize, innerRadius, center )`

DATA TYPES

binary, integer, float

FUNCTION

This function computes the radial projection of the maximum of the pixel intensities of `in`.

The radial projection is performed for the dimensions specified by `ps`. If the radial distance of a pixel to the center of the image is `r`, than the maximum of the intensities of all pixels with `n * binSize <= r < (n + 1) * binSize` is stored at position `n` in the radial dimension of `out`. The radial dimension is the first dimension to be processed (as specified by `ps`). If `innerRadius` is set to `DIP_TRUE`, the maximum radius that is projected is equal to the the smallest dimension of the to be projected dimensions. Otherwise, the maximum radius is set equal to the diagonal length of the dimensions to be processed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Binary mask (or 0)
dip_Image	out	Output
dip_BooleanArray	ps	Dimensions to project (or 0)
dip_float	binSize	Size of radial bins
dip_Boolean	innerRadius	Maximum radius
dip_FloatArray	center	Coordinates of center (or 0)

SEE ALSO

RadialSum, RadialMean, RadialMinimum, Sum, Mean, Maximum, Minimum

RadialMean

statistics function

SYNOPSIS

`dip_Error dip_RadialMean ( in, mask, out, ps, binSize, innerRadius, center )`

DATA TYPES

binary, integer, float

FUNCTION

This function computes the radial projection of the mean of the pixel intensities of `in`.

The radial projection is performed for the dimensions specified by `ps`. If the radial distance of a pixel to the center of the image is `r`, than the mean of the intensities of all pixels with `n * binSize <= r < (n + 1) * binSize` is stored at position `n` in the radial dimension of `out`. The radial dimension is the first dimension to be processed (as specified by `ps`). If `innerRadius` is set to `DIP_TRUE`, the maximum radius that is projected is equal to the the smallest dimension of the to be projected dimensions. Otherwise, the maximum radius is set equal to the diagonal length of the dimensions to be processed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Binary mask (or 0)
dip_Image	out	Output
dip_BooleanArray	ps	Dimensions to project (or 0)
dip_float	binSize	Size of radial bins
dip_Boolean	innerRadius	Maximum radius
dip_FloatArray	center	Coordinates of center (or 0)

SEE ALSO

RadialSum, RadialMaximum, RadialMinimum, Sum, Mean, Maximum, Minimum

RadialMinimum

statistics function

SYNOPSIS

`dip_Error dip_RadialMinimum ( in, mask, out, ps, binSize, innerRadius, center )`

DATA TYPES

binary, integer, float, complex

FUNCTION

This function computes the radial projection of the sum of the pixel intensities of `in`.

The radial projection is performed for the dimensions specified by `ps`. If the radial distance of a pixel to the center of the image is `r`, than the minimum of the intensities of all pixels with `n * binSize <= r < (n + 1) * binSize` is stored at position `n` in the radial dimension of `out`. The radial dimension is the first dimension to be processed (as specified by `ps`). If `innerRadius` is set to `DIP_TRUE`, the maximum radius that is projected is equal to the the smallest dimension of the to be projected dimensions. Otherwise, the maximum radius is set equal to the diagonal length of the dimensions to be processed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Binary mask (or 0)
dip_Image	out	Output
dip_BooleanArray	ps	Dimensions to project (or 0)
dip_float	binSize	Size of radial bins
dip_Boolean	innerRadius	Maximum radius
dip_FloatArray	center	Coordinates of center (or 0)

SEE ALSO

RadialSum, RadialMean, RadialMaximum, Sum, Mean, Maximum, Minimum

RadialSum

statistics function

SYNOPSIS

`dip_Error dip_RadialSum ( in, mask, out, ps, binSize, innerRadius, center )`

DATA TYPES

binary, integer, float

FUNCTION

This function computes the radial projection of the sum of the pixel intensities of `in`.

The radial projection is performed for the dimensions specified by `ps`. If the radial distance of a pixel to the center of the image is `r`, than the sum of the intensities of all pixels with `n * binSize <= r < (n + 1) * binSize` is stored at position `n` in the radial dimension of `out`. The radial dimension is the first dimension to be processed (as specified by `ps`). If `innerRadius` is set to `DIP_TRUE`, the maximum radius that is projected is equal to the the smallest dimension of the to be projected dimensions. Otherwise, the maximum radius is set equal to the diagonal length of the dimensions to be processed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Binary mask (or 0)
dip_Image	out	Output
dip_BooleanArray	ps	Dimensions to project (or 0)
dip_float	binSize	Size of radial bins
dip_Boolean	innerRadius	Maximum radius
dip_FloatArray	center	Coordinates of center (or 0)

SEE ALSO

RadialMean, RadialMaximum, RadialMinimum, Sum, Mean, Maximum, Minimum

RandomSeed

Initialise random number generator

SYNOPSIS

```
#include "dip_noise.h"
dip_Error dip_RandomSeed ( random, seed )
```

FUNCTION

Initializes a `dip_Random` structure using a given seed value. If `seed` is 0, the default value of 5489 is used instead, since 0 produces a uniquely poor initialisation.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_uint32	seed	Seed value

EXAMPLE

Initialize a `dip_Random` structure as follows:

```
dip_Random random;
dip_uint32 seed;

seed = 123758;
DIPXJ( dip_RandomSeed( &random, seed ));
```

SEE ALSO

RandomSeedVector, RandomVariable, UniformRandomVariable, GaussianRandomVariable, PoissonRandomVariable, BinaryRandomVariable

RandomSeedVector

Initialise random number generator

SYNOPSIS

```
#include "dip_noise.h"
dip_Error RandomSeedVector ( random, seedvector )
```

FUNCTION

Initializes a `dip_Random` structure with a given seed value vector. The vector must have `DIP_MT_STATE_SIZE` (==624) values. This is an alternative to `RandomSeed`, which, by initialising with a 32-bit integer, only gives 4 billion different sequences. `RandomSeedVector` allows to initialise the whole status of the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_uint32[DIP_MT_STATE_SIZE]	seedvector	Seed value vector

SEE ALSO

`RandomSeed`, `RandomVariable`, `UniformRandomVariable`, `GaussianRandomVariable`, `PoissonRandomVariable`, `BinaryRandomVariable`

RandomVariable

Random number generator

SYNOPSIS

include "dip_noise.h"

dip_Error dip_RandomVariable (random, value)

FUNCTION

Generates a random number between zero and one. The dip_Random structure must be initialized with the function RandomSeed. If the supplied dip_Random structure is not initialized, RandomVariable will initialize the dip_Random structure with the default seed. To guarantee the (psuedo) randomness between variables obtained with subsequent calls to RandomVariable (or to functions that use this function to obtain a random variable), a pointer to the same dip_Random structure has to supplied when calling RandomVariable.

The random number generator returns random numbers as 32-bit integers, which are normalised to to [0,1] range. If higher precision numbers are required, you can set random.highprecision to DIP_TRUE. This causes two random 32-bit values to be used for each floating point output value, doubling the precision of the output. There is no need to re-initialise the random structure after changing this setting.

This function is based on LGPL code by Geoff Kuenning (mtwist-0.8) implementing the Mersenne Twister pseudo-random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_float *	value	Random value

EXAMPLE

Obtain a random number as follows:

```
dip_Random random;
dip_float val;

DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_RandomVariable( &random, &val ));
```

LITERATURE

Makoto Matsumoto and Takuji Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Transactions on Modeling and Computer Simulation 8(1):3-30, 1998.

SEE ALSO

`RandomSeed`, `RandomSeedVector`, `UniformRandomVariable`, `GaussianRandomVariable`, `PoissonRandomVariable`, `BinaryRandomVariable`

Code source: mtwist-0.8 or mtwist-0.8

RangeThreshold

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_RangeThreshold ( in, out, lowerBound, upperBound, foreground,
background, binaryOutput )
```

DATA TYPES

integer, **float**

FUNCTION

`out` = (`lowerBound` <= `in` <= `upperBound` ? `foreground` : `background`) If the boolean `binaryOutput` is true, `RangeThreshold` will produce a binary image. Otherwise an image of the same type as the input image is produced, with the pixels set to either `foreground` or `background`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	lowerBound	Lower bound
dip_float	upperBound	Upper bound
dip_float	foreground	Foreground value
dip_float	background	Background value
dip_Boolean	binaryOutput	Convert output image to binary

SEE ALSO

Threshold, HysteresisThreshold, IsodataThreshold

Real

Arithmetic function

SYNOPSIS

`dip_Error dip_Real ( in, out )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

Computes the real part of the input image values, and outputs a float typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Modulus, Phase, Imaginary

Reciprocal

arithmetic function

SYNOPSIS

`dip_Error dip_Reciprocal ( in, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Computes the reciprocal (1/x) of the input image values. If pixel values of `in` are zero, the corresponding pixels in `out` is set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Div, DivFloat, DivComplex, Modulo

Register

Generic registry function

SYNOPSIS

`#include "dip_registry.h"`

`dip_Error dip_Register ( register )`

FUNCTION

The Registry functions `Register`, `Unregister`, `RegisterClass`, `RegistryList`, `RegistryGet` `RegistryValid` and `RegistryArrayNew` are the access functions for DIPlib's generic registry framework. These functions control the access to a registry containing information of items that are registered at run-time. Each item belongs to a certain class and is identified with an ID that is unique within the item's class.

DIPlib's Registry classes are registered at run-time as well (using `RegisterClass`) and should be registered before an item of that class can registered.

Although the generic Registry functions can be used to register, and obtain the data of registered items of a specific clas, it is more user friendly to use class-specific Registry functions like `MeasurementFeatureRegister` and companions.

The `dip_Register` function accepts one argument, a `dip_Registry` structure, which contains the `ID` and `class` of the to be registered data and `registry`, a pointer to class-specific data. Note that this pointer, `registry`, is freed when the (global) registry information is freed.

The following code gives an example of a class-specific register function:

```
dip_Error dip_MeasurementFeatureRegister
(
   dip_MeasurementFeatureRegistry registry
)
{
   DIP_FN_DECLARE("dip_MeasurementFeatureRegister");
   dip_Registry globalRegistry;
   void *data;
   dip_MeasurementFeatureRegistry *reg;

   switch( registry.type )
   {
      default:
         DIPSJ( DIP_E_REGISTRY_INCOMPLETE_REGISTRY );
         break;
```

```
      case DIP_MSR_FUNCTION_LINE_BASED:
         DIPTS( ! ( registry.create &&
              registry.measure.line &&
              registry.value &&
              registry.labels &&
              registry.description ),
          DIP_E_REGISTRY_INCOMPLETE_REGISTRY );
         break;

      case DIP_MSR_FUNCTION_IMAGE_BASED:
         DIPTS( ! ( registry.create &&
              registry.measure.image &&
              registry.value &&
              registry.labels &&
              registry.description ),
          DIP_E_REGISTRY_INCOMPLETE_REGISTRY );
         break;

      case DIP_MSR_FUNCTION_CHAINCODE_BASED:
         DIPTS( ! ( registry.create &&
              registry.measure.chaincode &&
              registry.value &&
              registry.labels &&
              registry.description ),
          DIP_E_REGISTRY_INCOMPLETE_REGISTRY );
         break;

      case DIP_MSR_FUNCTION_COMPOSITE:
         DIPTS( ! ( registry.create &&
              registry.measure.composite &&
              registry.value &&
              registry.convert &&
              registry.description ),
          DIP_E_REGISTRY_INCOMPLETE_REGISTRY );
         break;
   }
   /* copy the Measurement specific registry info */
   DIPXJ( dip_MemoryNew( &data, sizeof( dip_MeasurementFeatureRegistry ), 0 ));
   reg  = ( dip_MeasurementFeatureRegistry * ) data;
   *reg = registry;

   globalRegistry.id       = registry.id.rtid;
   globalRegistry.class    = DIP_REGISTRY_CLASS_MEASUREMENT;
   globalRegistry.registry = reg;
   globalRegistry.free    = dip_MemoryFree;
```

```
   /* register this measurement registry data */
   DIPXJ( dip_Register( globalRegistry ));

dip_error:
   DIP_FN_EXIT;
}
```

ARGUMENTS

Data type	Name	Description
dip_Registry	registry	Generic registry structure

SEE ALSO

Unregister, RegisterClass, RegistryList, RegistryGet, RegistryValid, RegistryArrayNew

RegisterClass

Register a registry class

SYNOPSIS

```
#include "dip_registry.h"
dip_Error dip_RegisterClass ( class )
```

FUNCTION

This function registers a Registry class. See Register for more information about DIPlib's Registry.

ARGUMENTS

Data type	Name	Description
dip_int	class	Registry class ID

SEE ALSO

Register, Unregister, RegistryValid, RegistryList, RegistryGet, RegistryArrayNew

RegistryArrayNew

Allocate a registry array

SYNOPSIS

`#include "dip_registry.h"`

`dip_Error dip_RegistryArrayNew ( array, size, resources )`

FUNCTION

This function allocates an array of `dip_Registry` structures.

ARGUMENTS

Data type	Name	Description
`dip_RegistryArray *`	`array`	Pointer to the allocated array
`dip_int`	`size`	Array size
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`Register`, `RegistryList`, `RegistryGet`

RegistryGet

Get a registry item

SYNOPSIS

```
#include "dip_registry.h"
dip_Error dip_RegistryGet ( id, class, registry )
```

FUNCTION

This function obtains the Registry information of the `ID` of the Registry class `class`. See `Register` for more information about DIPlib's Registry.

The following code gives an example of a class-specific register list function:

```
dip_Error dip_MsrRegistryGet
(
   dip_int id,
   dip_MsrRegistry *registry
)
{
   DIP_FN_DECLARE("dip_MsrRegistryGet");
   dip_MsrRegistry *reg;
   void *data;

   DIPXJ( dip_RegistryGet ( id, DIP_REGISTRY_CLASS_MEASUREMENT, &data ));
   reg = data;
   *registry = *reg;

dip_error:
   DIP_FN_EXIT;
}
```

ARGUMENTS

Data type	Name	Description
dip_int	id	Registry ID
dip_int	class	Registry class
dip_void **	registry	Pointer to registered data

SEE ALSO

Register, Unregister, RegisterClass, RegistryList, RegistryValid, RegistryArrayNew

RegistryList

Get an array of registry IDs

SYNOPSIS

```
#include "dip_registry.h"
dip_Error dip_RegistryList ( id, class, resources )
```

FUNCTION

This function obtains an array of the registered IDs of the Registry class `class`. See `Register` for more information about DIPlib's Registry.

The following code gives an example of a class-specific register list function:

```
dip_Error dip_MsrRegistryList
(
   dip_IntegerArray *measurement,
   dip_Resources resources
)
{
   DIP_FN_DECLARE("dip_MsrRegistryList");

   DIPXJ( dip_RegistryList( measurement,
      DIP_REGISTRY_CLASS_MEASUREMENT, resources ));

dip_error:
   DIP_FN_EXIT;
}
```

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	id	Pointer to an array of Registry IDs
dip_int	class	Registry class
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

Register, Unregister, RegisterClass, RegistryGet, RegistryValid, RegistryArrayNew

RegistryValid

Validate an registry item

SYNOPSIS

