# Cell Animation

## Introduction

CellAnimation is a framework for microscopy assays. To allow for fast assay creation and code reuse each assay is implemented as a modular pipeline.  A CellAnimation assay is a chain of MATLAB structures. Each structure describes a module and its connectivity.  We have provided a series of assays that we have developed for various microscopy tasks for our lab and others. The intended purpose for each of the assays is described in the “Assays” section.

The CellAnimation core functions are responsible for reading the module chain, creating a dependency tree, populating the input values, executing each module  and saving those output values  required by modules further downstream (Supp. Fig. 1A).  A module is a reusable set of functions that has a narrow specific use (image input-output, thresholding, segmentation, annotation, etc.). Each module we provide is documented in the “Internal Modules” section.  In addition we have also included a set of control modules. These are special modules that operate on other modules. Through the use of control modules we can implement looping and branching at the pipeline level.  The benefits to this approach are twofold.  First, the assay logic becomes easier to follow and modify.  Second, it allows us to use smaller, more reusable modules. Individual control modules are documented in the “Control Modules” section.

In addition to our modules, a pipeline may include modules that are just simple wrappers for MATLAB functions or modules that encapsulate functions developed by others.  For the MATLAB wrapper modules the documentation may be found in the in the MATLAB help file for that particular function. The caveat is that a particular wrapper may only provide access to some of the functionality of the MATLAB function. In general, MATLAB wrapper modules are named using the name of the MATLAB function concatenated with the string “Wrapper”, however there are some older wrapper modules that only use the name of the MATLAB function.  For other external modules the documentation is provided by the original developers.

## Assays

### assayBrightFieldCytoTestNN

#### Usage

This assay has been optimized for tracking cells imaged using bright-field microscopy using a nearest-neighbor algorithm.  The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

### Important Parameters
*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images. Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN(‘c:\test_movie’).
*TrackStruct.DS* – The directory separator to be used (‘\’ for Windows, ‘/’ for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as “image001.jpg”, '%04d' for an image containing 4 digits such as “image0001.jpg”, etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules
areaFilterLabel, assignCellToTrackUsingNN, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, generateBinImgUsingGradient, removeShortTracks, segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

## assayBrightFieldCytoTestWG

### Usage
This assay has been optimized for tracking cells imaged using bright-field microscopy using a custom algorithm. This algorithm is more accurate and flexible than the nearest-neighbor algorithm at the expense of execution speed. The assay tracks the cells, detects mitotic events and records tracking data, cell shape parameters and ancestry information to spreadsheets. For each tracked image, an overlayed image is saved to disk that displays the detected cell outlines, cell IDs and cell generation.

### Important Parameters
*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images. Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN(‘c:\test_movie’).
*TrackStruct.DS* – The directory separator to be used (‘\’ for Windows, ‘/’ for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.

*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as “image001.jpg”, '%04d' for an image containing 4 digits such as “image0001.jpg”, etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules
areaFilterLabel, assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, generateBinImgUsingGradient, removeShortTracks, segmentObjectsUsingClusters, segmentObjectsUsingMarkers, splitTracks.

## assayCellCoverage

### Usage
This assay is used to determine what percentage of an image is occupied by objects.

### Important Parameters
*img_file_name* – The absolute image file name of the image to be analyzed.

### Important Modules
manualSegmentationReview.

## assayCellPropsSingleImage

### Usage
This assay is used to segment objects in an image subject to manual review then extract their shape properties.

### Important Parameters
*path_name* – The absolute image file name of the image to be analyzed.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules
clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, manualSegmentationReview, segmentObjectsUsingMarkers.

## assayFlCytoLNCapManSegWG

### Usage
This assay is used to manually review segmentation of objects labeled using a fluorescent dye after they have been segmented using another assay.

