Users Guide - Main NeatVision 2.1 Methods

Vision Systems Laboratory, Dublin City University
neatvision.eeng.dcu.ie

The following list summarises some of the main NeatVision methods users may wish to interface too. Many of these are fairly self-explanatory, but if the method you require is not listed or does not have enough information to enable you to use it drop us an email at *tech@neatvision.com* with "NeatVision Methods" in the subject line. Additional help can be found in the input/output tags for each block in the NeatVision visual programming interface.. Also refer to *P.F. Whelan and D. Molloy (2000), **Machine Vision Algorithms in Java: Techniques and Implementation**, Springer (London), 298 Pages [ISBN 1-85233-218-2]* for additional details.

Normalization of greyscale image operations occurs to keep the output image within greyscale range 0-255.

Method	Description	Inputs (Index #: data type [descriptor])	Outputs (Index #: data type [descriptor])
DATA	Image, Integer, Double, Boolean, String, Array (of integers) and 3D (DICOM, Analyze, Vol, Sequence)		
FLOW CONTROL	SplitterX2,SplitterX3, SplitterX4, Feedback, If, Else, For and Terminate		
UTILITIES			
HalveImageSize	A grey-scale image whose size is halved	0:GrayImage	0:GrayImage
DoubleImageSize	A grey-scale image whose size is doubled	0:GrayImage	0:GrayImage
PointToSquare	A grey-scale image whose white pixels are represented by white squares.	0:GrayImage	0:GrayImage
PointToCross	A grey-scale image whose white pixels are represented by white crosses.	0:GrayImage	0:GrayImage
Rotate	A grey-scale image is rotated in a clockwise direction by a user specified amount	0:GrayImage 1: Integer [user specified rotation (degrees)]	0:GrayImage
RotatePlus90	A grey-scale image is rotated in a clockwise direction by 90 degrees	0:GrayImage	0:GrayImage

RotateMinus90	A grey-scale image is rotated in an anticlockwise direction by 90 degrees.	0:GrayImage	0:GrayImage
ROI[1]	A grey-scale image from which a rectangular region of interest is extracted by the user via the GUI.	0:GrayImage	0:GrayImage
PolyROI[2]	A grey-scale image from which a polygon region of interest is extracted by the user via the GUI.	0:GrayImage [User interaction]	0:GrayImage
EnhancePolyROI[2]	A grey-scale image from which a polygon region of interest shall be emphasised. User defined input region.	0:GrayImage [User interaction]	0:GrayImage
Measure_Line	An image from which the Euclidean distance between two user-selected points is calculated. Must rerun programme to generate new line length.	0:GrayImage [User interaction]	0:Double [Euclidean distance]
Scale	A grey-scale image is scaled by user defined dimensions	0:GrayImage 1: Integer [width of the scaled image] 2: Integer [height of the scaled image]	0:GrayImage
Mask	A grey-scale image whose border is masked by a user specified amount.	0:GrayImage 1: Integer [Mask size in pixels, Default =3]	0:GrayImage
Centroid	Replace the greyscale shapes (Range 0-255) in the original image by their respective centroids (commonly used after the 8-bit labelling operators)	0:GrayImage	0:GrayImage [Binary]
Centroid_16	Replace the greyscale shapes (Range 0-65535) in the original image by their respective centroids (commonly used after the Label_16 operators)	0:GrayImage	0:GrayImage [Binary]
BinaryToGreyscale	Convert WHITE pixels in a binary image to a given greyscale.	0:GrayImage [Binary] 1:Integer [greyscale (0-255)]	0:GrayImage
GreyScalePixelSum	Generates an integer which is the sum of all pixels contained in the input image	0:GrayImage	0:Integer
FirstWhitePixelLocator	Coordinate point representing the location of the first white pixel in the image input image.	0:GrayImage	0:Coordinate

[1] **Left click and hold** to draw the ROI, then release when complete.

[2] The user inputs a polygon by **left-clicking** a series of points (marked in red). When the user clicks a point within 4 pixels of the start point or alternatively **right-click** to finalize and close the polygon. Once closed the polygon will be displayed in green. To begin a new polygon use **shift-click**.

RemoveIsolatedWhitePixels	Remove isolate white pixels (3x3) region)	0:GrayImage	0:GrayImage
SaltnPepperGenerator	Add salt and pepper noise to the input image	0:GrayImage 1:Double (0-1.0)	0:GrayImage
AdditiveWhiteNoiseGenerator	Add a user defined level of white noise to the input image	0:GrayImage 1:Integer (0-1024)	0:GrayImage
GaussianNoiseGenerator	Add a user defined quantity of Gaussian noise to the input image	0:GrayImage 1:Double (0-255.0)	0:GrayImage
RayleighNoiseGenerator	Add a user defined quantity of Rayleigh noise to the input image	0:GrayImage 1:Double (1.0-255.0)	0:GrayImage
PoissonNoiseGenerator	Add a user defined quantity of Poisson noise to the input image	0:GrayImage 1:Double (0-511.0)	0:GrayImage
HTTPSendScript	Send arguments to a URL	0:String [URL] 1:String [Arguments]	0:String [Return values]
HTTPGetImage	Retrieve image from a URL	0:String [URL] 1:String [Arguments]	0:GrayImage [Retrieved Image]
ARITHMETIC			
Add	Image addition	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = A+B]
Subtract	Image subtraction	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = A-B]
Multiply	Image multiply	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = A*B]
Divide	Image division	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = A/B]
And	Boolean AND operation	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = AND(A,B)]
Or	Boolean OR operation	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = OR(A,B)]
Not	Boolean NOT operation	0:GrayImage [A]	0:GrayImage [C = NOT(A)]
Xor	Boolean Exclusive OR operation	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = XOR(A,B)]
Minimum	Minimum of two images	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = Min(A,B)]
Maximum	Maximum of two images	0:GrayImage [A] 1:GrayImage [B]	0:GrayImage [C = Max(A,B)]
HISTOGRAM			
HighestGreyLevelCalculator	Compute the highest grey level from the input image	0:GrayImage	0:Integer [highest grey level]
LowestGreyLevelCalculator	Compute the lowest grey level from the input image	0:GrayImage	0:Integer [lowest grey level]