```
#include "dip_registry.h"
dip_Error dip_RegistryValid ( id, class, verdict )
```

FUNCTION

This function checks whether `id` has been registered in the Registry in the Registry class `class`. If `verdict` is not zero, the validation information (`DIP_FALSE` or `DIP_TRUE`) is copied to `verdict`. Otherwise an error is returned in case the validation fails.

See Register for more information about DIPlib's Registry.

ARGUMENTS

Data type	Name	Description
dip_int	id	Registry ID
dip_int	class	Registry class
dip_Boolean *	verdict	Pointer to a boolean containing the validation data

SEE ALSO

Register, Unregister, RegisterClass, RegistryList, RegistryGet, RegistryArrayNew

Resampling

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Resampling ( in, out, zoom, shift, method )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function resmaples the input image `in` to `out` using various interpolation methods. Both a (subpixel) `shift` and a `zoom` factor are supported. The size of the output image is `zoom` times the size of `in`. If `shift` is zero, a shift of zero is assumed. If `zoom` is zero, a zoom of 1.0 is assumed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_FloatArray	zoom	Zoom factor
dip_FloatArray	shift	Shift
dipf_Interpolation	method	Interpolation method

The `dipf_Interpolation` enumaration consists of the following constants:

Name	Description
DIP_INTERPOLATION_DEFAULT	Default interpolation method
DIP_INTERPOLATION_BSPLINE	B-Spline interpolation
DIP_INTERPOLATION_FOURTH_ORDER_CUBIC	Forth order cubic interpolation
DIP_INTERPOLATION_THIRD_ORDER_CUBIC	Third order cubic interpolation
DIP_INTERPOLATION_LINEAR	Linear interpolation
DIP_INTERPOLATION_ZERO_ORDER_HOLD	Zero order hold interpolation

SEE ALSO

Subsampling

ResourcesFree

Free resources

SYNOPSIS

`dip_Error dip_ResourcesFree( resources, flags )`

FUNCTION

Free all resources registers in the resource tracking structure, as well as the resource tracking structure itself. To prevent errors, the resource tracking structure is set to 0. Passing a null pointer instead of a pointer to a dip_Resources structure is allowed and silently ignored.

ARGUMENTS

Data type	Name	Description
dip_Resources *	resources	The tracking structure which was used to register the resources that must be freed. Note the double pointer, allowing this routine to set your pointer to 0
dipf_ResourcesFree	flags	When set to DIP_RMRF_DONT_FREE, dip_ResourcesFree only frees the dip_Resources structure itself, not the resources associated with it

SEE ALSO

ResourcesNew, ResourcesMerge, ResourceSubscribe, ResourceUnsubscribe

ResourcesMerge

Add one resource list to another

SYNOPSIS

`dip_Error dip_ResourcesMerge( resources, mergee )`

FUNCTION

Adds one resource list to another. This function is very useful when writing functions that will support a `dip_Resources` parameter. Typically you want to allocate a number of resources and only add these to the user-supplied `dip_Resources` when all these allocations have been successful. This is where `ResourcesMerge` comes in. Allocate a local `dip_Resources` structure and register all resources with it. When no errors occured the local `dip_Resources` structure can be merged with the user-supplied `dip_Resources` structure. If an error did occur, simply free all local resources by calling `ResourcesFree`. In addition it is the convention that functions supporting resource tracking also accept a zero indicating that no tracking should be performed. When `resources` in `dip_ResourcesMerge` is 0, the `mergee` tracking structure is freed, but the resources it contains are not. In this way you get support for the "`resources = 0` means no tracking" convention for free.

ARGUMENTS

Data type	Name	Description
`dip_Resources`	`resources`	The dip_Resources structure with which the additional resources much be merged
`dip_Resources *`	`mergee`	A pointer to the `dip_Resources` structure containing the additional resources to be merged. After the merge, `mergee` is set to 0.

SEE ALSO

`ResourcesNew`, `ResourcesFree`, `ResourceSubscribe`, `ResourceUnsubscribe`

ResourcesNew

Allocate a resource tracking structure

SYNOPSIS

`dip_Error dip_ResourcesNew( resources, noItems )`

FUNCTION

This function allocates a `dip_Resources` structure. The resource structure can be used to register various resources as they are allocated, provided that the allocating function allows you to register the resource. All resources allocated in this manner can be freed with a single call to `ResourcesFree`. Examples of functions supporting this registration scheme are `ImageNew` and special versions of the memory allocation routines.

Some operations consist of an initialization and a cleanup stage. These stages are often performed by separate routines to allow the user to execute his/her own operations in between. In DIPlib there usually is no directly callable cleanup function. Instead the initialization routine registers its cleanup routine with a `dip_Resources` structure provided by the user. The cleanup operation is invoked through `ResourcesFree`.

All functions that support the registration leave you the choice not to register the resource. This is indicated by supplying a zero instead of a resource tracking structure, unless documented otherwise. The `noItems` parameters can be used to give the routine a hint about the number of resources you will register. The structure grows automagically whenever more resources are registered than indicated by the hint parameter.

ARGUMENTS

Data type	Name	Description
`dip_Resources *`	`resources`	This will be used to return a dip_Resources structure
`dip_int`	`noItems`	A hint about the number of resources you are planning to allocate. This parameter must be >= 2 or 0 to indicate you want the default value. By the way, don't worry too much about this parameter, because when the structure turns out to be too small, it will automatically be expanded

SEE ALSO

`ResourcesFree`, `ResourcesMerge`, `ResourceSubscribe`, `ResourceUnsubscribe`, `MemoryNew`

ResourceSubscribe

Track a resource

SYNOPSIS

`dip_Error dip_ResourceSubscribe( resource, freeResourceHandler, resources )`

FUNCTION

Track a resource. The resource must be represented by a `void *`. A handler function to free the resource must be given. This function will be called through dip_ResourcesFree with the resource as its only parameter. If `dip_ResourceSubscribe` fails, the resource is not registered. It is allowed to pass a zero instead of a `dip_Resources` structure, in which case `dip_ResourceSubscribe` returns silently.

ARGUMENTS

Data type	Name	Description
`void *`	`resource`	The resource that must be registered
`dip_ResourcesFreeHandler`	`freeResourceHandler`	The handler function that will be invoked by `dip_ResourcesFree` to free the resource
`dip_Resources`	`resources`	`dip_Resources` structure to register the resource with

SEE ALSO

ResourcesNew, ResourcesFree, ResourcesMerge, ResourceUnsubscribe

ResourceUnsubscribe

Stop tracking a resource

SYNOPSIS

`dip_Error dip_ResourceUnsubscribe( resource, resources )`

FUNCTION

Stop tracking a resource. It will be removed from the `dip_Resources` structure. The resource itself will not be freed. If a zero is passed instead of a resource or the `dip_Resources` structure, `dip_ResourceUnsubscribe` returns silently.

ARGUMENTS

Data type	Name	Description
`void *`	`resource`	The resource that should no longer be tracked
`dip_Resources`	`resources`	`dip_Resources` structure to remove the resource from

SEE ALSO

ResourcesNew, ResourcesFree, ResourcesMerge, ResourceSubscribe

RootMeanSquareError

difference measure

SYNOPSIS

`dip_Error dip_RootMeanSquareError ( in1, in2, mask, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Calculates the root mean square difference between each pixel value of `in1` and `in2`. Optionally the `mask` image can be used to exclude pixels from the calculation by setting the value of these pixels in `mask` to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	mask	Mask
dip_Image	out	Output

SEE ALSO

MeanError, MeanSquareError, MeanAbsoluteError, LnNormError, IDivergence

Rotation

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Rotation ( in, out, angle, method, bgval )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function rotates an 2-D image `in` over `angle` to `out` using three skews. The function implements the rotation in the mathmetical sense, **but** note the Y-axis is positive downwards! The rotation is over the centre of the image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	angle (radians)	Rotation angle
dipf_Interpolation	method	Interpolation method
dip_BackgroundValue	bgval	Background value

The `dipf_Interpolation` enumaration consists of the following constants:

Name	Description
DIP_INTERPOLATION_DEFAULT	Default interpolation method
DIP_INTERPOLATION_BSPLINE	B-Spline interpolation
DIP_INTERPOLATION_FOURTH_ORDER_CUBIC	Forth order cubic interpolation
DIP_INTERPOLATION_THIRD_ORDER_CUBIC	Third order cubic interpolation
DIP_INTERPOLATION_LINEAR	Linear interpolation
DIP_INTERPOLATION_ZERO_ORDER_HOLD	Zero order hold interpolation

The `dip_BackgroundValue` enumaration consists of the following flags:

Name	Description
DIP_BGV_DEFAULT	Default: fill with zeros
DIP_BGV_ZERO	Fill with zeros
DIP_BGV_MAX_VALUE	Fill with maximum value for data type
DIP_BGV_MIN_VALUE	Fill with minimum value for data type

KNOWN BUGS

This function is only implemented for 2D images.

SEE ALSO

Skewing

Rotation3d

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Rotation3d ( in, out, alpha, beta, gamma, method, bgval )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function rotates an 3-D image `in` via the three Euler angles `alpha`, `beta`, `gamma` to `out` using nine skews. The first rotation is about `alpha` around the initial 3-axis. The second about `beta` around the intermediate 2-axis and the last about `gamma` around the final 3-axis. The function implements the rotation in the mathmetical sense, **but** note the Y-axis is positive downwards! The rotation is over the centre of the image.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_float`	`alpha (radians)`	Euler angle
`dip_float`	`beta (radians)`	Euler angle
`dip_float`	`gamma (radians)`	Euler angle
`dipf_Interpolation`	`method`	Interpolation method
`dip_BackgroundValue`	`bgval`	Background value

The `dipf_Interpolation` enumaration consists of the following constants:

Name	Description
`DIP_INTERPOLATION_DEFAULT`	Default interpolation method
`DIP_INTERPOLATION_BSPLINE`	B-Spline interpolation
`DIP_INTERPOLATION_FOURTH_ORDER_CUBIC`	Forth order cubic interpolation
`DIP_INTERPOLATION_THIRD_ORDER_CUBIC`	Third order cubic interpolation
`DIP_INTERPOLATION_LINEAR`	Linear interpolation
`DIP_INTERPOLATION_ZERO_ORDER_HOLD`	Zero order hold interpolation

The `dip_BackgroundValue` enumaration consists of the following flags:

Name	Description
DIP_BGV_DEFAULT	Default: fill with zeros
DIP_BGV_ZERO	Fill with zeros
DIP_BGV_MAX_VALUE	Fill with maximum value for data type
DIP_BGV_MIN_VALUE	Fill with minimum value for data type

KNOWN BUGS

This function is only implemented for 3D images.

SEE ALSO

Skewing, Rotation3d_Axis

Rotation3d_Axis

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Rotation3d_Axis ( in, out, angle, axis, method, bgval )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function rotates an 3-D image `in` over `angle` around `axis` to `out` using three skews. The rotation axis is 0 (x), 1 (y) or 2 (z). The function implements the rotation in the mathmetical sense, **but** note the Y-axis is positive downwards! The rotation is over the centre of the image.

For backwards compatability, the macro `Rotation3dAxis` calls the function `Rotation3d_Axis` but uses 1, 2 and 3 to select the axis of rotation.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	angle (radians)	Rotation angle
dip_int	axis	Rotation axis
dipf_Interpolation	method	Interpolation method
dip_BackgroundValue	bgval	Background value

The `dipf_Interpolation` enumaration consists of the following constants:

Name	Description
DIP_INTERPOLATION_DEFAULT	Default interpolation method
DIP_INTERPOLATION_BSPLINE	B-Spline interpolation
DIP_INTERPOLATION_FOURTH_ORDER_CUBIC	Forth order cubic interpolation
DIP_INTERPOLATION_THIRD_ORDER_CUBIC	Third order cubic interpolation
DIP_INTERPOLATION_LINEAR	Linear interpolation
DIP_INTERPOLATION_ZERO_ORDER_HOLD	Zero order hold interpolation

The `dip_BackgroundValue` enumaration consists of the following flags:

Name	Description
DIP_BGV_DEFAULT	Default: fill with zeros
DIP_BGV_ZERO	Fill with zeros
DIP_BGV_MAX_VALUE	Fill with maximum value for data type
DIP_BGV_MIN_VALUE	Fill with minimum value for data type

KNOWN BUGS

This function is only implemented for 3D images.

SEE ALSO

Skewing, Rotation3d

ScalarImageNew

Allocate a scalar image

SYNOPSIS

`dip_Error dip_ScalarImageNew( newImage, dataType, dimensions, resources )`

FUNCTION

Allocate and forge a `dip_Image` structure of the `DIP_IMTP_SCALAR` type.

ARGUMENTS

Data type	Name	Description
dip_Image *	newImage	Used to return a pointer to your brand new dip_Image structure
dip_DataType	dataType	Data type. See DIPlib's data types
dip_IntegerArray	dimensions	Dimensions
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

ImageNew, ImageFree

ScanFrameWork

FrameWork for scanning multiple images

SYNOPSIS

`dip_Error dip_ScanFrameWork ( in, out, process, boundary, border, inBuffer, outBuffer, outImage )`

FUNCTION

This function provides a framework for scanning `nofin` input images and `nofout` output images in one dimension of the images. The dimension in which the image should be scanned can be specified or left to `ScanFrameWork` by specifiying the dimension with `DIP_MONADIC_OPTIMAL_DIMENSION`. See `SeparableFrameWork` for details.

ARGUMENTS

Data type	Name	Description
dip_ImageArray	in	Array of input images
dip_ImageArray	out	Array of output images
dip_FrameWorkProcess	process	Process
dip_BoundaryArray	boundary	Boundary conditions
dip_BorderArray	border	Border Array
dip_DataTypeArray	inBuffer	Array of dip_DataType of the input buffer
dip_DataTypeArray	outBuffer	Array of dip_DataType of each output buffer
dip_DataTypeArray	outImage	Array of dip_DataType of each output image

SEE ALSO

DIPlib's data types `SeparableFrameWork`, `PixelTableFrameWork`

SeededWatershed

Morphological segmentation

SYNOPSIS

`#include "dip_morphology.h"`

`dip_Error dip_SeededWatershed ( seeds, in, mask, out, connectivity, order, max_depth, max_size, binaryOutput )`

DATA TYPES

integer, **float**

FUNCTION

Watershed segmentation with built-in region merging. `max_depth` and `max_size` control the merging procedure. Any region with `max_size` or less pixels **and** with `max_depth` grey-value difference or less will be merged to neighbouring regions when they touch (as opposed to build a watershed). `max_size` equal to 0 means that the size of the region is not tested when merging. To avoid merging of seeds with no grey-value difference between them, set `max_size` to a negative value. The regions are grown according to the `connectivity` parameter. See The connectivity parameter for more information. The output is either a labelled image where the pixels belonging to a catchment basin are labelled, or a binary image where the watershed pixels are 1 and the rest is 0. This is controlled by `binaryOutput`.

As opposed to `Watershed`, this function takes a `seeds` input image, and grows the catchment basins from there. The output image, when `binaryOutput` is `DIP_TRUE`, will have label values as given by the seed image.

If `mask` is not 0, only the pixels within `mask` will be considered. All the other pixels will be untouched.

ARGUMENTS

Data type	Name	Description
dip_Image	seeds	Binary or labelled input image
dip_Image	in	Grey-value input image
dip_Image	mask	Mask image
dip_Image	out	Output
dip_int	connectivity	Connectivity
dipf_GreyValueSortOrder	order	Whether to grow from low to high or high to low
dip_float	max_depth	Maximum depth of a region that can be merged
dip_int	max_size	Maximum size of a region that can be merged
dip_Boolean	binaryOutput	DIP_FALSE if the output should be a labelled image

The dipf_GreyValueSortOrder enumeration consists of the following values:

Name	Description
DIP_GVSO_HIGH_FIRST	Process the pixels from high grey-value to low grey-value.
DIP_GVSO_LOW_FIRST	Process the pixels from low grey-value to high grey-value.

SEE ALSO

Watershed, LocalMinima, Minima, Maxima, GrowRegions

Select

Configurable selection function

SYNOPSIS

`dip_Error dip_Select ( in1, in2, in3, in4, out, selector )`

DATA TYPES

binary, integer, float

FUNCTION

This function can perform various pixel-by-pixel comparisons (smaller, smaller- equal, equal, not equal, greater-equal, greater) between `in1` ans `in2`. If the result of the comparison is true, the corresponding pixel value of `in3` is copied to `out`, otherwise it is copied from `in4`. In short the following operation is performed for each pixel in the five images:

```
out = in1 selector in2 ? in3 : in4
```

The images `in2`, `in3` and `in4` can be 0-D images acting as constants.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	in3	Third input
dip_Image	in4	Fourth input
dip_Image	out	Output
dipf_Select	selector	Select flag

The `dipf_Select` flag can be one of:

Name	Description
DIP_SELECT_LESSER	<, Lesser than
DIP_SELECT_LESSER_EQUAL	<=, Lesser or equal
DIP_SELECT_NOT_EQUAL	!=, Unequal
DIP_SELECT_EQUAL	==, Equal
DIP_SELECT_GREATER_EQUAL	>=, Greater or equal
DIP_SELECT_GREATER	>, Greater

SEE ALSO

Compare, Max, Min

SelectValue

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_SelectValue ( in, out, value )
```

DATA TYPES

integer, float

FUNCTION

This function returns a binary image with value 1 where `in == value` and value 0 elsewhere.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	value	Value to select