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images. Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN('c:\test_movie').
*TrackStruct.DS* – The directory separator to be used ('\' for Windows, '/' for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, ".jpg" or ".tif".
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are "Experiment-0002_Position(8)_t001.jpg","Experiment-0002_Position(8)_t002.jpg", etc., the root image file name is "Experiment-0002_Position(8)_t".
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as "image001.jpg", '%04d' for an image containing 4 digits such as "image0001.jpg", etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

clearSmallObjects, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, manualSegmentationReview, polygonalAssistedWatershed, segmentObjectsUsingMarkers.

## assayFlCytoLNCapTracksReviewWG

### Usage

This assay is used to manually review the automatic tracks generated using a tracking assay.

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images. Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN('c:\test_movie').
*TrackStruct.DS* – The directory separator to be used ('\' for Windows, '/' for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, ".jpg" or ".tif".
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are "Experiment-0002_Position(8)_t001.jpg","Experiment-0002_Position(8)_t002.jpg", etc., the root image file name is "Experiment-0002_Position(8)_t".
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as "image001.jpg", '%04d' for an image containing 4 digits such as "image0001.jpg", etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.

Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

manualTrackingReview.

## assayFlCytoLNCapTrackWG

### Usage

This assay is used to automatically track cells that have been segmented using another assay.

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images.  Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN(‘c:\test_movie’).
*TrackStruct.DS* – The directory separator to be used (‘\’ for Windows, ‘/’ for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as “image001.jpg”, '%04d' for an image containing 4 digits such as “image0001.jpg”, etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

assignCellToTrackUsingAll, detectMitoticEvents, splitTracks.

## assayFluoNuclLNCap

### Usage

This module is used to track cells that have been labeled using a nuclear stain, record shape and ancestry data and create an overlay movie for visual inspection.

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images.  Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN(‘c:\test_movie’).
*TrackStruct.DS* – The directory separator to be used (‘\’ for Windows, ‘/’ for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.

*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as “image001.jpg”, '%04d' for an image containing 4 digits such as “image0001.jpg”, etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, removeShortTracks, segmentObjectsUsingMarkers, solidityFilterLabel, splitTracks.

## assayFluoNuclTestCADir

### Usage

This module is used to track fast highly directional cells that have been stained using a nuclear stain.

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images.  Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN(‘c:\test_movie’).
*TrackStruct.DS* – The directory separator to be used (‘\’ for Windows, ‘/’ for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as “image001.jpg”, '%04d' for an image containing 4 digits such as “image0001.jpg”, etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

areaFilterLabel, areaOverPerimeterFilterLabel, assignCellToTrackUsingAll, clearSmallObjects, detectMergeCandidatesUsingDistance, detectMitoticEvents, distanceWatershed, generateBinImgUsingLocAvg, getConvexObjects, polygonalAssistedWatershed, removeShortTracks, segmentObjectsUsingMarkers, solidityFilterLabel, splitTracks.

## assayFluoNuclThresholding

### Usage

This module is used to threshold a series of images (no object segmentation and no tracking are performed).

### Important Parameters

*well_folder* – String that specifies the absolute location of the directory which contains the time-lapse images.  Needs to be specified every time the assay is run. For example, assayBrightFieldCytoTestNN('c:\test_movie').
*TrackStruct.DS* – The directory separator to be used ('\' for Windows, '/' for Linux/Unix).
*TrackStruct.FrameCount* – The number of frames to track.
*TrackStruct.ImgExt* – String indicating the image file extension. Usually, ".jpg" or ".tif".
*TrackStruct.ImageFileName* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are "Experiment-0002_Position(8)_t001.jpg","Experiment-0002_Position(8)_t002.jpg", etc., the root image file name is "Experiment-0002_Position(8)_t".
*TrackStruct.NumberFormat* – The number format of the counter in the image file name. For example '%03d' for an image file name containing 3 digits such as "image001.jpg", '%04d' for an image containing 4 digits such as "image0001.jpg", etc.
*TrackStruct.StartFrame* – Integer indicating at which frame the tracking should start.
*TrackStruct.TimeFrame* – The time frame between consecutive images.
Other important parameters are those listed in the module section for the important modules listed below.

### Important Modules

generateBinImgUsingLocAvg.