MeanSquareError	Compare the input images using the mean square error operation	0:GrayImage 1:GrayImage	0:Double [mean square error]
AverageIntensityCalculator	Compute the average intensity of the input image	0:GrayImage	0:Double [average intensity]
EntropyCalculator	Compute the entropy of the input image	0:GrayImage	0:Double [entropy]
VarienceCalculator	Compute the variance of the input image	0:GrayImage	0:Double [varience]
KurtosisCalculator	Compute the kurtosis of the input image	0:GrayImage	0:Double [kurtosis]
StandardDeviationCalculator	Compute the standard deviation of the input image	0:GrayImage	0:Double [standard deviation]
SkewnessCalculator	Compute the skewness deviation of the input image	0:GrayImage	0:Double [skewness]
LocalEqualisation3x3	Local histogram equalisation using a 3x3 region	0:GrayImage	0:GrayImage
LocalEqualisation5x5	Local histogram equalisation using a 5x5 region	0:GrayImage	0:GrayImage
PROCESSING			
Inverse	Inverse the LUT of the input image	0:GrayImage	0:GrayImage
Logarithm	Transform the linear LUT into logarithmic	0:GrayImage	0:GrayImage
Exponential	Transform the linear LUT into exponential	0:GrayImage	0:GrayImage
Power	The linear LUT is raised to a user specified double value	0:GrayImage 1:Integer [power, default=3.0]	0:GrayImage
Square	The linear LUT is raised to power of 2.	0:GrayImage	0:GrayImage
SingleThreshold	Single threshold operation	0:GrayImage 1:Integer [(1-255): Default = 128]	0:GrayImage [Binary]
MidlevelThreshold	Single threshold operation: threshold level = MIDGREY (127)	0:GrayImage	0:GrayImage [Binary]
DualThreshold	Dual threshold operation. All pixels between the upper and lower thresholds are marked in WHITE.	0:GrayImage 1:Integer [upper value, default =128] 2:Integer [lower value, default =1]	0:GrayImage [Binary]
TripleThreshold	This operation produces an LUT in which all pixels below the user specified lower level appear black, all pixels between the user specified lower level and the user specified upper level inclusively appear grey and all pixels above the user specified upper level appear white.	0:GrayImage 1:Integer [upper value, default =128] 2:Integer [lower value, default =1]	0:GrayImage

EntropicThreshold	Compute the entropy based threshold. Relies on maximising the total entropy of both the object and background regions to find the appropriate threshold	0:GrayImage	0:Integer
Threshold3x3	Adaptive threshold in a 3x3 region.	0:GrayImage 1:Integer [constant offset, default=0]]	0:GrayImage
Threshold5x5	Adaptive threshold in a 5x5 region.	0:GrayImage 1:Integer [constant offset, default=0]]	0:GrayImage
IntensityRangeEnhancer	Stretch the LUT in order to occupy the entire range between BLACK (0) and WHITE (255)	0:GrayImage	0:GrayImage
HistorgramEqualiser	Histogram equalisation	0:GrayImage	0:GrayImage
IntensityRangeStrecher	Stretch the LUT between the lower and upper threshold to occupy the entire range between BLACK (0) and WHITE (255)	0:GrayImage 1:Integer [lower grey level, default=0] 2:Integer [upper grey level, default=255]	0:GrayImage
IntegrateImageRows	Integrate image rows	0:GrayImage	0:GrayImage
IntegrateImageColumns	Integrate Image columns	0:GrayImage	0:GrayImage
LeftToRightSum	Pixel summation along the line	0:GrayImage	0:GrayImage
LeftToRightWashFunction	Left To Right wash function (once a white pixel is found, all pixels to its right are also set to white)	0:GrayImage	0:GrayImage
RightToLeftWashFunction	Right To Left wash function (once a white pixel is found, all pixels to its left are also set to white)	0:GrayImage	0:GrayImage
TopToBottomWashFunction	Top To Bottom wash function (once a white pixel is found, all pixels to its below are also set to white)	0:GrayImage	0:GrayImage
BottomToTopWashFunction	Bottom To Top wash function (once a white pixel is found, all pixels to its above are also set to white)	0:GrayImage	0:GrayImage
FILTER			
Convolution	Convolution. This operation requires coefficients to be specified in the form of a square, odd sized integer array, “null” represents “don’t cares”. See Appendix A.2 for an example.	0:GrayImage 1:Integer [] [Array of mask values. No entry default to null. “Don’t Care” = null statement]	0:GrayImage