SEE ALSO

Threshold, NotZero, Compare, RangeThreshold

SeparableConvolution

FrameWork for separable convolution filters

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_SeparableConvolution ( in, out, filters, bc, process )
```

DATA TYPES

integer, **float**

FUNCTION

This function is a frontend to the lower level `Convolve1d` function. Each dimension can be processed by a different filter.

`process` may be zero, indicating that all dimensions should be processed.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_SeparableConvolutionFilter *`	`filters`	Filters
`dip_BoundaryArray`	`bc`	Boundary conditions
`dip_BooleanArray`	`process (0)`	Dimensions to process

The `dip_SeparableConvolutionFilter` structure contains the following elements:

Data type	Name	Description
`dip_float *`	`filter`	Filter weights
`dip_int`	`filterSize`	Length of `filter` array
`dip_int`	`origin`	Origin of the filter, only valid in conjunction with `DIP_CNV_USE_ORIGIN`
`dipf_Convolve`	`flags`	Filter flags, see `Convolve1d` for their definitions

SEE ALSO

General information about convolution

`GeneralConvolution`, `ConvolveFT`, `SeparableFrameWork`, `Convolve1d`

SeparableFrameWork

FrameWork for separable filters

SYNOPSIS

`dip_Error dip_SeparableFrameWork (in, out, boundary, border, process )`

FUNCTION

The `dip_SeparableFrameWork` function is a framework for separable filters. This function takes care of all the "administrative work" involved when processing a n-D DIPlib image n times with a 1-D filter. In short, using this function, one has only to create an one dimension filter function and `dip_SeparableFrameWork` takes care of the other stuff. The `in` image is filtered `nrOfProcesses` times using the information in each element of the `process` array. If `nrOfProcesses` is zero, only the first element of `process` is used to filter `in` in all its dimensions.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input image
`dip_Image`	`out`	Output image
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_IntegerArray`	`border`	Border array
`dip_FrameWorkProcessArray`	`process`	Array of `dip_FrameWorkProcess` structures

NOTE

The `dip_FrameWorkProcess` structure contains the following members:

Data type	Name	Description
dip_Boolean	process	flags specifying to do processing or not
dipf_FrameWorkMethod	frameWorkMethod	flags specifying the method of how dip_SeparableFrameWork should transport data from in to out
dipf_FrameWorkOperation	frameWorkOperation	flags specifying requirements of the 1-D filter function
dip_int	processDimension	dimension of in to be processed
dip_int	roiOrigin	ignored in current implementation
dip_int	roiSize	ignored in current implementation
dipf_FrameWorkFilter	FrameWorkFilterType	specifying the type of 1-D filter function
dip_FrameWorkFilter	FrameWorkFilter	pointer to the 1-D filter function
void *	functionParameters	parameters of the 1-D filter function, for all dimensions
dip_DataType	inputBufferType	data type of input buffer
dip_DataType	outputBufferType	data type of output buffer
dip_DataType	suggestedOutputType	data type of output image

The dipf_FrameWorkMethod enum contains the following elements:

Name	Description
DIP_FRAMEWORK_DEFAULT_METHOD	use dip_SeparableFrameWorks most optimal method
DIP_FRAMEWORK_CLASSICAL	use a classical method
DIP_FRAMEWORK_DOUBLE_STRIPE	use two buffers to store temporary results

It is strongly advised to use the DIP_FRAMEWORK_DEFAULT_METHOD method.

The dipf_FrameWorkOperation enum contains the following elements:

Name	Description
DIP_FRAMEWORK_DEFAULT_OPERATION	default operation
DIP_FRAMEWORK_IN_PLACE	filtering operation can be performed in-place. It is up to dip_SeparableFrameWork whether the actual filtering is done in-place
DIP_FRAMEWORK_NO_IN_BORDER	the 1-D filter does not need border extension of the input data
DIP_FRAMEWORK_OUT_BORDER	the 1-D filter needs border extension of the output data
DIP_FRAMEWORK_WRITE_INPUT	the 1-D filter needs to write in the input data
DIP_FRAMEWORK_USE_BUFFER_TYPES	made the input and output buffers of the inputBufferType and outputBufferType data type
DIP_FRAMEWORK_NO_BUFFER_STRIDE	Create input and output buffers with a stride of one
DIP_FRAMEWORK_DO_NOT_ADJUST	Do not adjust the output image, just check it
DIP_FRAMEWORK_USE_OUTPUT_TYPE	Adjust output image to the suggestedOutputType data type

The dipf_FrameWorkFilter enum contains the following elements:

Name	Description
DIP_FRAMEWORK_SEPARABLE_FILTER	default filter type, process one line at the time
DIP_FRAMEWORK_TWO_LINES_SEPARABLE_FILTER	process two lines in one go
DIP_FRAMEWORK_SINGLE_OUTPUT_FILTER	this filter only needs an output buffer

The union dip_FrameWorkFunction consists of the types

Name	Description
dip_SeparableFilter	one line filter function
dip_TwoLinesSeparableFilter	two lines filter function
dip_SingleOutputFilter	single output image filter

The functions have the following arguments: dip_SeparableFilter

Data type	Name	Description
void *	inData	pointer to the input data
void *	outData	pointer to the output data
dip_int	elements	number of pixels in the inData array
dip_SeparableFilterParameters	params	parameter structure for the filter function

and dip_SeparableTwoLinesFilter

Data type	Name	Description
void *	inFirstLineData	pointer to the data of the first input line
void *	inSecondLineData	pointer to the data of the second input line
void *	outFirstLineData	pointer to the data of the first output line
void *	outSecondLineData	pointer to the data of the second output line
dip_int	elements	number of pixels in the inFirstLineData array
dip_TwoLinesSeparableFilterParameters	params	parameter structure for the two lines filter function

The inData, inFirstLineData and inSecondLineData will always point to the first pixel of the line of in that is processed. Therefore, the 1-D filter can access pixels with indices ranging from -border[dimension] up to elements + border[dimension]. If the flag DIP_FRAMEWORK_OUT_BOUNDARY is specified, the same holds for the output data pointers.

The structure dip_SeparableFilterParameters contains the following elements:

Data type	Name	Description
void *	functionParameters	parameters of the 1-D filter function per dimension
dip_int	dimension	the dimension in which direction the input buffer is taken from the input image
dip_int	processNumber	number of times dip_SeparableFrameWork has already filtered in with an 1-D filter including current filtering
dip_DataType	inType	dip_DataType of the input buffer
dip_int	inStride	stride of the elements in the input array
dip_int	inPlane	plane number in case in is a binary image
dip_DataType	outType	dip_DataType of the output buffer
dip_int	outStride	stride of the elements in the output array
dip_int	outPlane	plane number in case out is a binary image
dip_int	outDimension	the dimension in which direction the output buffer is taken from the output image
dip_IntegerArray	position	coordinate of the first pixel of the input buffer in the input image

The structure dip_TwoLinesSeparableFilterParameters contains the following elements:

Data type	Name	Description
void *	functionParameters	parameters of the 1-D filter function per dimension
dip_int	dimension	the dimension in which direction the input buffer is taken from the input image
dip_int	processNumber	number of times dip_SeparableFrameWork has already filtered in with an 1-D filter including current filtering
dip_DataType	inType	dip_DataType of the input buffer
dip_int	inStride	stride of the elements in the input array
dip_int	inPlane	plane number in case in is a binary image
dip_DataType	outType	dip_DataType of the output buffer
dip_int	outStride	stride of the elements in the output array
dip_int	outPlane	plane number in case out is a binary image
dip_int	outDimension	the dimension in which direction the output buffer is taken from the output image
dip_IntegerArray	position	coordinate of the first pixel of the input buffer in the input image

The structure dip_SingleOutputFilterParameters contains the following elements:

Data type	Name	Description
`void *`	`functionParameters`	parameters of the 1-D filter function per dimension
`dip_int`	`dimension`	the dimension in which direction the input buffer is taken from the input image
`dip_int`	`processNumber`	number of times `dip_SeparableFrameWork` has already filtered `in` with an 1-D filter including current filtering
`dip_DataType`	`type`	`dip_DataType` of the input buffer
`dip_int`	`stride`	stride of the elements in the input array
`dip_int`	`plane`	plane number in case `in` is a binary image
`dip_IntegerArray`	`position`	coordinate of the first pixel of the input buffer in the input image

SEE ALSO

DIPlib's data types `SeparableConvolution`, `MonadicFrameWork`, `SingleOutputFrameWork`, `PixelTableFrameWork`, `ScanFrameWork`

Set

the value of a pixel

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Set ( out, const, cor, adjust )
```

FUNCTION

This function set a value of a pixel at position `cor` in the image `out` to the value `const`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output image
dip_Image	constImage	0-D image
dip_IntegerArray	cor	Pixel coordinate
dip_Boolean	adjust	Adjust data type of output image

SEE ALSO

SetInteger, SetFloat, SetComplex, dip__PixelSetInteger, dip__PixelSetFloat, Get

SetComplex

Set a pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_SetComplex ( out, constant, cor, adjust )
```

FUNCTION

This function set a value of a pixel at position `cor` in the image `out` to the value `constant`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output
dip_complex	constant	Constant
dip_IntegerArray	cor	Pixel coordinate
dip_Boolean	adjust	Adjust data type of output image

SEE ALSO

Set, SetInteger, SetFloat, dip__PixelSetInteger, dip__PixelSetFloat, Get

SetFloat

Set a pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_SetFloat ( out, constant, cor, adjust )
```

FUNCTION

This function set a value of a pixel at position `cor` in the image `out` to the value `constant`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output
dip_float	constant	Constant
dip_IntegerArray	cor	Pixel coordinate
dip_Boolean	adjust	Adjust data type of output image

SEE ALSO

Set, SetInteger, SetComplex, dip__PixelSetInteger, dip__PixelSetFloat, Get

SetInteger

Set a pixel value

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_SetInteger ( out, constant, cor, adjust )
```

FUNCTION

This function set a value of a pixel at position `cor` in the image `out` to the value `constant`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output
dip_int	constant	Constant
dip_IntegerArray	cor	Pixel coordinate
dip_Boolean	adjust	Adjust data type of output image

SEE ALSO

Set, SetFloat, SetComplex, dip__PixelSetInteger, dip__PixelSetFloat, Get

Sharpen

Enhance an image

SYNOPSIS

```
#include "dip_derivatives.h"
dip_Error dip_Sharpen ( in, out, weight, bc, ps, sigmas, tc, flavour )
```

DATA TYPES

See `Laplace`

FUNCTION

This function enhances the high frequencies ("sharpens") of the input image `in` by subtracting a `Laplace` filtered version of `in` from it. The `weight` parameter determines by which amount the laplace information is subtracted from the original: `output = input - weight * laplace( input )` The `sigmas` are the Gaussian smoothing parameters of the `Laplace` operation, and determine how strongly the high-frequency noise in `in` is suppressed.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	weight	Laplacian weight
dip_BoundaryArray	boundary	Boundary conditions
dip_BooleanArray	process (0)	Dimensions to process
dip_FloatArray	sigmas	Sigma of Gaussian
dip_float	truncation (<0)	Truncation of Gaussian, see GlobalGaussianTruncationGet
dip_DerivativeFlavour	flavour	Derivative Flavour

SEE ALSO

`Laplace`

Shift

an image manipulation function

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Shift ( in, out, shift, killNy )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

This function shifts an image in the Fourier Domain. All frequiencies larger than the Nyquist frequency are set to zero if `killNy` is true. It performs:

```
out = Real(InverseFourierTransform(GeneratePhase(shift) * FourierTransform( in ))
```

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_FloatArray	shift	Shift array
dip_Boolean	killNy	set frequencies > Nyquist to zero?

SEE ALSO

Crop, Wrap, FourierTransform, Real

Sigma

Adaptive uniform smoothing filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_Sigma ( in, out, se, boundary, param, shape, sigma, outputCount )
```

DATA TYPES

integer, **float**

FUNCTION

The Sigma filter is an adaptive `Uniform` smoothing filter. The value of the pixel underinvestigation is replaced by the average of the pixelvalues in the filter region (as specified by `param`, `shape` and `se`) which lie in the interval `+/- 2 sigma` from the value of the pixel that is filtered. If `outputCount` is `DIP_TRUE`, the output values represent the number of pixels over which the average has been calculated. When `threshold` is `DIP_TRUE`, the pixel intensities are thresholded at `+/- 2 sigma`, when it is set to `DIP_FALSE`, the intensities are weighted with the Gaussian difference with the intensity of the center pixel.

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_Image	se	Custom filter window (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter sizes
dip_FilterShape	shape	Filter shape
dip_float	sigma	Sigma
dip_Boolean	outputCount	Output the Count

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

LITERATURE

John-Sen Lee, *Digital Image Smoothing and the Sigma Filter*, Computer Vision, Graphics and Image Processing, 24, 255-269, 1983

SEE ALSO

`BiasedSigma`, `GaussianSigma`, `Uniform`

Sign

Arithmetic function

SYNOPSIS

`dip_Error dip_Sign ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the sign of the input image values, and outputs a signed integer typed image. The sign of zero is defined as zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Ceil, Floor, Truncate, Fraction, NearestInt

SimulatedAttenuation

Simulation of the attenuation process

SYNOPSIS

```
#include "dip_microscopy.h"
dip_Error dip_SimulatedAttenuation ( in, out, fAttenuation, bAttenuation, NA,
refIndex, zxratio, oversample, rayStep )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function simulates an attenuation based on the model of a CSLM, using a ray tracing method.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	fAttenuation	Forward attenuation factor
dip_float	bAttenuation	Backward attenuation factor
dip_float	NA	Numerical aperture
dip_float	refIndex	Refractive index
dip_float	zxratio	Z/X sampling ratio
dip_int	oversample	Ray casting oversampling
dip_float	rayStep	Ray step

LITERATURE

K.C. Strasters, H.T.M. van der Voort, J.M. Geusebroek, and A.W.M. Smeulders, *Fast attenuation correction in fluorescence confocal imaging: a recursive approach*, BioImaging, vol. 2, no. 2, 1994, 78-92.

AUTHOR

Karel Strasters, adapted to DIPlib by Geert van Kempen.

SEE ALSO

`AttenuationCorrection`, `ExponentialFitCorrection`

Sin

trigonometric function

SYNOPSIS

`dip_Error dip_Sin ( in, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Computes the sine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Tanh

Sinc

mathematical function

SYNOPSIS

`dip_Error dip_Sinc ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the sinc (sin(x)/x) of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

BesselJ0, BesselJ1, BesselJN, BesselY0, BesselY1, BesselYN, LnGamma, Erf, Erfc

SingleOutputFrameWork

FrameWork for generation functions

SYNOPSIS

`dip_Error dip_SingleOutputFrameWork ( out, processBoundary, processBorder, process )`

FUNCTION

This function is a frontend on the SeparableFrameWork. It provides an easier interface for filters that only need to scan an single output image. The dimension in which the image should be scanned can be specified or left to `SingleOutputFrameWork` by specifiying the dimension with `DIP_MONADIC_OPTIMAL_DIMENSION`.

ARGUMENTS

Data type	Name	Description
dip_Image	out	Output
dip_Boundary	processBoundary	ProcessBoundary
dip_int	processBorder	ProcessBorder
dip_FrameWorkProcess	process	Process

SEE ALSO

SeparableFrameWork, MonadicFrameWork, PixelTableFrameWork, ScanFrameWork

SingularValueDecomposition

Singular value decomposition

SYNOPSIS

`dip_Error dip_SingularValueDecomposition ( in, sz, u, s, v )`

DATA TYPES

float

FUNCTION

Computes the SVD of the ImageArray `in`, such that `in` = `u` * `s` * transpose(`v`), with `s` being diagonal. The size of the `in` matrix is passed to the function via the integer array `sz`. If the input is of size MxN, then the outputs must be `u`: nxM, `s`: NxN, and `v`: NxN.

Optionally, set `u` and `v` to `NULL`, and let `s` have N elements, it will contain only the singular values.

ARGUMENTS

Data type	Name	Description
`dip_ImageArray`	`in`	Input
`dip_IntegerArray`	`sz`	Matrix size of Input
`dip_ImageArray`	`u`	Output
`dip_ImageArray`	`s`	Output
`dip_ImageArray`	`v`	Output

Sinh

trigonometric function

SYNOPSIS

`dip_Error dip_Sinh ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the hyperbolic sine of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Atan2, Cosh, Tanh

Skewing

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Skewing ( in, out, shear, skew, axis, method, bgval,
periodicSkew )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function skews the axis `axis` of `in` over an angle `angle` to `out` using the interpolation method `method`. The skew is over the centre of the image. If `periodicSkew` is set to `DIP_TRUE`, the output image will be of the same size as the input image, and its pixels in the `skew` dimension wrapped around the image boundaries. `bgval` is not used in this case.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	shear (radians)	Shear angle
dip_int	skew	Skew dimension
dip_int	axis	Skew axis
dipf_Interpolation	method	Interpolation method
dip_BackgroundValue	bgval	Background value
dip_Boolean	periodicSkew	Skew using periodic image boundaries

The `dipf_Interpolation` enumaration consists of the following constants:

Name	Description
DIP_INTERPOLATION_DEFAULT	Default interpolation method
DIP_INTERPOLATION_BSPLINE	B-Spline interpolation
DIP_INTERPOLATION_FOURTH_ORDER_CUBIC	Forth order cubic interpolation
DIP_INTERPOLATION_THIRD_ORDER_CUBIC	Third order cubic interpolation
DIP_INTERPOLATION_LINEAR	Linear interpolation
DIP_INTERPOLATION_ZERO_ORDER_HOLD	Zero order hold interpolation

The dip_BackgroundValue enumaration consists of the following flags:

Name	Description
DIP_BGV_DEFAULT	Default: fill with zeros
DIP_BGV_ZERO	Fill with zeros
DIP_BGV_MAX_VALUE	Fill with maximum value for data type
DIP_BGV_MIN_VALUE	Fill with minimum value for data type

SEE ALSO

Rotation, Rotation3d, Rotation3d_Axis

SmallObjectsRemove

Remove small objects from an image

SYNOPSIS