## assayGetStageOffset

### Usage

This assay is used to calculate the microscope stage offset at each frame by manually clicking on a fixed point at each frame.

### Important Parameters

*file_root* – The root file name of the image series. For example if the image series is "image001.jpg", "image002.jpg", etc. the root file name will be "image".
*frame_count* – The number of frames to analyze.
*FrameStep* – Optional. Read one out of every x frames when reading the image set. Default value is one meaning every frame will be read.
*ImageExtension* – The image extension of the images in the series (usually, ".tif" or ".jpg").
*NumberFormat* – String representing the format of the counter in the image series. Follows *sprintf* format.
*StartFrame* – Index indicating starting image from which the image sequence will be read.

### Important Modules
None.

## assayOffsetFrames

### Usage
This module is used to offset and crop frames to remove incorrect offsets due to errors in automatic stage return. Use with offset data acquired using an assay such as assayGetStageOffset.  The original images will be untouched and the cropped images will be saved in a “cropped frames” directory under the original image folder.

### Important Parameters
*file_root* – The root file name of the image series. For example if the image series is “image001.jpg”, “image002.jpg”, etc. the root file name will be “image”.
*frame_count* – The number of frames to analyze.
*FrameStep* – Optional. Read one out of every x frames when reading the image set. Default value is one meaning every frame will be read.
*ImageExtension* – The image extension of the images in the series (usually, “.tif” or “.jpg”).
*NumberFormat* – String representing the format of the counter in the image series. Follows *sprintf* format.
*StartFrame* – Index indicating starting image from which the image sequence will be read.

# Control Modules

## forLoop

### Usage
This module is used to loop through a series of modules. The modules that will be looped through are listed in the *LoopFunctions* input structure member.

### Input Structure Members
*EndLoop* – The loop will stop when the loop counter reaches this value.
*IncrementLoop* – The value by which the loop counter will be incremented every time one loop cycle is completed.
*LoopFunctions* – The list of module input structures that will be looped through.
*StartLoop* – The loop counter will be initialized to this value.
Any other arguments may be added to the *FunctionArgs* structure and they will be available to the modules contained in the *LoopFunctions* structure member. This allows the loop modules to have access to values generated by modules outside the loop.

### Output Structure Members
Any values generated by the modules in the loop can be made available to modules outside the loop by adding them to the *KeepValues* structure.

### Example

```
image_read_loop_functions=[];
image_read_loop.InstanceName='SegmentationLoop';
image_read_loop.FunctionHandle=@forLoop;
image_read_loop.FunctionArgs.StartLoop.Value=TrackStruct.StartFrame;
image_read_loop.FunctionArgs.EndLoop.Value=(TrackStruct.StartFrame+TrackStruc
t.FrameCount-1)*TrackStruct.FrameStep;
image_read_loop.FunctionArgs.IncrementLoop.Value=TrackStruct.FrameStep;
image_read_loop.FunctionArgs.MatchingGroups.Value=[];
image_read_loop.FunctionArgs.MatchingGroups.FunctionInstance='IfIsEmptyPrevio
usCellsLabel';
image_read_loop.FunctionArgs.MatchingGroups.OutputArg='MatchingGroups';
image_read_loop.FunctionArgs.Tracks.FunctionInstance='IfIsEmptyPreviousCellsL
abel';
image_read_loop.FunctionArgs.Tracks.OutputArg='Tracks';
image_read_loop.FunctionArgs.Tracks.Value=[];
image_read_loop.KeepValues.Tracks.FunctionInstance='IfIsEmptyPreviousCellsLab
el';
image_read_loop.KeepValues.Tracks.OutputArg='Tracks';
image_read_loop.KeepValues.MatchingGroups.FunctionInstance='IfIsEmptyPrevious
CellsLabel';
image_read_loop.KeepValues.MatchingGroups.OutputArg='MatchingGroups';

display_curtrackframe_function.InstanceName='DisplayCurFrame';
display_curtrackframe_function.FunctionHandle=@displayVariable;
display_curtrackframe_function.FunctionArgs.Variable.FunctionInstance='Segmen
tationLoop';
display_curtrackframe_function.FunctionArgs.Variable.OutputArg='LoopCounter';
display_curtrackframe_function.FunctionArgs.VariableName.Value='Current
Tracking Frame';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displa
y_curtrackframe_function);
…
image_read_loop.LoopFunctions=image_read_loop_functions;
functions_list=addToFunctionChain(functions_list,image_read_loop);
```