DOLPS	DOLPS – Difference of low pass 3x3 filters. Image A results from applying 3 iterations of the low pass filter. Image B results from applying 6 iterations of the low pass filter. DOLPS = A-B.	0:GrayImage	0:GrayImage
LowPass	Low pass 3x3 filter	0:GrayImage	0:GrayImage
Sharpen	High pass 3x3 filter	0:GrayImage	0:GrayImage
Median	Median 3x3 filter	0:GrayImage	0:GrayImage
Midpoint	Midpoint 3x3 filter	0:GrayImage	0:GrayImage
RectangularAverageFilter	Rectangular Average Filter operation. Size of filter is user defined	0:GrayImage 1:Integer [filter size, default = 5]	0:GrayImage
SmallestIntensityNeighbour	Replace the central pixel of the 3x3 mask with the minimum value	0:GrayImage	0:GrayImage
LargestIntensityNeighbour	Replace the central pixel of the 3x3 mask with the maximum value	0:GrayImage	0:GrayImage
AdaptiveSmooth	Adaptive smoothing of grey scale images. In order to apply it to colour images, the input image has to be split into RGB components and adaptive smooth has to be applied to each channel. If the colour image is applied directly the algorithm will smooth the average intensity image. (Slow process)	0:GrayImage 1:Integer [number of iterations: possible values: 1 to 10, default = 5] 2:Double [variance strength: possible values: 0.1 -> 0.9, default = 0.2] 3:Double [Diffusion parameter: possible values: 1.0 -> 20.0, default = 10.0]	0:GrayImage
EDGES			
Roberts	Roberts edge detector	0:GrayImage	0:GrayImage
Sobel	Sobel edge detector	0:GrayImage	0:GrayImage
Laplacian	Laplacian edge detector. User defined 4-connected or 8-connected neighbourhood	0:GrayImage 1:Integer [possible values: 4 or 8, default = 8]	0:GrayImage
Prewitt	Prewitt edge detector	0:GrayImage	0:GrayImage
FreiChen	FreiChen edge detector	0:GrayImage	0:GrayImage
BinaryBorder	Binary Border edge detector	0:GrayImage [Binary]	0:GrayImage [Binary]
NonMaxima	Edge detection using non maxima suppression	0:GrayImage	0:GrayImage
IntensityGradientDirection	Compute the 3x3 intensity gradient direction. Gradients range from 1 to 8.	0:GrayImage	0:GrayImage [pixel values from 1-8]
ZeroCrossingsDetector	Zero crossings edge detector	0:GrayImage	0:GrayImage

Canny	Canny edge detector	0:GrayImage 1:Double [standard deviation or spread parameter, possible values: 0.2 -> 20.0, default = 1.0] 2:Integer [lower threshold, default = 1] 3:Integer [upper threshold, default = 255]	0:GrayImage [edge magnitudes] 1:GrayImage [edge directions]
EdgeLabel	Edge labelling operation. Expects a binary image resulting from the application of the Canny edge detector.	0:GrayImage 1:Boolean [Set True if you want closed structures]	0:GrayImage [A binary image whose edge pixels are grouped into polygonal shapes]
LineFitting	Line fitting in the edge structure. Expects a binary image resulting from the application of the Canny edge detector.	0:GrayImage 1:Boolean [Set True if you want closed structures]	0:GrayImage [A binary image whose edge pixels are grouped into polygonal shapes]
ArcFitting	Arc fitting in the edge structure. Expects a binary image resulting from the application of the Canny edge detector.	0:GrayImage 1:Boolean [Set True if you want closed structures] 2:Boolean [Set True if you want display the circles associated with detected arcs] 3:Boolean [Set True if you want display the lines that are not grouped into arcs segments]	0:GrayImage [A binary image whose edge pixels are grouped into polygonal shapes]
EdgeLinking[3]	Edge linking (scanning window is user defined). Expects a binary image resulting from the application of the Canny edge detector.	0:GrayImage 1:Integer [The size of scanning window. (5-11)]	0:GrayImage [Edge linked image]
ANALYSIS			
ThinOnce	Full application of the thinning algorithm. Thin *till completion* resulting in a skeleton image.	0:GrayImage [Binary]	0:GrayImage [Binary]
Thin	The input binary image is thinned N times as specified by the user	0:GrayImage [Binary] 1:Integer [N – number of iterations]	0:GrayImage [Binary]
CornerPointDetector	Skeleton corner detection from a binary image based on a 3x3 region	0:GrayImage [Binary]	0:GrayImage [Binary]
JunctionDetector	Skeleton junction detection from a binary image based on a 3x3 region	0:GrayImage [Binary]	0:GrayImage [Binary]

[3] O. Ghita and P.F. Whelan (2002), “A computationally efficient method for edge thinning and linking using endpoints”, **Journal of Electronic Imaging**, 11(4), Oct. 2002, pp 479-485.