```
#include "dip_measurement.h"
dip_Error dip_SmallObjectsRemove ( in, out, threshold )
```

DATA TYPES

integer

FUNCTION

This function removes from the labeled image `in` those objects whose size (measured in the number of pixels) is smaller than `threshold`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_int	threshold	Minimum object size

SEE ALSO

Measure, ObjectToMeasurement, Label

SobelGradient

A linear gradient filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_SobelGradient ( in, out, boundary, processDim )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

The SobelGradient filter computes a finite difference gradient (1 0 -1)/2 in the `processDim`, and performs a local (1 2 1)/4 smoothing in the other dimensions. Note that in 2D, this differs by a multiplication factor of 1/8 to the original definition by Sobel.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_int`	`processDim`	ProcessDim

SEE ALSO

General information about convolution

`FiniteDifference`, `Uniform`, `Gauss`, `SeparableConvolution`, `Convolve1d`, `Derivative`

Sort

Sort a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_Sort ( data, size, algorithm, dataType )
```

FUNCTION

Sorts a block of data (of size `size` and data type `dataType`) using the algorithm specified by `algorithm`.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_int	size	Size
dip_Sort	algorithm	Sort algorithm
dip_DataType	dataType	Data type. See DIPlib's data types

The `sortType` parameter is one of:

Name	Description
DIP_SORT_DEFAULT	Default sort algorithm
DIP_SORT_QUICK_SORT	Quick sort
DIP_SORT_DISTRIBUTION_SORT	Distribution sort
DIP_SORT_INSERTION_SORT	Insertion sort

SEE ALSO

General information about sorting

DistributionSort, InsertionSort, QuickSort, ImageSort, SortIndices, SortIndices16, ImageSortIndices

SortAnything

Sort data of any type

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_SortAnything ( data, size, compareFunction, swapFunction,
tmpData, algorithm )
```

FUNCTION

Sorts a block of data (of size `size`) using the algorithm specified by `algorithm`. This routine requires the user to write two functions in order to fully implement the sorting procedure. These are `SortCompareFunction` and `SortSwapFunction`.

ARGUMENTS

Data type	Name	Description
`void *`	`data`	Data
`dip_int`	`size`	Size
`dip_SortCompareFunction`	`compareFunction`	Function for comparing two data points
`dip_SortSwapFunction`	`swapFunction`	Function for swapping two data points, or copying one to the other
`void *`	`tmpData`	Pointer to a variable of the same type as the data, used as temporary space by some of the algorithms
`dip_Sort`	`algorithm`	Sort algorithm

SEE ALSO

General information about sorting

`QuickSortAnything`, `SortCompareFunction`, `SortSwapFunction`

SortCompareFunction

Typedef for comparison function (sorting)

SYNOPSIS

`#include "dip_sort.h"`

`dip_Boolean (*dip_SortCompareFunction) ( data1, index1, data2, index2 )`

FUNCTION

A function of this type must be supplied to the sorting algorithms for data of arbitrary type. It should return `DIP_TRUE` if data1[index1] > data2[index2].

Example:

```
dip_Boolean MyComplexCompare( void *data1, dip_int index1, void *data2, dip_int index2 )
{
   dip_complex *cmplx1, *cmplx2;
   dip_float magnitude1, magnitude2;

   cmplx1 = data1;
   cmplx2 = data2;
   cmplx1 += index1;
   cmplx2 += index2;
   magnitude1 = sqrt( cmplx1->re * cmplx1->re + cmplx1->im * cmplx1->im );
   magnitude2 = sqrt( cmplx2->re * cmplx2->re + cmplx2->im * cmplx2->im );
   if ( magnitude1 > magnitude2 )
   {
      return( DIP_TRUE );
   }
   else
   {
      return( DIP_FALSE );
   }
}
```

ARGUMENTS

Data type	Name	Description
void *	data1	Pointer to first data array
dip_int	index1	Index to element in first data array
void *	data2	Pointer to second data array
dip_int	index2	Index to element in second data array

SEE ALSO

General information about sorting

SortAnything, QuickSortAnything, SortSwapFunction

SortIndices

Sort indices to a block of data

SYNOPSIS

```
#include "dip_sort.h"
dip_Error dip_SortIndices ( data, indices, size, algorithm, dataType,
indexType )
```

FUNCTION

Sorts a list of indices rather than the data itself using the algorithm specified by `algorithm`.

ARGUMENTS

Data type	Name	Description
void *	data	Data
void *	indices	Indices
dip_int	size	Size
dip_Sort	algorithm	Sort algorithm
dip_DataType	dataType	Data type. See DIPlib's data types
dip_DataType	indexType	Data type of the index array. Must be either DIP_DT_SINT32 or DIP_DT_SINT16.

The `sortType` parameter is one of:

Name	Description
DIP_SORT_DEFAULT	Default sort algorithm
DIP_SORT_QUICK_SORT	Quick sort
DIP_SORT_DISTRIBUTION_SORT	Distribution sort
DIP_SORT_INSERTION_SORT	Insertion sort

SEE ALSO

General information about sorting

DistributionSort, InsertionSort, QuickSort, Sort, ImageSort, SortIndices16, ImageSortIndices

SortIndices16

Sort indices to a block of data

SYNOPSIS

`#include "dip_sort.h"`

`dip_Error dip_SortIndices16 ( data, indices, size, algorithm, dataType )`

FUNCTION

Sorts a list of (16 bit) indices rather than the data itself using the algorithm specified by `algorithm`.

ARGUMENTS

Data type	Name	Description
void *	data	Data
dip_sint16 *	indices	Indices
dip_int	size	Size
dip_Sort	algorithm	Sort algorithm
dip_DataType	dataType	Data type. See DIPlib's data types

The `sortType` parameter is one of:

Name	Description
DIP_SORT_DEFAULT	Default sort algorithm
DIP_SORT_QUICK_SORT	Quick sort
DIP_SORT_DISTRIBUTION_SORT	Distribution sort
DIP_SORT_INSERTION_SORT	Insertion sort

SEE ALSO

General information about sorting

`DistributionSort`, `InsertionSort`, `QuickSort`, `Sort`, `ImageSort`, `SortIndices`, `ImageSortIndices`

SortSwapFunction

Typedef for swap and copy function (sorting)

SYNOPSIS

`#include "dip_sort.h"`

`void (*dip_SortSwapFunction) ( data1, index1, data2, index2, copy )`

FUNCTION

A function of this type must be supplied to the sorting algorithms for data of arbitrary type. It should swap data1[index1] and data2[index2] if `copy` = `DIP_FALSE`, and copy data1[index1] to data2[index2] if `copy` = `DIP_TRUE`.

Example:

```
void dip_MyComplexSwap( void *data1, dip_int index1, void *data2, dip_int index2, dip_Boolean
{
   dip_complex *cmplx1, *cmplx2, tmpValue;

   cmplx1 = data1;
   cmplx2 = data2;
   cmplx1 += index1;
   cmplx2 += index2;
   if ( copy == DIP_TRUE )
   {
      cmplx2->re = cmplx1->re;
      cmplx2->im = cmplx1->im;
   }
   else
   {
      tmpValue.re = cmplx2->re;
      tmpValue.im = cmplx2->im;
      cmplx2->re = cmplx1->re;
      cmplx2->im = cmplx1->im;
      cmplx1->re = tmpValue.re;
      cmplx1->im = tmpValue.im;
   }
   return;
}
```

ARGUMENTS

Data type	Name	Description
void *	data1	Pointer to first data array
dip_int	index1	Index to element in first data array
void *	data2	Pointer to second data array
dip_int	index2	Index to element in second data array
dip_Boolean	copy	if DIP_FALSE, swap data. if DIP_TRUE copy data from data1 to data2

SEE ALSO

General information about sorting

SortAnything, QuickSortAnything, SortCompareFunction

Sqrt

arithmetic function

SYNOPSIS

`dip_Error dip_Sqrt ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the square root of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Exp, Exp2, Exp10, Ln, Log2, Log10

StablePixelHeapFree

Destroy heap structure

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_StablePixelHeapFree ( heap )
```

FUNCTION

Frees all data associated to `heap` and sets `heap` to 0.

ARGUMENTS

Data type	Name	Description
dip_StablePixelHeap *	heap	The heap structure

SEE ALSO

StablePixelHeapNew, PixelHeapNew, PixelQueueNew, StablePixelHeapPush, StablePixelHeapPop, StablePixelHeapIsEmpty

StablePixelHeapIsEmpty

Query heap

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_StablePixelHeapIsEmpty ( heap, result )
```

FUNCTION

Checks to see if there are any items on the heap. See StablePixelHeapNew for information on the heap data structure.

ARGUMENTS

Data type	Name	Description
dip_StablePixelHeap	heap	The heap structure
dip_Boolean *	result	Set to true if there are no items in the heap

SEE ALSO

StablePixelHeapNew, PixelHeapNew, PixelQueueNew, StablePixelHeapFree, StablePixelHeapPush, StablePixelHeapPop

StablePixelHeapNew

Create a new heap structure

SYNOPSIS

`#include "dip_pixelqueue.h"`

`dip_Error dip_StablePixelHeapNew ( heap, ndims, blocksize, order, resources )`

FUNCTION

This function allocates space for a new `dip_StablePixelHeap` structure. Memory allocated is tracked in `resources`. The heap is dimensioned to hold pixels from an `ndims`-dimensional image, and initially enough space is allocated for `blocksize` elements. The heap will be expanded as necessary when used.

The heap stores the coordinates, the value and the pointer to a pixel in an image. Note that the value does not need to equal the data pointed to by the pointer. `ndims` can be set to zero, in which case no coordinates are stored; this does not affect the function of the value and the pointer.

A heap is a priority queue data structure. Just like a queue, items can be added (pushed) and subtracted (popped). However, in the priority queue the item popped is always the higherst priority one: either the one with the highest-valued item (`order` is `DIP_GVSO_HIGH_FIRST`) or lowest-valued item (`order` is `DIP_GVSO_LOW_FIRST`). When various identically-valued items are stored on the heap, they will be extracted in the same order as they were insterted (FIFO - first in, first out). If this order is unimportant (such as for the `GrowRegionsWeighted` algorithm, use the more efficient `dip_PixelHeap` instead. See `PixelHeapNew` for information on the unstable heap structure.

ARGUMENTS

Data type	Name	Description
dip_StablePixelHeap *	heap	The newly allocated heap structure
dip_int	ndims	Image dimensionality
dip_int	blocksize	Size of each allocation block
dipf_GreyValueSortOrder	order	Determines the heap's sort order
dip_Resources	resources	Resources tracking structure. See ResourcesNew

The `dipf_GreyValueSortOrder` enumeration consists of the following values:

Name	Description
DIP_GVSO_HIGH_FIRST	Process the pixels from high grey-value to low grey-value.
DIP_GVSO_LOW_FIRST	Process the pixels from low grey-value to high grey-value.

IMPLEMENTATION

This data structure is implemented identically to `PixelHeapNew` (see that function's description for details), but an insertion order value is attached to each pixel pushed onto the heap. This is used to maintain stability.

SEE ALSO

`PixelHeapNew`, `PixelQueueNew`, `StablePixelHeapFree`, `StablePixelHeapPush`, `StablePixelHeapPop`, `StablePixelHeapIsEmpty`

StablePixelHeapPop

Pop item onto heap

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_StablePixelHeapPop ( heap, coords, pointer, value )
```

FUNCTION

Pops the next pixel from the heap. See StablePixelHeapNew for information on the heap data structure. `coords` is a pointer to an array of `dip_ints`, such as that obtained with `dip_IntegerArray->array`. It should have as many elements as the image dimensionality. If the stack was created with `ndims` set to 0, the `coords` pointer is ignored. `coords`, `pointer` and `value` can be `NULL` if you are not interested in either those values.

ARGUMENTS

Data type	Name	Description
dip_StablePixelHeap	heap	The heap structure
dip_int *	coords	Receives the coordinates of the popped item
void **	pointer	Receives the pointer of the popped item
dip_sfloat *	value	Receives the value of the popped item

SEE ALSO

StablePixelHeapNew, PixelHeapNew, PixelQueueNew, StablePixelHeapFree, StablePixelHeapPush, StablePixelHeapIsEmpty

StablePixelHeapPush

Push item onto heap

SYNOPSIS

```
#include "dip_pixelqueue.h"
dip_Error dip_StablePixelHeapPush ( heap, coords, pointer, value )
```

FUNCTION

Pushes a pixel onto the heap. See StablePixelHeapNew for information on the heap data structure. All 3 values `coords`, `pointer` and `value` are stored, except if the heap was created with `ndims` set to 0, in which case the `coords` pointer is ignored.

`coords` is a pointer to an array of `dip_int`s, such as that obtained with `dip_IntegerArray->array`. It should have as many elements as the image dimensionality. `pointer` is a pointer to any memory location, and `value` is the value to be used when sorting.

ARGUMENTS

Data type	Name	Description
dip_StablePixelHeap	heap	The heap structure
dip_int *	coords	Coordinates to be pushed
void *	pointer	Pointer to be pushed
dip_sfloat	value	Value to be pushed

SEE ALSO

StablePixelHeapNew, PixelHeapNew, PixelQueueNew, StablePixelHeapFree, StablePixelHeapPop, StablePixelHeapIsEmpty

StandardDeviation

statistics function

SYNOPSIS

`dip_Error dip_StandardDeviation ( in, mask, out, ps )`

DATA TYPES

binary, integer, float

FUNCTION

Calculates the standard deviation of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

StringAppend

Append a string to another

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringAppend ( str1, str2, cstr )
```

FUNCTION

Concatenates `str1` and `str2` and puts the result in `str1`, which is increased in size if necessary. If `str2` is `0`, `cstr` is used instead.

ARGUMENTS

Data type	Name	Description
dip_String	str1	First string
dip_String	str2	Second string
char *	cstr	Second string

SEE ALSO

StringCat, StringCompare, StringCompareCaseInsensitive, StringCopy, StringCrop, StringNew, StringReplace, UnderscoreSpaces

StringArrayCopy

Copy a string array

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringArrayCopy ( new, src, resources )
```

FUNCTION

This function copies the complete `src` string array to `new`.

ARGUMENTS

Data type	Name	Description
`dip_StringArray *`	`new`	Pointer to the destination `dip_StringArray` structure
`dip_StringArray`	`src`	Source string array
`dip_Resources`	`resources`	Resources tracking structure. See ResourcesNew

SEE ALSO

StringArrayNew, StringArrayFree

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

StringArrayFree

Array free function

SYNOPSIS

`dip_Error dip_StringArrayFree ( array )`

FUNCTION

This function frees *array, and sets array to zero.

ARGUMENTS

Data type	Name	Description
dip_ImageArray *	array	string array

SEE ALSO

StringArrayNew, StringArrayCopy

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

StringArrayNew

Allocate an array of strings

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringArrayNew ( array, size, stringSize, init, resources )
```

FUNCTION

This function allocates an array of strings. `size` specifies the size of the array, `stringSize` the size of the individual strings, which are allocated too. If `StringSize` is zero, and `init` is not, the strings in the array are initialised with `init` (in size and content). If both `stringSize` and `init` are zero, strings of zero length are created. (see StringNew).

ARGUMENTS

Data type	Name	Description
dip_StringArray *	array	Pointer to the array
dip_int	size	Size of the array
dip_int	stringSize	Size of the strings
char *	init	Initialisation string
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

StringArrayFree, StringArrayCopy

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

StringCat

Concatenate two strings

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringCat ( newStr, str1, str2, cstr, resources )
```

FUNCTION

Concatenates `str1` and `str2` and puts the result in `newStr`, which is allocated. If `str2` is `0`, `cstr` is used instead.

ARGUMENTS

Data type	Name	Description
dip_String *	newStr	Destination
dip_String	str1	First string
dip_String	str2	Second string
char *	cstr	Second string
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

StringAppend, StringCompare, StringCompareCaseInsensitive, StringCopy, StringCrop, StringNew, StringReplace, UnderscoreSpaces

StringCompare

Compare two strings

SYNOPSIS

`#include "dip_string.h"`

`dip_Error dip_StringCompare ( orig, copy, verdict )`

FUNCTION

This function uses the `strcmp` function to compare `orig` and `copy`. If the strings are different, an error is generated, or `verdict` obtains the value `DIP_FALSE`, if it is not zero.