## if_statement

### Usage

This module is used to create branching execution in an assay. Depending on the result of a test function either the modules in the *IfFunctions* or the modules in the *ElseFunctions* structures will be executed.

### Input Structure Members

*TestFunction* – A structure that details the function which will be used to determine what subset of modules will be executed. This function needs to return a *true/false* value.
*IfFunctions* – Set of modules which will be executed if the value returned by the test function is *true*.
*ElseFunctions* – Set of modules which will be executed if the value returned by the test function is *true*.

Any other arguments may be added to the *FunctionArgs* structure and they will be available to the modules contained in the *IfFunctions* and *ElseFunctions* structure members. This allows the modules in the "if statement" to have access to values generated by modules outside it.

## Output Structure Members

Any values generated by the modules in the loop can be made available to modules outside the loop by adding them to the *KeepValues* structure.

## Example

```
if_is_empty_cells_label_functions=[];
else_is_empty_cells_label_functions=[];
if_is_empty_cells_label_function.InstanceName='IfIsEmptyPreviousCellsLabel';
if_is_empty_cells_label_function.FunctionHandle=@if_statement;
if_is_empty_cells_label_function.FunctionArgs.TestResult.FunctionInstance='Is
EmptyPreviousCellsLabel';
if_is_empty_cells_label_function.FunctionArgs.TestResult.OutputArg='Boolean';
if_is_empty_cells_label_function.FunctionArgs.CellsLabel.FunctionInstance='Re
sizeCytoLabel';
if_is_empty_cells_label_function.FunctionArgs.CellsLabel.OutputArg='Image';
if_is_empty_cells_label_function.FunctionArgs.CurFrame.FunctionInstance='Segm
entationLoop';
if_is_empty_cells_label_function.FunctionArgs.CurFrame.OutputArg='LoopCounter
';
if_is_empty_cells_label_function.TestFunction.InstanceName='IsEmptyPreviousCe
llsLabel';
if_is_empty_cells_label_function.TestFunction.FunctionHandle=@isEmptyFunction
;
if_is_empty_cells_label_function.TestFunction.FunctionArgs.TestVariable.Funct
ionInstance='IfIsEmptyPreviousCellsLabel';
if_is_empty_cells_label_function.TestFunction.FunctionArgs.TestVariable.Input
Arg='PreviousCellsLabel'; %only works for subfunctions
if_is_empty_cells_label_function.TestFunction.FunctionArgs.PreviousCellsLabel
.Value=[];
if_is_empty_cells_label_function.KeepValues.Tracks.FunctionInstance='StartTra
cks';
if_is_empty_cells_label_function.KeepValues.Tracks.OutputArg='Tracks';
…

get_shape_params_function.InstanceName='GetShapeParameters';
get_shape_params_function.FunctionHandle=@getShapeParams;
get_shape_params_function.FunctionArgs.LabelMatrix.FunctionInstance='IfIsEmpt
yPreviousCellsLabel';
get_shape_params_function.FunctionArgs.LabelMatrix.InputArg='CellsLabel';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_
functions,get_shape_params_function);
…

make_unassigned_cells_list_function.InstanceName='MakeUnassignedCellsList';
make_unassigned_cells_list_function.FunctionHandle=@makeUnassignedCellsList;
make_unassigned_cells_list_function.FunctionArgs.CellsCentroids.FunctionInsta
nce='GetShapeParameters';
make_unassigned_cells_list_function.FunctionArgs.CellsCentroids.OutputArg='Ce
ntroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,make_unassigned_cells_list_function);
```

```
…

if_is_empty_cells_label_function.IfFunctions=if_is_empty_cells_label_function
s;
if_is_empty_cells_label_function.ElseFunctions=else_is_empty_cells_label_func
tions;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,if_is_
empty_cells_label_function);
```

## whileLoop

### Usage

This module is used to loop through a series of modules. The modules that will be looped through are listed in the *LoopFunctions* input structure member.