LimbEndDetector	Skeleton limb end detection from a binary image based on a 3x3 region	0:GrayImage [Binary]	0:GrayImage [Binary]
BiggestBlob	Extract the biggest white blob from a binary image	0:GrayImage [Binary]	0:GrayImage [Binary]
SmallestBlob	Extract the smallest white blob from a binary image	0:GrayImage [Binary]	0:GrayImage [Binary]
BlobFill	Fill the holes in a binary image	0:GrayImage [Binary]	0:GrayImage [Binary]
Labeller	Label by location unconnected shapes in a binary image (Range 0-255)	0:GrayImage [Binary]	0:GrayImage
LabelByArea	Label the unconnected shapes in a binary image in relation to their size (Range 0-255)	0:GrayImage [Binary]	0:GrayImage
MeasureLabelledObjects	Measure the N (user specified) largest objects in a binary image (Range 0-255)	0:GrayImage [Binary] 1:Integer [limit on the number of labelled objects measured, default=5]	0:String [contains data describing the measured objects: (Grey Scale, Area, Centroid)]
WhiteBlobCount	Count the number of white bobs in a binary image (Range 0-255)	0:GrayImage [Binary]	0:Integer [Range 0-255] 1:GrayImage [A white cross is overlaid on each blob found.]
Label_16	Label by location the unconnected shapes in a binary image (Range 0-65535). Note: This is outside the 8-bit display range. Slow process.	0:GrayImage [Binary]	0:GrayImage
WhiteBlobCount_16	Count the number of white bobs in a binary image (Range 0-65535). Slow process.	0:GrayImage [Binary]	0:Integer [Range 0-65535] 1:GrayImage [A white cross is overlaid on each blob found.]
ConvexHull	Compute the convex hull boundary	0:GrayImage [Binary]	0:GrayImage [Binary]
FilledConvexHull	Compute the filled convex hull	0:GrayImage [Binary]	0:GrayImage [Binary]
CrackDetector	Highlight cracks in the input image	0:GrayImage	0:GrayImage
EulerNumberCalculator	Compute the Euler number from a binary image	0:GrayImage [Binary]	0:Integer [Euler number]
WhitePixelCounter	Compute the number of white pixels	0:GrayImage	0:Integer [pixel count]
IsolateHoles	Isolate holes in a binary image	0:GrayImage [Binary]	0:GrayImage [Binary]
IsolateBays	Isolate bays in a binary image	0:GrayImage [Binary]	0:GrayImage [Binary]
ConnectivityDetector	Connectivity detection in a thinned skeleton binary image. Mark points critical for connectivity in a 3x3 region.	0:GrayImage [Binary]	0:GrayImage
BoundingBox	Minimum area bounding rectangle	0:GrayImage	0:GrayImage
FilledBoundingBox	Filled minimum area bounding rectangle	0:GrayImage	0:GrayImage
BoundingBoxTopCoordinate	Compute the top left coordinate of the minimum area bounding rectangle	0:GrayImage	0:Coordinate [top left]

BoundingBoxBottomCoordinate	Compute the bottom right coordinate of the minimum area bounding rectangle	0:GrayImage	0:Coordinate [bottom right]
CornerDetector	Grey Scale (SUSAN) corner detector	0:GrayImage 1:Integer [Brightness threshold] 2:Integer [Geometric threshold]	0:GrayImage [Corner points]
K-MEANS CLUSTERING			
GrayScaleCluster	Cluster a grey scale image (number of clusters are user defined) using the k-means algorithm.	0:GrayImage 1:Integer [Number of clusters]	0:GrayImage [Gray-scale]
ColorCluster	Cluster a colour image (number of clusters are user defined) using the k-means algorithm.	0:Image [Color Image Input] 1:Integer [Number of clusters]	0:GrayImage [Gray-scale]
Un_ColorCluster	Unsupervised colour clustering using the k-means algorithm.	0:Image 1:Double [Low threshold (possible values 0.5-1.0), default=0.6] 2:Double [High threshold (possible values 1.0-1.5), default=1.2]	0:GrayImage [Gray-scale] 1:Image [Colour] 2:Integer [Number of clusters]
PseudoColor	Pseudo-colour operation	0:Image [grey-scale or colour image]	0:Image [false colour image]
TRANSFORM[#]			
MedialAxisTransform	Medial axis transform operation. Binary image showing the simple skeleton	0:Image [binary]	0:Image [binary]
MedialAxisTransform_GS	Medial axis transform operation. GS image where each point on the skeleton has an intensity which represents its distance to a boundary in the original object	0:Image [binary]	0:GrayImage [grey scale]
FFT	Fast Fourier Transform: FFT	0:GrayImage [Input image dimension must be a power of 2]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
IFFT	Inverse Fourier Transform	0:File [A Fourier data file which shall be interpreted as an image.]	0:GrayImage [The resulting gray-scale image which represents the interpreted Fourier data]

[#] Some of these functions use data types / variables that are for internal NeatVision use **only**. Access to such data (e.g. pixel access) is can be done directly in Java, see example in Appendix A.1

FFTLowpass	Low pass frequency filter	0:File [Fourier Data File] 1:Double [cut-off value (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTHighpass	High pass frequency filter	0:File 1:Double [cut-off value (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTAdaptiveLowpass	FFT adaptive lowpass filter	0:File 1:Double [limit (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTBandpass	FFT band-pass filter	0:File [Fourier Data File] 1:Double [inner limit (0-1.0)] 2:Double [outer limit (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTBandstop	FFT band-stop filter	0:File [Fourier Data File] 1:Double [inner limit (0-1.0)] 2:Double [outer limit (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTMultiply	Multiply two Fourier data files	0:File [Fourier Data File] 1:File [Fourier Data File]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTDivide	Divide one Fourier data file by another	0:File [Fourier Data File] 1:File [Fourier Data File]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTGaussian	FFT Gaussian filter. Input 0 requires an integer value that = 2^n where n is a +ve integer. Note: size = width = height	0:Integer [size of a new Fourier data file which contains Gaussian coefficients] 1:Double [Standard deviation of the Gaussian coefficients (0.1-5.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]