ARGUMENTS

Data type	Name	Description
dip_String	orig	The original string
dip_String	copy	The fake (or not) string
dip_Boolean *	verdict	Verdict of the comparison

SEE ALSO

StringAppend, StringCat, StringCompareCaseInsensitive, StringCopy, StringCrop, StringNew, StringReplace, UnderscoreSpaces

StringCompareCaseInsensitive

Compare two strings without minding case

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringCompareCaseInsensitive ( orig, copy, verdict )
```

FUNCTION

This function uses the `strcasecmp` (or `stricmp`) function to compare `orig` and `copy`. If the strings are different, an error is generated, or `verdict` obtains the value `DIP_FALSE`, if it is not zero.

ARGUMENTS

Data type	Name	Description
dip_String	orig	The original string
dip_String	copy	The fake (or not) string
dip_Boolean *	verdict	Verdict of the comparison

SEE ALSO

StringAppend, StringCat, StringCompare, StringCopy, StringCrop, StringNew, StringReplace, UnderscoreSpaces

StringCopy

Copy a String

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringCopy ( new, src, resources )
```

FUNCTION

Thsi function copies string `src` to `new`.

ARGUMENTS

Data type	Name	Description
`dip_String *`	`new`	Pointer to a destination `dip_String` strcture
`dip_String`	`src`	Source string
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`StringAppend`, `StringCat`, `StringCompare`, `StringCompareCaseInsensitive`, `StringCrop`, `StringNew`, `StringReplace`, `UnderscoreSpaces`

StringCrop

Crop a string

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringCrop ( str, length )
```

FUNCTION

Crops `str` to `length` characters.

ARGUMENTS

Data type	Name	Description
dip_String	str	String to be cropped
dip_int	length	New string length

SEE ALSO

StringAppend, StringCat, StringCompare, StringCompareCaseInsensitive, StringCopy, StringNew, StringReplace, UnderscoreSpaces

StringFree

Free a string

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringFree ( void )
```

FUNCTION

This function frees a string data structure that has been allocated using StringNew.

ARGUMENTS

Data type	Name	Description
dip_String *	string	Pointer to the string to be freed

SEE ALSO

StringNew

StringNew

Allocate a string

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringNew ( string, size, init, resources )
```

FUNCTION

This function allocates a string of size `size`. If `init` is not zero, its contents is copied into the new string. If `size` is zero, and `init` is not, the size of `string` is made equal to `init` plus one.

ARGUMENTS

Data type	Name	Description
`dip_String *`	`string`	Pointer to the new string
`dip_int`	`size`	Size of the string
`char *`	`init`	Initialisation string
`dip_Resources`	`resources`	Resources tracking structure. See `ResourcesNew`

SEE ALSO

`StringArrayNew`, `StringAppend`, `StringCat`, `StringCompare`, `StringCompareCaseInsensitive`, `StringCopy`, `StringCrop`, `StringReplace`, `UnderscoreSpaces`

StringReplace

Replace the contents of one string with that of another

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_StringReplace ( str1, str2, cstr )
```

FUNCTION

Replaces the content of `str1` with `str2`. `str1` is increased in size if necessary. If `str2` is `0`, `cstr` is used instead.

ARGUMENTS

Data type	Name	Description
dip_String	str1	Destination string
dip_String	str2	Source string
char *	cstr	Source string

SEE ALSO

StringAppend, StringCat, StringCompare, StringCompareCaseInsensitive, StringCopy, StringCrop, StringNew, UnderscoreSpaces

StructureTensor2D

Two dimensional Structure Tensor

SYNOPSIS

`#include "dip_structure.h"`

`dip_Error dip_StructureTensor2D( in, mask, orientation, energy, l1, l2, anisotropy1, anisotropy2, boundary, gradSpec, gradSigmas, tensorSpec, tensorSigmas )`

DATA TYPES

integer,**float**

FUNCTION

This function computes the Structure Tensor (ST) at each point in the image. For a description of this technique see the references. There are two stages in the computation. The first stage computes the gradient vector at each point, using `Derivative` with parameters `gradSpec` and `gradSigmas`. The second stage, the tensor smoothing, is also performed using `Derivative` (with order = 0). The parameters used are `tensorSpec` and `tensorSigmas`.

If a mask image is given, a technique called normalised convolution (see references) is used to "fill in" the missing data.

The routine has a number of output images. Each of these can be set to zero. If set to zero, the corresponding result will not be computed. The following quantities are computed by this routine:

`orientation`	Orientation. Lies in the interval (-pi/2,pi/2).
`energy`	Sum of the two eigenvalues `l1` and `l2`.
`l1`	The largest eigenvalue.
`l2`	The smallest eigenvalue.
`anisotropy1`	Measure for local anisotropy: `( l1 - l2 ) / ( l1 + l2 )`.
`anisotropy2`	Measure for local anisotropy: `1 - l2 / l1`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Mask image (0=missing data)
dip_Image	orientation	Orientation
dip_Image	energy	Energy (l1+l2)
dip_Image	l1	Largest eigenvalue
dip_Image	l2	Smallest eigenvalue
dip_Image	anisotropy1	Local anisotropy: (l1-l2)/(l1+l2)
dip_Image	anisotropy2	Local anisotropy: 1-l2/l1
dip_BoundaryArray	boundary	Boundary conditions
dip_DerivativeSpec	gradSpec	Parameters for derivative to compute gradient (see DerivativeSpec data structure)
dip_FloatArray	gradSigmas	Sigmas of derivative to compute gradient
		Sigma of derivative to compute gradient
dip_DerivativeSpec	tensorSpec	Parameters for Gaussian for tensor smoothing (see DerivativeSpec data structure)
dip_FloatArray	tensorSigmas	Sigmas of Gaussian for tensor smoothing
		Sigma of Gaussian for tensor smoothing

LITERATURE

Bernd Jahne, *Practical Handbook on Image Processing for Scientific Applications*, chapter 13, CRC Press, 1997

L.J. van Vliet and P.W. Verbeek, *Estimators for Orientation and Anisotropy in Digitized Images*, in: J. van Katwijk, J.J. Gerbrands, M.R. van Steen, J.F.M. Tonino (eds.), ASCI'95, Proc. First Annual Conference of the Advanced School for Computing and Imaging (Heijen, NL, May 16-18), ASCI, Delft, 1995, pp. 442-450.

C.F. Westin, *A Tensor Framework for Multidimensional Signal Processing*, PhD thesis, Linkoping University, Sweden, 1994

SEE ALSO

Derivative

Sub

arithmetic function

SYNOPSIS

`dip_Error dip_Sub ( in1, in2, out )`

Calls `Arith ( in1, in2, out, DIP_ARITHOP_SUB, DIP_DT_MINIMUM )`

SubComplex

arithmetic function

SYNOPSIS

`dip_Error dip_SubComplex ( in, out, constant )`

DATA TYPES

binary, integer, float, **complex**

FUNCTION

This function computes `out = in - constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_complex	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, SubInteger, SubFloat, AddComplex, MulComplex, MulConjugateComplex, DivComplex

SubFloat

arithmetic function

SYNOPSIS

`dip_Error dip_SubFloat ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in - constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, SubInteger, SubComplex, AddFloat, MulFloat, DivFloat

SubInteger

arithmetic function

SYNOPSIS

`dip_Error dip_SubInteger ( in, out, constant )`

DATA TYPES

binary, **integer**, **float**, **complex**

FUNCTION

This function computes `out` = `in - constant` on a pixel by pixel basis. The data types of the `in1` image and `constant` may be of different types. See Information about dyadic operations for more information about what the type of the output will be.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_int	constant	Constant

SEE ALSO

Arith, Arith_ComplexSeparated, SubFloat, SubComplex, AddInteger, MulInteger, DivInteger

SubpixelLocation

Gets coordinates of an extremum with sub-pixel precision

SYNOPSIS

`#include "dip_analysis.h"`

`dip_Error dip_SubpixelLocation ( in, pos, coords, val, method, polarity )`

DATA TYPES

integer, float

FUNCTION

Determines the sub-pixel location of a local maximum or minimum close to `pos`. `pos` should point to a pixel that is larger than its direct neighbours (if `polarity` is `DIP_SEP_MAXIMUM`) or smaller than its direct neighbours (`polarity` is `DIP_SEP_MINIMUM`). `coords` will contain the the sub-pixel location of this local extremum. `val` will contain the interpolated grey value at the location of the extremum. `method` determines which method is used.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input grayscale image
dip_IntegerArray	pos	Input coordinates
dip_FloatArray	coords	Output coordinates
dip_float*	val	Output grey value
dipf_SubpixelExtremumMethod	method	Sub-pixel detection method
dipf_SubpixelExtremumPolarity	pol	Maximum or minimum?

The `dipf_SubpixelExtremumMethod` flag can be any of these values:

Name	Description
DIP_SEM_DEFAULT	Same as DIP_SEM_PARABOLIC_SEPARABLE.
DIP_SEM_LINEAR	Computes the center of gravity of 3 pixels around the extremum, in each dimension independently. The val returned is that of the pixel at pos.
DIP_SEM_PARABOLIC_SEPARABLE	Fits a parabola to 3 pixels around the extremum, for each dimension independently. The val returned is the maximum (minimum) value of these 1D extrema, and thus not equivalent to the grey value obtained by true interpolation.
DIP_SEM_PARABOLIC	Fits a parabolic patch to a region 3x3 or 3x3x3 pixels around the extremum (only for 2D or 3D images).
DIP_SEM_GAUSSIAN_SEPARABLE	Same as DIP_SEM_PARABOLIC_SEPARABLE, but using the log of the pixel values, very accurate if peak is a Gaussian.
DIP_SEM_GAUSSIAN	Same as DIP_SEM_PARABOLIC, but using the log of the pixel values (only for 2D or 3D images).
DIP_SEM_BSPLINE	Fits a B-spline to 11 pixels around the extremum, in each dimension independently. The val returned is the maximum (minimum) value of these 1D extrema, and thus not equivalent to the grey value obtained by true interpolation

SEE ALSO

SubpixelMaxima, SubpixelMinima

SubpixelMaxima

Gets coordinates of local maxima with sub-pixel precision

SYNOPSIS

```
#include "dip_analysis.h"
dip_Error dip_SubpixelMaxima ( in, mask, out_coord, out_val, method )
```

DATA TYPES

integer, **float**

FUNCTION

Detects local maxima in the image, and returns their coordinates, with sub-pixel precision, in the output image `out_coord`. Only pixels where `mask` is on will be examined. Local maxima are detected using `Maxima`, then their position is determined accurately using `SubpixelLocation`. `out_coord` will have `ndims` pixels along the first dimension (`ndims` being the number of dimensions in `in`), and as many pixels along the second dimension as there are local maxima in `in`. Thus, each row of the image `out_coord` contains the coordinates of one local maximum. `out_coord` is always `dip_float`. `out_val`, when not `0`, will contain the interpolated values of the image at the local maxima. `out_val` will have the same size and type as `out_coord`, except only one pixel along the first dimension.

A local maximum can not touch the edge of the image. That is, its integer location must be one pixel away from the edge.

See `SubpixelLocation` for the definition of the `method` parameter.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input grayscale image
`dip_Image`	`mask`	Binary mask for ROI processing
`dip_Image`	`out_coord`	Output coordinates, image will be N_dims x N_maxima
`dip_Image`	`out_val`	Output values, image will be 1 x N_maxima
`dipf_SubpixelExtremumMethod`	`method`	Sub-pixel detection method

SEE ALSO

`SubpixelMinima`, `SubpixelLocation`, `Maxima`

SubpixelMinima

Gets coordinates of local minima with sub-pixel precision

SYNOPSIS

`#include "dip_analysis.h"`

`dip_Error dip_SubpixelMinima ( in, mask, out_coord, out_val, method )`

DATA TYPES

integer, **float**

FUNCTION

Detects local minima in the image, and returns their coordinates, with sub-pixel precision, in the output image `out_coord`. Only pixels where `mask` is on will be examined. Local minima are detected using `Minima`, then their position is determined accurately using `SubpixelLocation`. `out_coord` will have `ndims` pixels along the first dimension (`ndims` being the number of dimensions in `in`), and as many pixels along the second dimension as there are local minima in `in`. Thus, each row of the image `out_coord` contains the coordinates of one local minimum. `out_coord` is always `dip_float`. `out_val`, when not `0`, will contain the interpolated values of the image at the local minima. `out_val` will have the same size and type as `out_coord`, except only one pixel along the first dimension.

A local minimum can not touch the edge of the image. That is, its integer location must be one pixel away from the edge.

See `SubpixelLocation` for the definition of the `method` parameter.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input grayscale image
`dip_Image`	`mask`	Binary mask for ROI processing
`dip_Image`	`out_coord`	Output coordinates, image will be N_dims x N_minima
`dip_Image`	`out_val`	Output values, image will be 1 x N_minima
`dipf_SubpixelExtremumMethod`	`method`	Sub-pixel detection method

SEE ALSO

SubpixelMaxima, SubpixelLocation, Minima

Subsampling

Interpolation function

SYNOPSIS

```
#include "dip_interpolation.h"
dip_Error dip_Subsampling ( in, out, sample )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

This function subsamples `in` by copying each `sample`th pixel to `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_IntegerArray	sample	Sample spacing

SEE ALSO

Resampling

Sum

statistics function

SYNOPSIS

`dip_Error dip_Sum ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the sum of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Mean, Variance, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

SumModulus

statistics function

SYNOPSIS

`dip_Error dip_SumModulus ( in, mask, out, ps )`

DATA TYPES

binary, integer, float, complex

FUNCTION

Calculates the sum of the modulus the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, Variance, StandardDeviation, MeanModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

Tan

trigonometric function

SYNOPSIS

`dip_Error dip_Tan ( in, out )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

Computes the tangent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Atan2, Sinh, Cosh, Tanh

Tanh

trigonometric function

SYNOPSIS

`dip_Error dip_Tanh ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the hyperbolic tangent of the input image values.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Sin, Cos, Tan, Asin, Acos, Atan, Atan2, Sinh, Cosh

TensorImageInverse

Invert tensor image

SYNOPSIS

dip_Error dip_TensorImageInverse (in, out)

DATA TYPES

float

FUNCTION

Inverts the NxN tensor image `in` (stored as an array with N*N elements) using LU decomposition.

ARGUMENTS

Data type	Name	Description
`dip_ImageArray`	`in`	Input
`dip_ImageArray`	`out`	Output

TestObjectAddNoise

TestObject generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_TestObjectAddNoise ( object, noisy, background, backvalue,
gaussianNoise, poissonNoise, snr, conversion, variance, random )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function adds a mixture of Gaussian and Poisson noise to a testobject at a specified signal-to-noise ratio. The SNR is defined as the average object energy divided by the average noise power.

ARGUMENTS

Data type	Name	Description
dip_Image	object	Input Object Image
dip_Image	noisy	Output Image
dip_Image	background	Background Image
dip_float	backvalue	Constant Background Value
dip_float	gaussianNoise	Relative Amount of Gaussian Noise
dip_float	poissonNoise	Relative Amount of Poisson Noise
dip_float	snr	Signal to Noise Ratio
dip_float *	conversion (0)	Pointer to the Poisson Conversion Factor
dip_float *	variance (0)	Pointer to the Gaussian Variance
dip_Random *	random	Pointer to a random value structure

SEE ALSO

TestObjectCreate, TestObjectModulate, TestObjectBlur

TestObjectBlur

TestObject generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_TestObjectBlur ( object, psf, convolved, xNyquist, testPSF )
```

DATA TYPES

binary, integer, **float**

FUNCTION

This function blurs a testobject with a Gaussian psf, with a two dimensional in focus diffraction limited incoherent PSF or with an user-supplied PSF. The `xNyquist` parameter specifies the oversampling factor of the incoherent PSF and Gaussian PSF. The sigma of the Gaussian PSF is equal to 0.9 * `xNyquist`.