### Input Structure Members

*TestFunction* – This structure details the function that will be used to determine when to exit the *while* loop. When the *TestResult* variable is set to *false* by *TestFunction* the loop exits.
*TestResult* – The name of the variable in the output structure of *TestFunction* that determines when the *while* loop exits. This needs to be a boolean variable or convertible to one.
Any other arguments may be added to the *FunctionArgs* structure and they will be available to the modules contained in the *LoopFunctions* structure member. This allows the loop modules to have access to values generated by modules outside the loop.

### Output Structure Members

Any values generated by the modules in the loop can be made available to modules outside the loop by adding them to the *KeepValues* structure.

### Example

```
assign_cells_to_tracks_loop.InstanceName='AssignCellsToTracksLoop';
assign_cells_to_tracks_loop.FunctionHandle=@whileLoop;
assign_cells_to_tracks_loop.TestFunction.InstanceName='IsNotEmptyUnassignedCe
lls';
assign_cells_to_tracks_loop.TestFunction.FunctionHandle=@isNotEmptyFunction;
assign_cells_to_tracks_loop.TestFunction.FunctionArgs.TestVariable.FunctionIn
stance='AssignCellsToTracksLoop';
assign_cells_to_tracks_loop.TestFunction.FunctionArgs.TestVariable.InputArg='
UnassignedCells';
assign_cells_to_tracks_loop.FunctionArgs.TestResult.FunctionInstance='IsNotEm
ptyUnassignedCells';
assign_cells_to_tracks_loop.FunctionArgs.TestResult.OutputArg='Boolean';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance='Ma
keUnassignedCellsList';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg='Unassigne
dCellsIDs';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance2='A
ssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg2='Unassign
edIDs';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.FunctionInstance='Mak
eExcludedTracksList';
```

```
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTr
acks';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.FunctionInstance2='As
signCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.OutputArg2='ExcludedT
racks';
assign_cells_to_tracks_loop.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmp
tyPreviousCellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.CellsLabel.InputArg='CellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.PreviousCellsLabel.FunctionInstance=
'IfIsEmptyPreviousCellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.PreviousCellsLabel.InputArg='Previou
sCellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.ShapeParameters.FunctionInstance='Ge
tShapeParameters';
assign_cells_to_tracks_loop.FunctionArgs.ShapeParameters.OutputArg='ShapePara
meters';
assign_cells_to_tracks_loop.FunctionArgs.ShapeParameters.FunctionInstance2='S
etMatchingGroupIndex';
assign_cells_to_tracks_loop.FunctionArgs.ShapeParameters.OutputArg2='ShapePar
ameters';
assign_cells_to_tracks_loop.FunctionArgs.CellsCentroids.FunctionInstance='Get
ShapeParameters';
assign_cells_to_tracks_loop.FunctionArgs.CellsCentroids.OutputArg='Centroids'
;
assign_cells_to_tracks_loop.FunctionArgs.CurrentTracks.FunctionInstance='GetC
urrentTracks';
assign_cells_to_tracks_loop.FunctionArgs.CurrentTracks.OutputArg='Tracks';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.FunctionInstance='I
fIsEmptyPreviousCellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.InputArg='TrackAssi
gnments';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.FunctionInstance2='
AssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.OutputArg2='TrackAs
signments';
assign_cells_to_tracks_loop.FunctionArgs.MaxTrackID.FunctionInstance='GetMaxT
rackID';
assign_cells_to_tracks_loop.FunctionArgs.MaxTrackID.OutputArg='MaxTrackID';
assign_cells_to_tracks_loop.FunctionArgs.Tracks.FunctionInstance='IfIsEmptyPr
eviousCellsLabel';
assign_cells_to_tracks_loop.FunctionArgs.Tracks.InputArg='Tracks';
assign_cells_to_tracks_loop.FunctionArgs.PreviousTracks.FunctionInstance='Get
PreviousTracks';
assign_cells_to_tracks_loop.FunctionArgs.PreviousTracks.OutputArg='Tracks';
assign_cells_to_tracks_loop.KeepValues.TrackAssignments.FunctionInstance='Ass
ignCellToTrackUsingAll';
assign_cells_to_tracks_loop.KeepValues.TrackAssignments.OutputArg='TrackAssig
nments';
assign_cells_to_tracks_loop.KeepValues.ShapeParameters.FunctionInstance='SetM
atchingGroupIndex';
assign_cells_to_tracks_loop.KeepValues.ShapeParameters.OutputArg='ShapeParame
ters';
assign_cells_to_tracks_loop.KeepValues.MatchingGroups.FunctionInstance='Assig
nCellToTrackUsingAll';
assign_cells_to_tracks_loop.KeepValues.MatchingGroups.OutputArg='MatchingGrou
ps';
```