FFTSelectivePass	FFT selective frequency filter	0:File [Fourier Data File] 1:Double [The cutoff value of the filter (0-1.0)] 2:Double [The x-offset of the symmetric selective filter (0-1.0)] 3:Double [The y-offset of the symmetric selective filter (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
FFTSymmetricSelectivePass	FFT selective symmetric frequency filter	0:File [Fourier Data File] 1:Double [The cutoff value of the filter (0-1.0)] 2:Double [The x-offset of the symmetric selective filter (0-1.0)] 3:Double [The y-offset of the symmetric selective filter (0-1.0)]	0:File [Fourier Data File] 1:GrayImage [Grey-scale image transformed to its Fourier coefficients]
DCT2D	Direct Cosine Transform operation	0:GrayImage [Input image dimension must be a power of 2]	0:GrayImage [Real Part] 1:GrayImage [DCT Magnitude]
IDCT2D	Inverse DCT (filtering factor is user defined)	0:GrayImage 1:Double [DCT quality coefficient (0-2.0)]	0:GrayImage [IDCT image]
Hough	Line Hough Transform	0:GrayImage [Binary]	0:GrayImage
InverseHough	Inverse Hough Transform. The integer input specifies how many of the brightest pixels shall be taken into account when performing the Inverse Hough operation.	0:GrayImage 1:Integer [Number of bright points to be considered, default=10]	0:GrayImage
CircHough	Circular Hough Transform	0:GrayImage [binary image to be subjected to the circular Hough transform]	0:GrayImage [Image] 1:GrayImage [Transform space]
CooccurrenceMatrixGenerator	Compute the co-occurrence matrix	0:GrayImage	0:GrayImage
CooccurrenceMatrixEnergyCalculator	Compute the energy of the co-occurrence matrix	0:GrayImage	0:Double
CooccurrenceMatrixEntropyCalculator	Compute the entropy of the co-occurrence matrix	0:GrayImage	0:Double
CooccurrenceMatrixContrastCalculator	Compute the contrast of the co-occurrence matrix	0:GrayImage	0:Double
CooccurrenceMatrixHomogenityCalculator	Compute the homogeneity of the co-occurrence matrix	0:GrayImage	0:Double

DistanceTransform3x3	Compute the distance transform in a 3x3 window (input binary image)	0:GrayImage [Binary]	0:GrayImage
DistanceTransform5x5	Compute the distance transform in a 5x5 window (input binary image)	0:GrayImage [Binary]	0:GrayImage
LeftToRightDistanceTransform	Left to right distance transform (input binary image)	0:GrayImage [Binary]	0:GrayImage
RightToLeftDistanceTransform	Right to left distance transform (input binary image)	0:GrayImage [Binary]	0:GrayImage
TopToBottomDistanceTransform	Top to bottom distance transform (input binary image)	0:GrayImage [Binary]	0:GrayImage
BottomToTopDistanceTransform	Bottom to top distance transform (input binary image)	0:GrayImage [Binary]	0:GrayImage
GrassFireTransform	Grass fire transform (input binary image) [8-connected]	0:Image [Binary]	0:Image [grey-scale]
MORPHOLOGY			
Dilation	Dilation operation (user specify connectivity of the structured element 4 or 8)	0:GrayImage 1:Integer [(4 or 8), default=8]	0:GrayImage
Erosion	Erosion operation (user specify connectivity of the structured element 4 or 8)	0:GrayImage 1:Integer [(4 or 8), default=8]	0:GrayImage
Open	Opening operation (user specify connectivity of the structured element 4 or 8)	0:GrayImage 1:Integer [(4 or 8), default=8]	0:GrayImage
Close	Closing operation (user specify connectivity of the structured element 4 or 8)	0:GrayImage 1:Integer [(4 or 8), default=8]	0:GrayImage
ErodeNxN	Erosion operation with a user defined NxN structuring element (X or null = don't cares)	0:GrayImage 1:Integer [Array]	0:GrayImage
DilateNxN	Dilation operation with a user defined NxN structuring element (X or null = don't cares)	0:GrayImage 1:Integer [Array]	0:GrayImage
MorphologicalValley	Morphological valley operation (user specify connectivity of the structured element 4 or 8) [Default=8]	0:GrayImage 1:Integer (4 or 8)	0:GrayImage
MorphologicalTophat	Morphological top hat operation (user specify connectivity of the structured element 4 or 8) [Default=8]	0:GrayImage 1:Integer (4 or 8)	0:GrayImage
HitAndMiss	Hit and miss transformation. Hit and miss array masks must not overlap.	0:GrayImage 1:Integer [user defined hit array, blanks correspond to DON'T CARE)] 2:Integer [user defined miss array]	0:GrayImage