ARGUMENTS

Data type	Name	Description
dip_Image	object	Input Object Image
dip_Image	psf	User supplied PSF
dip_Image	convolved	Output Image
dip_float	xNyquist	Oversampling Factor
dipf_TestPSF	testPSF	TestPSF

The `dipf_TestPSF` enumaration consists of the following flags:

Name	Description
DIP_TEST_PSF_GAUSSIAN	Gaussian PSF
DIP_TEST_PSF_INCOHERENT_OTF	in-focus, diffraction limited, incoherent PSF
DIP_TEST_PSF_USER_SUPPLIED	User supplied PSF with the `psf` image
DIP_TEST_PSF_NONE	no blurring

SEE ALSO

TestObjectCreate, TestObjectModulate, TestObjectAddNoise

TestObjectCreate

TestObject generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_TestObjectCreate ( object, testObject, objectHeight,
objectRadius, scale, scaleRadius, scaleAmplitude, objSigma, position, random
)
```

DATA TYPES

Output: sfloat

FUNCTION

This function can generate an aliasing free object (ellips, box, ellipsoid shell, box shell) or uses an user-supplied object. The generated objects have their origin at the center in the image, but can be generated with a sub-pixel random shift around the center, to average out dicretization effects over several instances of the same generated object. Optinally the generated object can be convolved with an isotropic Gaussian with a width specified by `objSigma`. Elliptical objects are only supported for images with a dimsnionality equal or less than three. The `position` boolean variable specifies whether a subpixel random shift should be applied to the object. This can be used to average out digitisation error over a repetition of the generation of the same object.

ARGUMENTS

Data type	**Name**	**Description**
dip_Image	object	Output Object Image
dipf_TestObject	testObject	Type of Test Object
dip_float	objectHeight	Object Height
dip_float	objectRadius	Object Radius
dip_FloatArray	scale	Relative Radii for each dimension
dip_float	scaleRadius	ScaleRadius
dip_float	scaleAmplitude	ScaleAmplitude
dip_float	objSigma	Sigma of Gaussian Object Blur
dip_Boolean	position	Random Subpixel Position Shift
dip_Random *	random	Pointer to a random value structure

SEE ALSO

`TestObjectModulate`, `TestObjectBlur`, `TestObjectAddNoise`

TestObjectModulate

TestObject generation function

SYNOPSIS

```
#include "dip_generation.h"
dip_Error dip_TestObjectModulate ( in, out, modulation, modulationDepth )
```

DATA TYPES

Output: sfloat

FUNCTION

This function adds a sine modulation to a test object, with `modulation` the modulation frequency and `modulationDepth` the modulation depth.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_FloatArray	modulation	Modulation Frequency
dip_float	modulationDepth	ModulationDepth

SEE ALSO

TestObjectCreate, TestObjectBlur, TestObjectAddNoise

Threshold

Point Operation

SYNOPSIS

```
#include "dip_point.h"
dip_Error dip_Threshold ( in, out, threshold, foreground, background,
binaryOutput )
```

DATA TYPES

integer, **float**

FUNCTION

This function thresholds an image at the `threshold` value. If the boolean `binaryOutput` is true, `Threshold` will produce a binary image. Otherwise an image of the same type as the input image is produced, with the pixels set to either `foreground` or `background`. In other words: `out` = (`in` >= threshold ? `foreground` : `background`)

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_float	threshold	Threshold value
dip_float	foreground	Foreground value
dip_float	background	Background value
dip_Boolean	binaryOutput	Convert output image to binary

SEE ALSO

See section 10.3, “Segmentation”, in Fundamentals of Image Processing.

`RangeThreshold`, `SelectValue`, `NotZero`, `Compare`, `HysteresisThreshold`, `IsodataThreshold`, `Clip`

TikhonovMiller

Image restoration filter

SYNOPSIS

```
#include "dip_restoration.h" #include "dip_transform.h"
dip_Error dip_TikhonovMiller ( in, psf, out, reg, background, method, var,
lambda, flags )
```

FUNCTION

The TikhonovMiller restoration filter is a linear least squares restoration algorithm.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Point spread function image
dip_Image	out	Output image
dip_Image	reg	Regularisation filter image
dip_Image	background (0)	Background image
dipf_RegularizationParameter	method	Method used to determine the regularisation parameter
dip_float	var	Noise variance
dip_float *	lambda	Regularisation parameter
dipf_ImageRestoration	flags	Restoration flags

LITERATURE

G.M.P. van Kempen, *Image Restoration in FLuorescence Microscopy*, Ph.D. Thesis, Delft University of Technology, 1999

SEE ALSO

Wiener, TikhonovRegularizationParameter

TikhonovRegularizationParameter

Determine the value of the regularisation parameter

SYNOPSIS

```
#include "dip_restoration.h"
dip_Error dip_TikhonovRegularizationParameter ( in, psf, reg, background,
max, min, lambda, method, var, flags )
```

FUNCTION

This function implements different methods to estimate the value of the regularistion parameter `lambda` of the TikhonovMiller restoration filter.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Point spread function image
dip_Image	reg	Regularisation filter rimage
dip_Image	background (0)	Background image
dip_float	max	Maximum value of `lambda`
dip_float	min	Minimum value of `lambda`
dip_float *	lambda	pointer to the regularisation parameter
dipf_RegularizationParameter	method	Method used to determine `lambda`
dip_float	var	Noise variance
dipf_ImageRestoration	flags	Restoration flags

LITERATURE

G.M.P. van Kempen, *Image Restoration in FLuorescence Microscopy*, Ph.D. Thesis, Delft University of Technology, 1999

SEE ALSO

TikhonovRegularizationParameter

TimerGet

Timing functions

SYNOPSIS

```
#include "dip_timer.h"
dip_Error dip_TimerGet ( timer )
```

FUNCTION

This function gets three timer values elapsed since the last call to TimerSet.

The dip_Timer struct contains the following values:

Data type	Name	Description
dip_int	setTime	Time stamp when TimerSet was called.
dip_int	getTime	Time stamp when TimerGet was called.
dip_float	getClockTime	Amount of CPU time (in seconds) between TimerSet and TimerGet.
dip_float	getSystemTime	Amount of CPU time (in seconds) executing system calls in the process.
dip_float	getUserTime	Amount of CPU time (in seconds) executing instructions in the process.
dip_float	setClockTime	Set by TimerSet, just ignore!
dip_float	setSystemTime	Set by TimerSet, just ignore!
dip_float	setUserTime	Set by TimerSet, just ignore!

setTime and getTime give the time, in seconds, elapsed since the Epoch (00:00:00 UTC, January 1, 1970). The C function ctime can convert this value into a date string.

getClockTime gives the CPU time, in seconds, between the call to TimerSet and TimerGet. The number of significant digits depends on your system. getUserTime contains the portion of this time that was used by the CPU to process instructions for the current process. getSystemTime contains the portion of time spent in the system while executing tasks on behalf of the current process (e.g. doing file I/O). getUserTime and getSystemTime do not necessarily add up to getClockTime if there are other processes running on the same processor.

ARGUMENTS

Data type	Name	Description
dip_Timer *	timer	Pointer to a dip_Timer struct

NOTES

Note that `getClockTime`, `getUserTime` and `getSystemTime` can wrap around. The system returns these values as a `clock_t` value. If this is a 32-bit integer, these timers wrap around after only 72 minutes.

`getUserTime` and `getSystemTime` are not supported on some systems, it is possible that these values are always 0.

SEE ALSO

`TimerSet`

TimerSet

Timing functions

SYNOPSIS

```
#include "dip_timer.h"
dip_Error dip_TimerSet ( timer )
```

FUNCTION

This function resets three timers that can be obtained by TimerGet.

ARGUMENTS

Data type	Name	Description
dip_Timer *	timer	pointer to a dip_Timer structure

SEE ALSO

TimerGet

Tophat

Morphological high-pass filter

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Tophat ( in, out, se, boundary, param, shape, edgeType,
polarity )
```

DATA TYPES

integer, **float**

FUNCTION

The top-hat is the difference between a morphological operation and the original image, comparable to a high-pass filter. Which operation is used can be chosen through the `dip_MphEdgeType` and `dip_MphTophatPolarity` parameters.

The rectangular, elliptic and diamond structuring elements are "flat", i.e. these structuring elements have a constant value. For these structuring elements, `param` determines the sizes of the structuring elements.

When `shape` is `DIP_FLT_SHAPE_DISCRETE_LINE` or `DIP_FLT_SHAPE_INTERPOLATED_LINE`, the structuring element is a line. `param->array[0]` determines the length, `param->array[1]` the angle. This is currently only supported for 2D images. Interpolated lines use interpolation to obtain a more accurate result, but loose the strict increasingness and extensivity (these properties are satisfied only by approximation).

When `shape` is set to `DIP_FLT_SHAPE_PARABOLIC`, `params` specifies the curvature of the parabola.

When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` is used as structuring element. It can be either a binary or a grey-value image. Its origin is the center, or one pixel to the left of the center if the size is even.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom structuring element
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Structuring element
dip_MphEdgeType	edgeType	edgeType
dip_MphTophatPolarity	polarity	polarity

The enumerator `dip_FilterShape` contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

The enumerator `dip_MphEdgeType` contains the following constants:

Name	Description
DIP_MPH_TEXTURE	Response is limited to edges in texture
DIP_MPH_OBJECT	Response is limited to object edges
DIP_MPH_BOTH	All edges produce equal response

The enumerator `dip_MphTophatPolarity` contains the following constants:

Name	Description
DIP_MPH_TEXTURE	Response is limited to edges in texture
DIP_MPH_OBJECT	Response is limited to object edges
DIP_MPH_BOTH	All edges produce equal response

SEE ALSO

`Lee`, `MorphologicalGradientMagnitude`, `MorphologicalRange`, `MultiScaleMorphologicalGradient`, `MorphologicalSmoothing`, `MorphologicalThreshold`

tpi.h

Type iterator

SYNOPSIS

`#include "dip_tpi.h"`

FUNCTION

Type iterator. For each data type specified by the define `DIP_TPI_ALLOW`, `dip_tpi.h` will include the file specified by the define `DIP_TPI_FILE`. If `DIP_TPI_ALLOW` is not defined the file will be included for all data types. `DIP_TPI_TYPES` must be defined as a logical OR of identifier flags and identifier group flags, as given in DIPlib's data types and the table below. During each "iteration" the main symbols defined by `dip_tpi.h` are `DIP_TPI`, `DIP_TPI_DATA_TYPE`, `DIP_TPI_IDENTIFIER` and `DIP_TPI_EXTENSION`. The following table shows how these are defined for each data type:

DIP_TPI	DIP_TPI_DATA_TYPE	DIP_TPI_IDENTIFIER	DIP_TPI_EXTENSION
dip_bin8	DIP_DT_BIN8	DIP_DTID_BIN8	_b8
dip_bin16	DIP_DT_BIN16	DIP_DTID_BIN16	_b16
dip_bin32	DIP_DT_BIN32	DIP_DTID_BIN32	_b32
dip_uint8	DIP_DT_UINT8	DIP_DTID_UINT8	_u8
dip_uint16	DIP_DT_UINT16	DIP_DTID_UINT16	_u16
dip_uint32	DIP_DT_UINT32	DIP_DTID_UINT32	_u32
dip_sint8	DIP_DT_SINT8	DIP_DTID_SINT8	_s8
dip_sint16	DIP_DT_SINT16	DIP_DTID_SINT16	_s16
dip_sint32	DIP_DT_SINT32	DIP_DTID_SINT32	_s32
dip_sfloat	DIP_DT_SFLOAT	DIP_DTID_SFLOAT	_sfl
dip_dfloat	DIP_DT_DFLOAT	DIP_DTID_DFLOAT	_dfl
dip_scomplex	DIP_DT_SCOMPLEX	DIP_DTID_SCOMPLEX	_scx
dip_dcomplex	DIP_DT_DCOMPLEX	DIP_DTID_DCOMPLEX	_dcx

Using this include file it is possible to compile source code for different data types. We recommend that instead of splitting your code into two files, one for generic code and one for type specific code, that you use `dip_tpi.h` to let the source file include itself. This also prevents dependency problems with makefiles. A source file that includes itself through `dip_tpi.h` should have the following format:

```
contents of example.c:
#ifndef DIP_TPI

#include "diplib.h"
```

```
#define DIP_TPI_FILE "example.c"
#include "dip_tpi.h"

/* This is where the generic code should be */

#else

/* This is where the type specific code should be */

#endif
```

In addition to the main defines as described above, there are a number of macro's that are defined by `dip_tpi.h`:

`DIP_TPI_FUNC` (function name)	attaches the current type suffix to the function name.
`DIP_TPI_DEFINE` (function name)	equivalent to: `dip_Error DIP_TPI_FUNC( function name )` useful for function definitions.
`DIP_TPI_DECLARE` (function name)	equivalent to: `dip_Error DIP_TPI_FUNC( function name )` useful for function declarations. Don't forget the trailing ";".
`DIP_TPI_NAME` (function name)	attaches the current type suffix to the function name and puts double quotes around the result, thus creating a string.

There are also a couple of defines that are only available for some of the data types:

When DIP_TPI is	
`dip_sfloat`	`DIP_TPI_CAST_R2C` is defined as `dip_scomplex`
`dip_dfloat`	`DIP_TPI_CAST_R2C` is defined as `dip_dcomplex`
`dip_scomplex`	`DIP_TPI_CAST_C2R` is defined as `dip_sfloat`
`dip_dcomplex`	`DIP_TPI_CAST_C2R` is defined as `dip_dfloat`

Other type iterators may be created by making a copy of the `dip_tpi.h` file and replacing `DIP_TPI` throughout the file by a different name for the new type iterator.

ARGUMENTS

Name	**Description**
`DIP_TPI_ALLOW`	logical OR of data type identifier and identifier group flags to indicate for which data types the file should be included
`DIP_TPI_FILE`	Name of the file to be included by `dip_tpi.h`

SEE ALSO

DIPlib's data types

`DataTypeGetInfo`, `ovl.h`

Truncate

Arithmetic function

SYNOPSIS

`dip_Error dip_Truncate ( in, out )`

DATA TYPES

binary, integer, **float**

FUNCTION

Computes the truncation of the input image values, and outputs a signed integer typed image.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output

SEE ALSO

Abs, Ceil, Floor, Sign, Fraction, NearestInt

UnderscoreSpaces

Replace spaces with underscores

SYNOPSIS

```
#include "dip_string.h"
dip_Error dip_UnderscoreSpaces ( string )
```

FUNCTION

This function replaces spaces in `string` with underscores. This function works in-place.

ARGUMENTS

Data type	Name	Description
dip_String	string	String to be examined

SEE ALSO

StringAppend, StringCat, StringCompare, StringCompareCaseInsensitive, StringCopy, StringCrop, StringNew, StringReplace

Uniform

Uniform filter

SYNOPSIS

```
#include "dip_linear.h"
dip_Error dip_Uniform ( in, out, se, boundary, param, shape )
```

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This functions implements an uniform convolution filter with support for various filter shapes.

Only the rectangular, elliptic and diamond filter shapes are supported (DIP_FLT_SHAPE_RECTANGULAR, DIP_FLT_SHAPE_ELLIPTIC and DIP_FLT_SHAPE_DIAMOND). Other filter shapes can be implemented by setting shape to DIP_FLT_SHAPE_STRUCTURING_ELEMENT, and passing a binary image in se. The "on" pixels define the shape of the filter window. Other values of shape are illegal.

If shape is not equal to DIP_FLT_SHAPE_STRUCTURING_ELEMENT, se can be set to zero. When shape is set to DIP_FLT_SHAPE_STRUCTURING_ELEMENT, param is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_Image	se	Custom filter shape (binary)
dip_BoundaryArray	boundary	Boundary conditions
dip_FloatArray	param	Filter parameters
dip_FilterShape	shape	Filter shape

The enumerator dip_FilterShape contains the following constants:

Name	Description
DIP_FLT_SHAPE_DEFAULT	Default filter window, same as DIP_FLT_SHAPE_RECTANGULAR
DIP_FLT_SHAPE_RECTANGULAR	Rectangular filter window, can be even in size
DIP_FLT_SHAPE_ELLIPTIC	Elliptic filter window, always odd in size
DIP_FLT_SHAPE_DIAMOND	Diamond-shaped filter window, always odd in size
DIP_FLT_SHAPE_PARABOLIC	Parabolic filter window (morphology only)
DIP_FLT_SHAPE_DISCRETE_LINE	Rotated line structuring element (morphology only)
DIP_FLT_SHAPE_INTERPOLATED_LINE	Rotated line structuring element, through interpolation (morphology only)
DIP_FLT_SHAPE_PERIODIC_LINE	(not implemented)
DIP_FLT_SHAPE_STRUCTURING_ELEMENT	Use `se` as filter window, can be any size

SEE ALSO

General information about convolution

`Gauss`, `GeneralConvolution`

UniformNoise

Generate an image disturbed by uniform noise

SYNOPSIS

`#include "dip_noise.h"`

`dip_Error dip_UniformNoise ( in, out, lowerBound, upperBound, random )`

DATA TYPES

integer, **float**

FUNCTION

Generate an image disturbed by additive uniform noise. See `UniformRandomVariable` for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	out	Output
dip_float	lowerBound	Lower bound of the uniform distribution the noise is drawn from
dip_float	upperBound	Upper bound of the uniform distribution the noise is drawn from
dip_Random *	random	Pointer to a random value structure

EXAMPLE

Get a image with additive uniform noise as follows:

```
dip_Image in, out;
dip_float lower, upper;
dip_Random random;

lower = 1.0;
upper = 10.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_UniformNoise( in, out, lower, upper, &random ));
```

SEE ALSO

UniformRandomVariable, RandomVariable, RandomSeed, RandomSeedVector, GaussianNoise, PoissonNoise, BinaryNoise

UniformRandomVariable

Uniform random variable generator

SYNOPSIS

#include "dip_noise.h"

dip_Error dip_UniformRandomVariable (random, lowerBound, upperBound, output)

FUNCTION

Generate an uniform distributed random variable. See RandomVariable for more information on the random number generator.