## Internal Modules

### addFunction

#### Usage
This module adds two variables.

#### Input Structure Members
*Number1* – The first variable to be added.
*Number2* – The second variable to be added.

#### Output Structure Members
*Sum* – The result of the addition.

#### Example
```
get_previous_frame_nr_function.InstanceName='GetPreviousFrameNr';
get_previous_frame_nr_function.FunctionHandle=@addFunction;
get_previous_frame_nr_function.FunctionArgs.Number1.FunctionInstance='Segment
ationLoop';
get_previous_frame_nr_function.FunctionArgs.Number1.OutputArg='LoopCounter';
get_previous_frame_nr_function.FunctionArgs.Number2.Value=-1;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,get_pr
evious_frame_nr_function);
…

make_mat_name_function.FunctionArgs.CurFrame.FunctionInstance='GetPreviousFra
meNr';
make_mat_name_function.FunctionArgs.CurFrame.OutputArg='Sum';
```

### areaFilterLabel

#### Usage
This module is used to remove objects below and/or above a certain area from a label matrix.

#### Input Structure Members
*MaxArea* – Objects with an area larger than this value will be removed.
*MinArea* – Objects with an area smaller than this value will be removed.
*ObjectsLabel* – The label matrix from which objects will be removed.

#### Output Structure Members
*LabelMatrix* – The filtered label matrix.

#### Example
```
area_filter_function.InstanceName='AreaFilter';
area_filter_function.FunctionHandle=@areaFilterLabel;
area_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjec
tsUsingMarkers';
area_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
area_filter_function.FunctionArgs.MinArea.Value=20;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,area_f
ilter_function);
...
resize_cyto_label_function.FunctionArgs.Image.FunctionInstance='AreaFilter';
resize_cyto_label_function.FunctionArgs.Image.OutputArg='LabelMatrix';
```

## areaOverPerimeterFilterLabel

### Usage

This module is used to remove objects below and/or above a certain area/perimeter ratio from a label matrix.  Montages consisting of multiple images can exhibit noise at the points where the individual images are joined.  The objects from this type of noise have higher area/perimeter ratio than true objects and can be removed using this filter.

### Input Structure Members

*MaxAreaOverPerimeter – Objects with an AOP ratio larger than this value will be removed.*
*MinAreaOverPerimeter – Objects with an AOP ratio smaller than this value will be removed.*
*ObjectsLabel* – The label matrix from which objects will be removed.

### Output Structure Members

LabelMatrix – The filtered label matrix.