MorphGradient	Morphological Gradient (user specify connectivity of the structured element 4 or 8) [Default=8]	0:GrayImage 1:Integer	0:GrayImage
ReconByDil	Reconstruction by dilation	0:GrayImage 1:GrayImage [Seed] 2:Integer [SE size]	0:GrayImage [Reconstructed] 1:GrayImage [Elements removed]
ReconByDil_UI	Reconstruction by dilation via a user selected seed point (8-connected).	0:GrayImage [User interaction]	0:GrayImage [Reconstructed] 1:GrayImage [Elements removed]
DBLT	Double [Hysteresis] Threshold based reconstruction. Binary Outputs. Seed threshold to reduce noise Mask threshold to maximise signal	0:GrayImage 1:Integer [seed threshold] 2:Integer [mask threshold]	0:GrayImage [Reconstructed] 1:GrayImage [Seed Image] 2:GrayImage [Seed Image]
Watershed	Watershed transform (return the watershed image and the region boundaries image)	0:GrayImage	0:GrayImage [Watershed Image] 1:GrayImage [Binary, Watershed boundaries]
COLOUR			
GreyScaler	Average three colour planes	0:Image [colour]	0:GrayImage
ColourToRGB	Extract the RGB color planes	0:Image [colour]	0:GrayImage [R] 1:GrayImage [G] 2:GrayImage [B]
RGBToColour	Create an image from individual RGB channels	0:GrayImage [R] 1:GrayImage [G] 2:GrayImage [B]	0:Image [colour]
ColourToHSI	Extract the HSI colour planes	0:Image [colour]	0:GrayImage [H] 1:GrayImage [S] 2:GrayImage [I]
HSIToColour	Create an image from individual HSI planes	0:GrayImage [H] 1:GrayImage [S] 2:GrayImage [I]	0:Image [colour]
ColourToOpponent	Extract the opponent process colour representation	0:Image [colour]	0:GrayImage [Red_Green] 1:GrayImage [Blue_Yellow] 2:GrayImage [White_Black]
ViewOpponent	Normalize (0-255) opponent process colour channels. Used to view the normalized colour (unsaturated) channels	0:GrayImage [Red_Green] 1:GrayImage [Blue_Yellow] 2:GrayImage [White_Black]	0:GrayImage [Red_Green] 1:GrayImage [Blue_Yellow] 2:GrayImage [White_Black]
ColourToCMY	Extract the CMY (Cyan, Magenta, Yellow) colour planes	0:Image [colour]	0:GrayImage [C] 1:GrayImage [M] 2:GrayImage [Y]
CMYToColour	Create an image from individual CMY (Cyan, Magenta, Yellow) planes	0:GrayImage [C] 1:GrayImage [M] 2:GrayImage [Y]	0:Image [colour]

ViewCMY	Normalize (0-255) CMY channels. Used to view the normalized colour (unsaturated) channels	0:GrayImage [C] 1:GrayImage [M] 2:GrayImage [Y]	0:GrayImage [C] 1:GrayImage [M] 2:GrayImage [Y]
ColourToYUV	Extract the YUV colour planes	0:Image [colour]	0:GrayImage [Y] 1:GrayImage [U] 2:GrayImage [V]
YUVToColour	Create an image from individual YUV planes	0:GrayImage [Y] 1:GrayImage [U] 2:GrayImage [V]	0:Image [colour]
ViewYUV	Normalize (0-255) YUV channels. Used to view the normalized colour (unsaturated) channels	0:GrayImage [Y] 1:GrayImage [U] 2:GrayImage [V]	0:GrayImage [Y] 1:GrayImage [U] 2:GrayImage [V]
ColourToYIQ	Extract the YIQ colour planes.	0:Image [colour]	0:GrayImage [Y] 1:GrayImage [I] 2:GrayImage [Q]
YIQToColour	Create an image from individual YIQ planes	0:GrayImage [Y] 1:GrayImage [I] 2:GrayImage [Q]	0:Image [colour]
ViewYIQ	Normalize (0-255) YIQ channels. Used to view the normalized colour (unsaturated) channels	0:GrayImage [Y] 1:GrayImage [I] 2:GrayImage [Q]	0:GrayImage [Y] 1:GrayImage [I] 2:GrayImage [Q]
ColourToXYZ	Extract the XYZ colour planes	0:Image [colour]	0:GrayImage [X] 1:GrayImage [Y] 2:GrayImage [Z]
XYZToColour	Create an image from individual XYZ planes	0:GrayImage [X] 1:GrayImage [Y] 2:GrayImage [Z]	0:Image [colour]
ViewXYZ	Normalize (0-255) XYZ channels. Used to view the normalized colour (unsaturated) channels	0:GrayImage [X] 1:GrayImage [Y] 2:GrayImage [Z]	0:GrayImage [X] 1:GrayImage [Y] 2:GrayImage [Z]
ColourToLAB	Extract the Lab colour planes.	0:Image [colour]	0:GrayImage [L] 1:GrayImage [a] 2:GrayImage [b]
LABToColour	Create an image from individual Lab planes	0:GrayImage [L] 1:GrayImage [a] 2:GrayImage [b]	0:Image [colour]
ViewLAB	Normalize (0-255) Lab channels. Used to view the normalized colour (unsaturated) channels.	0:GrayImage [L] 1:GrayImage [a] 2:GrayImage [b]	0:GrayImage [L] 1:GrayImage [a] 2:GrayImage [b]