ARGUMENTS

Data type	Name	Description
dip_Random *	random	Pointer to a random value structure
dip_float	lowerBound	Lower bound of the uniform distribution the variable is drawn from
dip_float	upperBound	Upper bound of the uniform distribution the variable is drawn from
dip_float*	output	output

EXAMPLE

Get a uniform random variable as follows:

```
dip_Random random;
dip_float lower, upper, value;

lower = -1.0;
upper = 1.0;
DIPXJ( dip_RandomSeed( &random, 0 ));
DIPXJ( dip_UniformRandomVariable( &random, lower, upper,  &value ));
```

SEE ALSO

RandomVariable, RandomSeed, RandomSeedVector, GaussianRandomVariable, PoissonRandomVariable, BinaryRandomVariable

Unregister

Remove a registry item

SYNOPSIS

```
#include "dip_registry.h"
dip_Error dip_Unregister ( id, class )
```

FUNCTION

This function removes the Registry information of the `ID` of the Registry class `class`. See `Register` for more information about DIPlib's Registry.

ARGUMENTS

Data type	Name	Description
dip_int	id	Registry ID
dip_int	class	Registry class

SEE ALSO

`Register`, `RegistryList`, `RegistryGet`, `RegistryArrayNew`

UpperEnvelope

Upper envelope transform (a flooding and an algebraic closing)

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_UpperEnvelope ( in, out, bottom, labels, connectivity,
max_depth, max_size )
```

DATA TYPES

integer, **float**

FUNCTION

The Upper envelope transform produces a flooding of the input image (which is an algebraic closing). See any article by F. Meyer for further explanations.

The Upper envelope is based on the watershed transform, each region being filled up to the level where it meets a neighbouring region. See `Watershed` for information on the parameters.

The `bottom` image is a second output image that contains the whole watershed region painted with the lowest value in it. It is useful for stretching the input image: (`out - in`) / (`in - bottom`) . `labels` returns the label image used during region growing.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_Image`	`bottom`	Optional output
`dip_Image`	`labels`	Optional output
`dip_int`	`connectivity`	Connectivity
`dip_float`	`max_depth`	Maximum depth of a region that can be merged
`dip_int`	`max_size`	Maximum size of a region that can be merged

SEE ALSO

`Watershed`, `LocalMinima`

Variance

statistics function

SYNOPSIS

`dip_Error dip_Variance ( in, mask, out, ps )`

DATA TYPES

binary, integer, float

FUNCTION

Calculates the variance of the pixel values over all those dimensions which are specified by `ps`.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask (0)	Mask
dip_Image	out	Output
dip_BooleanArray	ps (0)	Dimensions to project

SEE ALSO

From images to scalars

Sum, Mean, StandardDeviation, MeanModulus, SumModulus, MeanSquareModulus, Maximum, Minimum, Median, Percentile

VarianceFilter

Sample Variance Filter

SYNOPSIS

```
#include "dip_filtering.h"
dip_Error dip_VarianceFilter ( in, out, se, boundary, param, shape )
```

DATA TYPES

binary, **integer**, **float**

FUNCTION

This function calculates for every pixel the sample variance of the pixels in the filter window (its size specified by `param`).

Only the rectangular, elliptic and diamond filter shapes are supported (`DIP_FLT_SHAPE_RECTANGULAR`, `DIP_FLT_SHAPE_ELLIPTIC` and `DIP_FLT_SHAPE_DIAMOND`). Other filter shapes can be implemented by setting `shape` to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, and passing a binary image in `se`. The "on" pixels define the shape of the filter window. Other values of `shape` are illegal.

If `shape` is not equal to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `se` can be set to zero. When `shape` is set to `DIP_FLT_SHAPE_STRUCTURING_ELEMENT`, `param` is ignored, and can be set to zero.

ARGUMENTS

Data type	Name	Description
`dip_Image`	`in`	Input
`dip_Image`	`out`	Output
`dip_Image`	`se`	Custom filter window (binary)
`dip_BoundaryArray`	`boundary`	Boundary conditions
`dip_FloatArray`	`param`	Filter sizes
`dip_FilterShape`	`shape`	Filter shape

The enumerator `dip_FilterShape` contains the following constants:

Name	**Description**
`DIP_FLT_SHAPE_DEFAULT`	Default filter window, same as `DIP_FLT_SHAPE_RECTANGULAR`
`DIP_FLT_SHAPE_RECTANGULAR`	Rectangular filter window, can be even in size
`DIP_FLT_SHAPE_ELLIPTIC`	Elliptic filter window, always odd in size
`DIP_FLT_SHAPE_DIAMOND`	Diamond-shaped filter window, always odd in size
`DIP_FLT_SHAPE_PARABOLIC`	Parabolic filter window (morphology only)
`DIP_FLT_SHAPE_DISCRETE_LINE`	Rotated line structuring element (morphology only)
`DIP_FLT_SHAPE_INTERPOLATED_LINE`	Rotated line structuring element, through interpolation (morphology only)
`DIP_FLT_SHAPE_PERIODIC_LINE`	(not implemented)
`DIP_FLT_SHAPE_STRUCTURING_ELEMENT`	Use `se` as filter window, can be any size

SEE ALSO

Kuwahara

VectorDistanceTransform

Euclidean vector distance transform

SYNOPSIS

```
#include "dip_distance.h"
```

```
dip_Error dip_VectorDistanceTransform ( in, outx, outy, outz, distance,
border, method )
```

DATA TYPES

binary

FUNCTION

This function produces the vector components of the Euclidean distance transform. These are stored in the output images, one for each dimension of the input image. See the EuclideanDistanceTransform for detailed information about the parameters.

To compute the Euclidean distance from the vector compoments produced by this function, one needs to multiply each componemt with the sampling distance, square the result, sum the results for all components and take the square root of the sum.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_ImageArray	out	Output images
dip_FloatArray	distance	Sampling distances
dip_Boolean	border	Image border type
dipf_DistanceTransform	method	Transform method

dipf_DistanceTransform defines the following distance transform types:

Name	Description
DIP_EDT_FAST	fastest, but most errors
DIP_EDT_TIES	slower, but fewer errors
DIP_EDT_TRUE	slow, uses lots of memory, but is “error free”
DIP_EDT_BRUTE_FORCE	gives a result from which errors are calculated for the other methods. This method is extremly slow and should only be used for testing purposes.

LITERATURE

See EuclideanDistanceTransform

KNOWN BUGS

See EuclideanDistanceTransform

AUTHOR

James C. Mullikin, adapted to DIPlib by Geert M.P. van Kempen

SEE ALSO

EuclideanDistanceTransform, GreyWeightedDistanceTransform

VoidPointerArrayCopy

Copy an array

SYNOPSIS

`dip_Error dip_VoidPointerArrayCopy ( dest, src, resources )`

FUNCTION

This function copies the void pointer array `src` to `dest`. The array `dest` is created by this function as well.

ARGUMENTS

Data type	Name	Description
dip_IntegerArray *	dest	Destination array
dip_IntegerArray	src	Source array
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

VoidPointerArrayNew, VoidPointerArrayFree, VoidPointerArrayCopy, VoidPointerArrayFind

IntegerArrayCopy, FloatArrayCopy, ComplexArrayCopy, DataTypeArrayCopy, BooleanArrayCopy, VoidPointerArrayCopy, StringArrayCopy

VoidPointerArrayFind

Find value in array

SYNOPSIS

`dip_Error dip_VoidPointerArrayFind ( array, value, index, found )`

FUNCTION

Finds a value in an array and "returns" its index in the array. If `found` is zero, `VoidPointerArrayFind` will produce an error if `value` is not found, otherwise `found` obtains the search result (`DIP_FALSE` if `value` is not found).

ARGUMENTS

Data type	Name	Description
`dip_VoidPointerArray`	`array`	Array to find value in
`void *`	`value`	Value to find
`dip_int *`	`index`	Index of the found value
`dip_VoidPointer *`	`found`	Value found or not

SEE ALSO

`VoidPointerArrayNew`, `VoidPointerArrayFree`, `VoidPointerArrayCopy`, `VoidPointerArrayFind`

`IntegerArrayFind`, `FloatArrayFind`, `ComplexArrayFind`, `DataTypeArrayFind`, `BooleanArrayFind`, `VoidPointerArrayFind`

VoidPointerArrayFree

Array free function

SYNOPSIS

`dip_Error dip_VoidPointerArrayFree ( array )`

FUNCTION

This function frees `*array`, and sets `array` to zero.

ARGUMENTS

Data type	Name	Description
dip_VoidPointerArray *	array	Array

SEE ALSO

BooleanArrayNew, BooleanArrayFree, BooleanArrayCopy, BooleanArrayFind

ArrayFree, IntegerArrayFree, FloatArrayFree, ComplexArrayFree, BoundaryArrayFree, FrameWorkProcessArrayFree, DataTypeArrayFree, ImageArrayFree, BooleanArrayFree, VoidPointerArrayFree, StringArrayFree, CoordinateArrayFree

VoidPointerArrayNew

Array allocation function

SYNOPSIS

`dip_Error dip_VoidPointerArrayNew ( array, size, resources )`

FUNCTION

This function allocates the `size` elements of a `dip_VoidPointerArrayNew` and sets the size of the array to `size`.

ARGUMENTS

Data type	Name	Description
dip_VoidPointerArray *	array	Array
dip_int	size	Size
dip_Resources	resources	Resources tracking structure. See ResourcesNew

SEE ALSO

VoidPointerArrayNew, VoidPointerArrayFree, VoidPointerArrayCopy, VoidPointerArrayFind

ArrayNew, IntegerArrayNew, FloatArrayNew, ComplexArrayNew, BoundaryArrayNew, FrameWorkProcessArrayNew, DataTypeArrayNew, ImageArrayNew, BooleanArrayNew, VoidPointerArrayNew, StringArrayNew, CoordinateArrayNew

Watershed

Morphological segmentation

SYNOPSIS

```
#include "dip_morphology.h"
dip_Error dip_Watershed ( in, mask, out, connectivity, max_depth, max_size,
binaryOutput )
```

DATA TYPES

integer, **float**

FUNCTION

Watershed segmentation with built-in region merging. `max_depth` and `max_size` control the merging procedure. Any region with `max_size` or less pixels **and** with `max_depth` grey-value difference or less will be merged to neighbouring regions when they touch (as opposed to build a watershed). `max_size` equal to 0 means that the size of the region is not tested when merging. The regions are grown according to the `connectivity` parameter. See The connectivity parameter for more information. The output is either a labelled image where the pixels belonging to a catchment basin are labelled, or a binary image where the watershed pixels are 1 and the rest is 0. This is controlled by `binaryOutput`.

If `mask` is not 0, only the pixels within `mask` will be considered. All the other pixels will be marked as watershed pixels.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input
dip_Image	mask	Mask
dip_Image	out	Output
dip_int	connectivity	Connectivity
dip_float	max_depth	Maximum depth of a region that can be merged
dip_int	max_size	Maximum size of a region that can be merged
dip_Boolean	binaryOutput	DIP_FALSE if the output should be a labelled image

SEE ALSO

SeededWatershed, UpperEnvelope, LocalMinima, GrowRegions

`WeightedAdd`

arithmetic function

SYNOPSIS

`dip_Error dip_WeightedAdd ( in1, in2, out, weight )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This function calculates `out` = `in1` + `weight` * `in2`;

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dip_float	weight	Weight

SEE ALSO

WeightedMul, WeightedSub, WeightedDiv

WeightedDiv

arithmetic function

SYNOPSIS

`dip_Error dip_WeightedDiv ( in1, in2, out, weight )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This function calculates `out` = `in1` / `weight` * `in2`; If (`weight` * `in2`) is zero, `out` will be set to zero as well.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dip_float	weight	Weight

SEE ALSO

WeightedAdd, WeightedMul, WeightedSub

WeightedMul

arithmetic function

SYNOPSIS

`dip_Error dip_WeightedMul ( in1, in2, out, weight )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This function calculates `out` = `in1` * `weight` * `in2`;

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dip_float	weight	Weight

SEE ALSO

WeightedAdd, WeightedSub, WeightedDiv

WeightedSub

arithmetic function

SYNOPSIS

`dip_Error dip_WeightedSub ( in1, in2, out, weight )`

DATA TYPES

binary, integer, **float**, **complex**

FUNCTION

This function calculates `out` = `in1 - weight` * `in2`;

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First input
dip_Image	in2	Second input
dip_Image	out	Output
dip_float	weight	Weight

SEE ALSO

WeightedAdd, WeightedMul, WeightedDiv

Wiener

Image Restoration Filter

SYNOPSIS

`#include "dip_restoration.h"`

`dip_Error dip_Wiener ( in, psf, signalPower, noisePower, out, flags )`

FUNCTION

This function performs an image restoration using the Wiener filter. The Wiener filter is the linear restoration filter that is optimal in mean square error sense.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	psf	Point spread function image
dip_Image	signalPower	SignalPower image
dip_Image	noisePower	NoisePower image
dip_Image	out	Output image
dipf_Restoration	flags	Restoration flags

LITERATURE

G.M.P. van Kempen, *Image Restoration in FLuorescence Microscopy*, Ph.D. Thesis, Delft University of Technology, 1999

SEE ALSO

PseudoInverse, TikhonovMiller

Wrap

Wrap an image

SYNOPSIS

```
#include "dip_manipulation.h"
dip_Error dip_Wrap ( in, out, wrap )
```

DATA TYPES

binary, integer, float, complex

FUNCTION

This function wraps the `in` image around its image borders. `wrap` specifies the number of pixels over which the image has to wrapped in each dimension.

ARGUMENTS

Data type	Name	Description
dip_Image	in	Input image
dip_Image	out	Output image
dip_IntegerArray	wrap	Wrap parametrs

SEE ALSO

Wrap, Crop, Shift

Xor

logic operation

SYNOPSIS

`dip_Error dip_Xor ( in1, in2, out )`

DATA TYPES

binary

FUNCTION

The function `Xor` performs the logic XOR operation between the corresponding pixels in `in1` and `in2`, and stores the result in `out`.

ARGUMENTS

Data type	Name	Description
dip_Image	in1	First binary input image
dip_Image	in2	Second binary input image
dip_Image	out	Output image

SEE ALSO

Arith, And, Or, Invert

Chapter 3

Assorted topics

3.1 Boundary conditions

Neighbourhood operations pose a problem. What happens when the neighbourhood operator operates near the border of the image and needs data from the area outside the image? The usual solution, also adopted by DIPlib, is to silently extend the image. There are various ways of extending the boundary. Below is a list of the possible methods. More details can be found in the user guide. Note that not all functions support all of these.

Name	**Description**
DIP_BC_SYM_MIRROR	Symmetric mirroring
DIP_BC_ASYM_MIRROR	Asymmetric mirroring
DIP_BC_PERIODIC	Periodic copying
DIP_BC_ASYM_PERIODIC	Asymmetric periodic copying
DIP_BC_ADD_ZEROS	Extending the image with zeros
DIP_BC_ADD_MAX_VALUE	Extending the image with +infinity
DIP_BC_ADD_MIN_VALUE	Extending the image with -infinity

SEE ALSO

BoundaryArrayNew, BoundaryArrayFree

FillBoundaryArray, SeparableFrameWork

3.2 Compression methods for image files

The dipio_Compression structure

The structure dipio_Compression specifies the compression method to use when writing an image file, and contains the following elements:

Data type	Name	Description
dipio_CompressionMethod	method	Compression method
dip_int	level	Compression parameter, dependent on method

dipio_CompressionMethod is an enum with the known compression methods. File formats typically only support one or a few of these, and most of these methods do not have a parameter to set, in which case level is ignored. If an unsupported compression method is selected, no compression is done. The dipio_CompressionMethod has the following values:

Name	Description
DIPIO_CMP_DEFAULT	Default compression method for the file format
DIPIO_CMP_NONE	No compression
DIPIO_CMP_GZIP	ZIP compression, using zlib. The level parameter is between 1 and 10, 1 being the faster, lesser compression and 10 being the slower, higher compression.
DIPIO_CMP_DEFLATE	Deflate (same as DIPIO_CMP_GZIP)
DIPIO_CMP_COMPRESS	Using UNIX's "compress" utility, which uses the LZW algorithm
DIPIO_CMP_LZW	LZW compression (same as DIPIO_CMP_COMPRESS)
DIPIO_CMP_JPEG	Lossy JPEG compression. The level parameter is between 1 and 100, higher numbers giving better quality output but larger files.
DIPIO_CMP_PACKBITS	PackBits
DIPIO_CMP_THUNDERSCAN	ThunderScan
DIPIO_CMP_NEXT	NeXT
DIPIO_CMP_CCITTRLE	CCITT RLE
DIPIO_CMP_CCITTRLEW	CCITT RLE/W
DIPIO_CMP_CCITTFAX3	CCITT Group 3
DIPIO_CMP_CCITTFAX4	CCITT Group 4

Thus only DIPIO_CMP_GZIP and DIPIO_CMP_JPEG currently have a level to set.

Supported compression methods for the various file formats

The TIFF file writer understand the methods DIPIO_CMP_NONE, DIPIO_CMP_DEFLATE, DIPIO_CMP_LZW, DIPIO_CMP_JPEG, DIPIO_CMP_PACKBITS, DIPIO_CMP_THUNDERSCAN, DIPIO_CMP_NEXT, DIPIO_CMP_CCITTRLE, DIPIO_CMP_CCITTRLEW, DIPIO_CMP_CCITTFAX3 and DIPIO_CMP_CCITTFAX4. It defaults to DIPIO_CMP_DEFLATE. The level parameter is currently not used.

The ICS file writer understands `DIPIO_CMP_NONE`, `DIPIO_CMP_GZIP` and `DIPIO_CMP_COMPRESS`, although `DIPIO_CMP_COMPRESS` is currently not implemented. It defaults to `DIPIO_CMP_GZIP`.

The GIF file writer only understands `DIPIO_CMP_LZW`. The compression method selected is simply ignored.

The JPEG file writer only understands `DIPIO_CMP_JPEG`. The compression method selected is simply ignored.

All other file writers do not compress, and simply ignore the compression method requested.

3.3 DerivativeSpec data structure

STRUCTURE

This structure is an aggregate of common parameters for derivative operators. Its current definition is:

```
typedef struct
{
   dip_DerivativeFlavour flavour;
   dip_float truncation;
} dip_DerivativeSpec;
```

The enumerator `flavour` parameter is one of:

Name	**Description**
DIP_DF_DEFAULT	Default derivative flavour (==DIP_DF_FIRGAUSS)
DIP_DF_FIRGAUSS	Gaussian family, FIR implementation, Gauss
DIP_DF_IIRGAUSS	Gaussian family, IIR implementation, GaussIIR
DIP_DF_FTGAUSS	Gaussian family, FT implementation, GaussFT
DIP_DF_FINITEDIFF	Finite difference implementation, FiniteDifferenceEx

SEE ALSO

StructureTensor2D, Derivative

3.4 DIPlib's data types

Pixel values are represented by different types, called data types. DIPlib supports the data types given in the following table:

data type	dip_DataType	**data type identifier**	**suffix**
dip_bin8	DIP_DT_BIN8	DIP_DTID_BIN8	_b8
dip_bin16	DIP_DT_BIN16	DIP_DTID_BIN16	_b16
dip_bin32	DIP_DT_BIN32	DIP_DTID_BIN32	_b32
dip_uint8	DIP_DT_UINT8	DIP_DTID_UINT8	_u8
dip_uint16	DIP_DT_UINT16	DIP_DTID_UINT16	_u16
dip_uint32	DIP_DT_UINT32	DIP_DTID_UINT32	_u32
dip_sint8	DIP_DT_SINT8	DIP_DTID_SINT8	_s8
dip_sint16	DIP_DT_SINT16	DIP_DTID_SINT16	_s16
dip_sint32	DIP_DT_SINT32	DIP_DTID_SINT32	_s32
dip_sfloat	DIP_DT_SFLOAT	DIP_DTID_SFLOAT	_sfl
dip_dfloat	DIP_DT_DFLOAT	DIP_DTID_DFLOAT	_dfl
dip_scomplex	DIP_DT_SCOMPLEX	DIP_DTID_SCOMPLEX	_scx
dip_dcomplex	DIP_DT_DCOMPLEX	DIP_DTID_DCOMPLEX	_dcx

The data types can be divided into five classes: the binary, unsigned integer, signed integer, floating point and complex classes. Different data types in the same class (e.g. `dip_uint8` and `dip_uint16`) provide a different range of values they can represent.

The complex data types are defines as follows:

```
typedef struct
{
   dip_sfloat re;
   dip_sfloat im;
} dip_scomplex;
```

```
typedef struct
{
   dip_dfloat re;
   dip_dfloat im;
} dip_dcomplex;
```

The binary data types are simply aliases for a set of corresponding unsigned integer types. The reason for having a separate typedef for the binary types is that they are not used like ordinary integers. Each bit of the integer can store one binary value. When manipulating binary data, care must be taken not to change any of the other bits of the integer used for storing it.

The `dip_DataType` enumeration is used to represent data types symbolically. It is used in `dip_Image`'s to indicate what the data type of the image is. Data type identifiers are used by the type iterator (see `tpi.h`) and overload schemes (see `ovl.h` and overload.h). Type suffixes are used to give type specific routines a unique name. Using a standard set of suffixes enables the type iterator and overload schemes to deal with these type specific routines. The `dip_DataType` enumeration, data type identifiers and suffixes can be found in the table above.

In addition to the data type identifiers for individual data types, there are also defines to represent an entire group. These are given in the following table:

Data type identifier group	**data types**
DIP_DTGID_UINT	unsigned integer
DIP_DTGID_UNSIGNED	unsigned integer
DIP_DTGID_SINT	signed integer
DIP_DTGID_INT	signed and unsigned integer
DIP_DTGID_INTEGER	signed and unsigned intege
DIP_DTGID_FLOAT	floating-point
DIP_DTGID_REAL	integer and floating-point
DIP_DTGID_COMPLEX	complex floating-point
DIP_DTGID_SIGNED	signed integer, floating-point and complex
DIP_DTGID_BINARY	binary
DIP_DTGID_ALL	all

SEE ALSO

DataTypeGetInfo

3.5 Description of DIPlib's pixel tables

Pixel tables provide an efficient way to encode a multi-dimensional binary object. DIPlib's dip_PixelTable implements this using runlength encoding (in 2-D this coding scheme is known as pxy-tables).

A DIPlib pixel table is a structure (defined in `dip_pixel_table.h`) that incorporates a link-list of runlengths. Each run-length consists of a n-D coordinate (integer array) and the length of the run along the X dimension. All the runlengths in total encode the binary object.

LITERATURE

See section 3.6, "Contour representations", in Fundamentals of Image Processing.

I.T. Young, R.L. Peverini, P.W. Verbeek and P.J. van Otterloo, *A New Implementation for Binary and Minkowski Operators*, Computer Graphics and Image Processing, Volume 17, No. 3, 189-210, 1981

3.6 File formats recognized by dipIO

The Registry

A number of file reading and writing functions are included in dipIO. These are registered in the `ImageReadRegistry` and the `ImageWriteRegistry`. Through this registry, `ImageRead` and `ImageWrite` are able to read from and write to any registered file format. You can add your own functions to these (the interface functions for this are not documented yet), thereby increasing the possibilities of `ImageRead` and `ImageWrite`.

Below you can find a list of currently supported file formats for both reading and writing. To obtain the format ID from the registry, you need to include the specified file and call the specified function.

Reading

These are the file formats currently supported for reading:

format	include file	registry ID retrieval function	dimension-ality	colour	data types
ICS (Image Cytometry Standard)	dipio_ics.h	dipio_ReadICSID	any	yes	any
TIFF (Tagged Image File Format)	dipio_tiff.h	dipio_ReadTIFFID	2D	yes	any
JPEG (JPEG File Interchange Format)	dipio_jpeg.h	dipio_ReadJPEGID	2D	yes	uint8
GIF (Graphics Interchange Format)	dipio_gif.h	dipio_ReadGIFID	2D	yes	uint8
LSM (Zeiss LSM file format)	dipio_lsm.h	dipio_ReadLSMID	1D - 4D	no	uint8, uint16 and sfloat
PIC (BioRad PIC file format)	dipio_pic.h	dipio_ReadPICID	2D and 3D	no	uint8
CVS (Comma Separated Values)	dipio_csv.h	dipio_ReadCSVID	2D	no	sfloat

Writing

These are the file formats currently supported for writing:

format	include file	registry ID retrieval function	dimension-ality	colour	data types
ICS v1 (Image Cytometry Standard)	`dipio_ics.h`	`dipio_WriteICSv1ID`	any	yes	any, binXX converted to uintXX
ICS v2 (Image Cytometry Standard)	`dipio_ics.h`	`dipio_WriteICSv2ID`	any	yes	any, binXX converted to uintXX
TIFF (Tagged Image File Format)	`dipio_tiff.h`	`dipio_WriteTIFFID`	2D	yes	any in grey-value, uint8 in colour
JPEG (JPEG File Interchange Format)	`dipio_jpeg.h`	`dipio_WriteJPEGID`	2D	yes	uint8
GIF (Graphics Interchange Format)	`dipio_gif.h`	`dipio_WriteGIFID`	2D	no	uint8
CVS (Comma Separated Values)	`dipio_csv.h`	`dipio_WriteCSVID`	2D	no	any except complex
FLD (AVS field file)	`dipio_fld.h`	`dipio_WriteFLDID`	any	no	any
PS (PostScript)	`dipio_ps.h`	`dipio_WritePSID`	2D	yes	uint8, others automatically converted
EPS (Encapulated PostScript)	`dipio_ps.h`	`dipio_WriteEPSID`	2D	yes	uint8, others automatically converted

3.7 From images to scalars

Within DIPlib all data, i.e. multi-dimensional data, such as images, and scalar, are all represented by the same object: the image. Scalars are stored as zero dimensional images. Examine, for example, the following code to compute the sum over all the grey values:

```
dip_Image img;
dip_Image value;
...
dip_Sum ( img, 0, value, 0 );
```

Which stores the sum over all the pixel values of `img` in the 0-D image `value`. We often want to directly manipulate scalars, in which case we need to extract the value. This can be accomplished easily with the `GetInteger`, `GetFloat` or the `GetComplex` functions:

```
dip_Image img;
dip_Image valueimg;
dip_float value;
...
dip_Sum ( img, 0, valueimg, 0 );
dip_GetFloat ( valueimg, &value, 0 );
printf ( "The sum is: %f\n", value );
```

3.8 General information about convolution

Convolution can be explained in just a few words: it is a local weighted average (the weights can be negative). This of course does not explain how to use it or what its properties are. For this we refer to the following sources:

Ian T. Young, Jan J. Gerbrands and Lucas J. van Vliet, Fundamentals of Image Processing.

Alan V. Oppenheim, Alan S. Willsky and I.T. Young, *"Signals and Systems"*, Prentice-Hall, 1983.

Anil K. Jain, *"Fundamentals of Digital Image Processing"*, Prentice-Hall, 1989.

"The Digital Signal Processing Handbook", Vijay K. Madisetti and Douglas B. Williams (eds), CRC Press + IEEE Press, 1998.

Kenneth R. Castleman, *"Digital Image Processing"*, Prentice-Hall, 1996.

3.9 General information about sorting

There are two kinds of sorting routines in DIPlib. The first sorts a one-dimensional array of data, the second sorts a set of indices to a one-dimensional array of data. The result of the sort routines can be summarised as follows:

Sort: `data[ i ] <= data[ i + 1 ]`

Sort indices: `data[ indices[ i ] ] <= data[ indices[ i + 1 ] ]`

Note that the number of indices does not have to be equal to the amount of pixels in the image, it may be either smaller or larger. The indices themselves should of course "point" to a valid pixel.

The sorting algorithms are described in the following reference:

Donald E. Knuth, *"The Art of Computer Programming, volume 3: Sorting and Searching"*, second edition, Addison-Wesley, 1998.

3.10 Information about dyadic operations

There are two types of dyadic operations. First there are operations such as `Add`, `Sub`, etc... which take two input images. The second category consists of functions such as `AddFloat`, `AddComplex` etc... The data type of the output image given the data types of the input images is given by the following table:

	dcomplex	scomplex	dfloat	sfloat	sint32	sint16
dcomplex	dcomplex	dcomplex	dcomplex	dcomplex	dcomplex	dcomplex
scomplex	dcomplex	scomplex	dcomplex	scomplex	scomplex	scomplex
dfloat	dcomplex	dcomplex	dfloat	dfloat	dfloat	dfloat
sfloat	dcomplex	scomplex	dfloat	sfloat	sfloat	sfloat
sint32	dcomplex	scomplex	dfloat	sfloat	sint32	sint32
sint16	dcomplex	scomplex	dfloat	sfloat	sint32	sint16
sint8	dcomplex	scomplex	dfloat	sfloat	sint32	sint16
uint32	dcomplex	scomplex	dfloat	sfloat	sint32	sint32
uint16	dcomplex	scomplex	dfloat	sfloat	sint32	sint16
uint8	dcomplex	scomplex	dfloat	sfloat	sint32	sint16
binary	dcomplex	scomplex	dfloat	sfloat	sint32	sint16

	sint8	uint32	uint16	uint8	binary	
dcomplex	dcomplex	dcomplex	dcomplex	dcomplex	dcomplex	
scomplex	scomplex	scomplex	scomplex	scomplex	scomplex	
dfloat	dfloat	dfloat	dfloat	dfloat	dfloat	
sfloat	sfloat	sfloat	sfloat	sfloat	sfloat	
sint32	sint32	sint32	sint32	sint32	sint32	
sint16	sint16	sint32	sint16	sint16	sint16	
sint8	sint8	sint32	sint16	sint8	sint8	
uint32	sint32	uint32	uint32	uint32	uint32	
uint16	sint16	uint32	uint16	uint16	uint16	
uint8	sint8	uint32	uint16	uint8	uint8	
binary	sint8	uint32	uint16	uint8	sint8	

The output data type of an operation involving an image and a constant of one of the types: `dip_complex`, `dip_float`, `dip_int`, is given by the following table:

	dip_complex	dip_float	dip_int
dcomplex	dcomplex	dcomplex	dcomplex
scomplex	scomplex	scomplex	scomplex
dfloat	dcomplex	dfloat	dfloat
sfloat	scomplex	sfloat	sfloat
sint32	scomplex	sint32	sint32
sint16	scomplex	sint16	sint16
sint8	scomplex	sint8	sint8
uint32	scomplex	uint32	uint32
uint16	scomplex	uint16	uint16
uint8	scomplex	uint8	uint8
binary	scomplex	sint8	sint8

3.11 The image structure

DESCRIPTION

dip_Image is the structure that is used to store images in DIPlib. It contains a number of fields that are used to describe an image. The type field stores the type of the image using a dip_ImageType enumeration. Currently scalar images are the only supported type (DIP_IMTP_SCALAR). The DIP_IMTP_ALIEN type is used internally by DIPlib for creating interfaces to other packages. Whether the other fields in the dip_Image are meaningful depends on the image type. A dip_Image may contain fields specific to the current image type. These will be discussed on the pages pertaining to the type in question. The standard fields that are always present are:

field type	short description	access functions
dip_ImageType	The image type	ImageGetType, ImageSetType
dip_ImageState	The image state	(none)
dip_DataType	Data type used to store pixel values	ImageGetDataType, ImageSetDataType
dip_IntegerArray	Dimensions of the image	ImageGetDimensions, ImageGetDimensionality, ImageSetDimensions
void *	Pointer to the pixel data	ImageGetData
dip_int	Plane number, for binary images	ImageGetPlane
dip_IntegerArray	Stride array (see below)	ImageGetStride

Pixel values are stored in the data type specified by the data type field. For a list of possible data types see DIPlib's data types.

The dimensionality of the image and the size of each individual dimension is stored in the dimensions Array.

The data pointer points to the pixel at the origin of the image. For each dimension the stride array holds the interleave between two neigbouring pixels in memory. The following equation may be used to compute the address of a pixel at a coordinate specified by an array called cor[]:

```
                    (N-1)
address = origin +   Sum   cor[i] * stride[i]
                     i=0
```

A dip_Image structure does not necessarily have pixel data associated with it. When a dip_Image does not contain pixel data, it is said to be in the "raw" state. A dip_Image that does contain data, is said to be "forged". For binary images the plane field holds the number of the bit in which the binary data is stored. Access to the fields of a dip_Image is restricted to a number of functions, which are given in the table above. The “set” functions can only be used on “raw” images.

SEE ALSO

DIPlib’s data types

`ImageNew`

3.12 The connectivity parameter

DIPlib uses a different name for the various possible connectivites than you might be used to. This is to generalize this parameter to images of any dimensionality. It is defined as follows: if `connectivity` is 1 all pixels for which only one coordinate differs from the pixel's coordinates by maximally 1 are considered neighbours; if it is 2, all pixels for which one or two coordinates differ maximally 1 are considered neighbours. The connectivity can never be larger than the image dimensionality.

In terms of the obsolete connectivity definitions we have:

In 2-D	**this connectivity**	**corresponds to**	**and forms this structuring element**
	1	4 connectivity	diamond
	2	8 connectivity	square
	-1	4-8 connectivity	octagon
	-2	8-4 connectivity	octagon
In 3-D	**this connectivity**	**corresponds to**	**and forms this structuring element**
	1	6 connectivity	octahedron
	2	18 connectivity	cuboctahedron
	3	26 connectivity	cube
	-1	6-26 connectivity	small rhombicuboctahedron
	-3	26-6 connectivity	small rhombicuboctahedron

The negative connectivities are only defined for the functions in binary morphology such as `BinaryDilation` and `BinaryErosion`. These alternate steps with different connectivity to produce a better approximation to an isotropic structuring element.