3D VOLUME			
DicomSave	A grey-scale volume image whose pixels shall be saved into DICOM format (*.dcm) (Double-click to activate). Requires the DICOM header file generated by *DicomRead*.	0:VolumeImage 1:String [Path of the original DICOM header file]	
DicomRead	Extract the grey-scale volume image data from a DICOM image. The header information is also made available to be passed to *DicomSave*		0:VolumeImage 1:String [Path of the original DICOM header file] 2:String [DICOM header]
XYZviewer	Slices in a grey-scale 3D image are viewed from their X, Y and Z directions	0:VolumeImage	
IMGfrom3D	Get a slice from a 3D data set (slice is specified by user). Returns the min max pixel values from within the slice.	0:VolumeImage 1:Integer [Range: 1 to the number of slices in the volume, default=1]	0:GrayImage 1:Integer [minimum pixel value within slice] 2:Integer [maximum value within slice]
Scale3dData	Scale pixel values in a 3D grey-scale image to the range 0 – integer input	0:VolumeImage 1:Integer [Scale range required, (default=255)]	0:VolumeImage
Thres3D	Threshold the 3D data	0:VolumeImage [Grey-scale] 1:Integer [Threshold value, default=200]	0:VolumeImage [Binary]
Mask3D	Generate a 3D mask. Zeros a user-defined number of rows, columns and slices.	0:VolumeImage [Grey-scale] 1:Integer [size of 3D mask, default=1]	0:VolumeImage [Grey-scale]
Sobel3D	3D Sobel 3x3x1 (18-neighbourhood) edge detector	0:VolumeImage [Grey-scale]	0:VolumeImage [Grey-scale]
Blob3D	Extract the 3D blobs from binary 3D image. Each blob is assigned a grey scale value.	0:VolumeImage [Binary]	0:VolumeImage [Binary]
BigestBlob3D	Extract the N (user defined) biggest 3D blobs from 3D binary image.	0:VolumeImage [Binary] 1:Integer [Number of large blobs required, range 0-255, default=1]	0:VolumeImage [Binary]
Thinning3D	3D thinning operation of a binary 3D data set	0:VolumeImage [Binary]	0:VolumeImage [Binary]
MIP	Maximum intensity projection transform	0:VolumeImage	0:GrayImage
AIP	Average intensity projection transform	0:VolumeImage	0:GrayImage

PushSlice	Push (insert) an image slice into the 3D data set	0:VolumeImage 1: GrayImage [Image to be inserted] 2:Integer [slice number - between 1 and depth (default=1)] 3:Integer [minimum pixel value within slice (default=1)] 4:Integer [maximum pixel value within slice (default=255)]	0:VolumeImage
RenderEngine	Surface rendering of a binary image. Image can be displayed as a cloud of points, wire frame, flat shading, Gouraund shading and Phong shading. (Double-click to activate). Allows user to translate, scale and rotate image.	0:VolumeImage	0:VolumeImage
LOW LEVEL[#]			
GetPixel	A grey-scale image from which a pixel intensity at a certain coordinate is obtained. **NB:** See Appendix A.1 for a more efficient way to directly manipulate pixel data. NeatVision low level methods are not recommend for such low level pixel operations.	0:GrayImage 1: Coordinate [coordinate of the pixel in question]	0: Integer [intensity of the pixel at the specified coordinate]
SetPixel	A grey-scale image from which a pixel at a certain coordinate is replaced with one of a user defined intensity.	0:GrayImage 1: Integer [grey-scale intensity of the replacement pixel] 2: Coordinate [coordinate of the pixel in question]	0:GrayImage
RemovePixel	A grey-scale image from which a pixel at a certain coordinate is removed (removing a pixels sets that pixel to black).	0:GrayImage 1: Coordinate [coordinate of the pixel in question]	0:GrayImage

[#] Some of these functions use data types / variables that are for internal NeatVision use **only**. Access to such data (e.g. pixel access) is can be done directly in Java, see example in Appendix A.1

DrawLine	Draw a line in the grey-scale image	0:GrayImage 1: Coordinate [starting coordinate of the line] 2: Coordinate [finishing coordinate of the line] 3: Integer [gray-scale intensity of the line]	0:GrayImage
DrawBox	Draw a hollow box in the grey-scale image	0:GrayImage 1: Coordinate [upper top left] 2: Coordinate [lower bottom right] 3: Integer [grey-scale intensity]	0:GrayImage
FillBox	Draw a filled box in the grey-scale image	0:GrayImage 1: Coordinate [upper top left] 2: Coordinate [lower bottom right] 3: Integer [fill grey-scale intensity]	0:GrayImage
DrawCircle	Draw a white hollow circle in the grey-scale image	0:GrayImage 1: Coordinate [coordinate of the centre of the circle] 2: Integer [radius]	0:GrayImage
FillCircle	Draw a white filled circle in the grey-scale image	0:GrayImage 1: Coordinate [coordinate of the centre of the circle] 2: Integer [radius]	0:GrayImage
GetImageWidth	Width of the input grey-scale image	0:GrayImage	0: Integer [width of the input grey-scale image]
GetImageHeight	Height of the input grey-scale image	0:GrayImage	0: Integer [height of the input grey-scale image]
GenerateCoordinate	Generate the coordinate value from the (x,y) components.	0: Integer [x] 1: Integer [y]	0: Coordinate
GeneratePoints	Generate the (x,y) components of a given coordinate.	0: Coordinate	0: Integer [x] 1: Integer [y]

STRING			
StringAdd	Combine two strings (objects)	0: Undefined [first of two strings (objects) which are to be added] 1: Undefined [second of two strings (objects) which are to be added] 2:	0: String [The resulting string which is made up from the two input strings]
StringToLowerCase	A string which shall be converted to lower case	0: String	0: String
StringToUpperCase	A string which shall be converted to upper case	0: String	0: String
MATH[#]	Library of standard mathematical operators.		
JAIColour	See the Java™ Advanced Imaging website: *http://java.sun.com/products/java-media/jai/*		

OSMIA – Tina 5 Interface[4]			
NemaReader	Read a *Nema* image	Path of the file through the graphical interface	0: GrayImage 1: Double [minimum pixel value] 2: Double [maximum pixel value]
AiffReader	Read an *Aiff* image	Path of the file through the graphical interface	0: GrayImage 1: Double [minimum pixel value] 2: Double [maximum pixel value]
RenderEngineN	Render a binary volume image	0: VolumeImage [binary] 1: Integer [Thickness factor: Values: 1 to n]	
Ej_Frac	Compute the ejection fraction from two volume images.	0: VolumeImage [systole] 1: VolumeImage [diastole]	0: VolumeImage [binary] 1: VolumeImage [binary] 2: Double [Ejection fraction value]

[#] Some of these functions use data types / variables that are for internal NeatVision use **only**. Access to such data (e.g. pixel access) is can be done directly in Java, see example in Appendix A.1

[4] See http://www.eeng.dcu.ie/~whelanp/osmia/ for details on interfacing NeatVision with Tina 5.0

Flow2D	Compute the 2D optical flow (Horn-Schunck or Lucas-Kanade). Original code by Prof. John Barron, UWO, Canada	0: String [Directory path for optical flow RAS sequence] 1: String [Stem name of the sequence] 2: Boolean [Swap data: PC should be TRUE] 3: Boolean [Method: TRUE : Horn-Schunck. FALSE: Lucas-Kanade] 4: Integer [Flow number: middle index of the sequence] 5: Double [Tau parameter, Default value: 0.5] 6: Double [Alpha parameter, Default value: 1.0] 7: Integer [Number of iterations, Default value: 50] 8: Integer [Offset parameter, Default value: 6] 9: Double [Scale parameter, Default value 12.0]	0: GrayImage [Output image illustrating the flow vectors]
Aorta_n	Detect the aorta outline in a greyscale image.	0: GrayImage [Input image] 1: String [Path of the model data] 2: String [Path of the pca_model data] 3: Double [minimum pixel value] 4: Double [maximum pixel value]	0: Image [RGB image highlighting the aorta outline]
TSmooth	Tangential smooth operator	0: GrayImage [Input image] 1: Integer [Number of iterations]	0: GrayImage [Output image]
XY_Norm	Coil correction algorithm	0: GrayImage [Input image] 1: Double [Standard deviation]	0: GrayImage [Output image]
st_rec	Stereo rectification algorithm	0: String [Path of left image – aiff format] 1: String [Path of right image – aiff format] 2: String [Path of left cam file] 3: String [Path of right cam file]	0: GrayImage [Input left image] 1: GrayImage [Input right image] 2: GrayImage [Output left image] 3: GrayImage [Output right image]

Pairwise	Pairwise geometric histograms 2D object recognition	0: String [Directory path] 1: String [Filename – scene data] 2: String [Filename – model data]	0: Image [RGB image: Scene data] 1: Image [RGB image: Model data] 2: Image [RGB image: Recognised model superimposed on the input scene data]

References:

1. Paul F. Whelan, Robert J. T. Sadleir, and Ovidiu Ghita, (2004) "Informatics in Radiology (infoRAD): NeatVision: Visual Programming for Computer-aided Diagnostic Applications", Radiographics; 24(6):1779-1789

2. Robert J. T. Sadleir, Paul F. Whelan, Padraic MacMathuna, and Helen M. Fenlon (2004) "Informatics in Radiology (infoRAD): A Portable Toolkit for Providing Straightforward Access to Medical Image Data" Radiographics. 2004 Jul-Aug:24(4):1193-1202

3. Paul F. Whelan and R.J.T. Sadleir (2004), "A Visual Programming Environment for Machine Vision Engineers", Sensor Review, 24(3):265-270

4. Paul F. Whelan (2004), Neatvision 2.1 Developers Guide.

5. Paul F. Whelan (2003), "Automated cutting of natural products: A practical packing strategy", Chapter 11 in Machine Vision for the Inspection of Natural Products. Mark Graves and Bruce Batchelor (Eds.), Springer (London). , ISBN: 1-85233-525-4, pp 307-329

6. Paul F. Whelan (2001), "Visual programming for machine vision", Chapter 8 in Intelligent Machine Vision: Techniques, Implementation & Interfacing B.G. Batchelor and F. Waltz, Springer-Verlag UK, 448 pages, ISBN: 3-540-762248, pp 229-320

7. Paul F. Whelan and D. Molloy (2000), Machine Vision Algorithms in Java: Techniques and Implementation, Springer (London), 298 Pages. ISBN 1-85233-218-2.

8. B.G. Batchelor and Paul F. Whelan (1997), Intelligent Vision Systems for Industry, Springer-Verlag (London), 457 pages, ISBN 3-540-19969-1.

9. Paul F. Whelan (1997), "Remote access to continuing engineering education - RACeE", IEE Engineering Science and Education Journal, 6(5), pp 205-211. Also published in the IEE Computer Forum.

10. Paul F. Whelan, B.G. Batchelor, M.R.F. Lewis and R. Hack (1997), "Machine vision and the World Wide Web: Design and training aids", Proceedings of the SPIE - The International Society for Optical Engineering, Vol. 3205 - Machine Vision Applications, Architectures, and Systems Integration VI, Pittsburgh (USA), pp 284-294.

11. B.G. Batchelor and Paul F. Whelan (1995), "Real-time colour recognition in symbolic programming for machine vision systems", Machine Vision and Applications; 8(6):385-398

12. B.G. Batchelor and Paul F. Whelan (Eds) (1994), Selected Papers on Industrial Machine Vision Systems, SPIE Milestone Series MS 97, SPIE Optical Engineering Press, 629 pages