### Example

```
aop_filter_function.InstanceName='AOPFilter';
aop_filter_function.FunctionHandle=@ areaOverPerimeterFilterLabel;
aop_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObject
sUsingMarkers';
aop_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
aop_filter_function.FunctionArgs.MinAreaOverPerimeter.Value=TrackStruct.MinCy
toAOP;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,aop_fi
lter_function);
...
resize_cyto_label_function.FunctionArgs.Image.FunctionInstance='AOPFilter';
resize_cyto_label_function.FunctionArgs.Image.OutputArg='LabelMatrix';
```

## assignCellToTrackUsingAll

### Usage

This module tracks objects in a time-lapse sequence of label matrices. It uses distance, speed, direction and shape parameters to determine which candidate object best matches a track. It can be optimized for tracking fast moving directional objects or slow moving objects.

## Input Structure Members

*CheckCellPath* – To allow paths that go through another cell set this value to true otherwise set it to false.
*CellsCentroids* – Current object centroids.
*CellsLabel* – The current time step label matrix.
*CurrentTracks* – Matrix containing the track assignments and shape parameters for the objects in the previous frame.
*DefaultParamWeights* – A set of weights is assigned to each parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. If an object can be assigned to a matching group or is a sure match for a track the weights listed in this variable are used.
*DirectionRankingOrder* – When the order of the parameters based on prediction power cannot be determined using a group match a set of default parameter ranks are used. The parameter order for fast directional objects is provided in this variable.
*DistanceRankingOrder* – When the order of the parameters based on prediction power cannot be determined using a group match a set of default parameter ranks are used. The parameter order for slow-moving objects is provided in this variable.
*ExcludedTracks* – List of track assignments that should not be changed.
*FrontParams* – To force a set of parameters to the front so they are heavily weighted enter their column indices in the variable, otherwise set the variable to an empty vector.
*MatchingGroups* – Matrix of current matching groups (vectors of shape or motility indices ordered by their prediction power).
*MatchingGroupsStats* – Matrix of mean values for each parameter in each group.
*MaxAngleDiff* – The maximum allowed angle difference between a track and a candidate object. If the angle is larger than this value direction ranking will not be used for this object.
*MaxDistRatio* – The maximum allowed distance ratio between the two nearest candidate objects. If the ratio is higher than this value distance ranking will not be used.
*MaxSearchRadius* - Sets an absolute lower bound for the search radius to prevent selecting too few candidate objects for a track.
*MaxTrackID* – Current maximum track ID.
*MinSecondDistance* – Minimum significant distance between the closest candidate object to a track and the second closest. Used to determine when distance should be used as a ranking parameter.
*MinSearchRadius* – Sets an absolute higher bound for the search radius to prevent selecting too many candidate objects for a track.
*NrParamsForSureMatch* – This value is used to indicate the minimum number of closest matches between a candidate object parameters and a track's object parameters that make the candidate object a sure match to the track.
*ParamsCoeffOfVariation* – Matrix containing the coefficient of variation for each parameter.
*PreviousCellsLabel* – The matrix label from the previous time step.
*PreviousTracks* – Matrix containing the track assignments and shape parameters for the objects in the frame previous to those in *CurrentTracks*.
*RelevantParametersIndex* – Shape or motility parameters that have been determined to be irrelevant for tracking the objects can be eliminated by setting the corresponding index in the

variable to false. This indicates to the module not to use the parameters in computing track assignment probabilities.
*SearchRadiusPct* – This value determines the size of the neighborhood from which candidate objects for matching the track are selected. It is a multiple of the distance to the nearest candidate in the current frame. Setting this variable equal to 1 turns this module into a nearest-neighbor algorithm (only the nearest cell can be a candidate). It does not make sense to have a value lower than 1.
*ShapeParameters* – Shape parameters (area, eccentricity, perimeter, etc.) extracted from the current label.
*TrackAssignments* – List of track assignments that have already been completed.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.
*UnassignedCells* – List of object IDs currently unassigned.
*UnknownParamWeights* – A set of weights is assigned to each parameter based on its prediction power. Parameters with high prediction power are assigned high weights and parameters with low prediction power are assigned lower weights. If an object cannot be assigned to a matching group and is not a sure match for a track the weights listed in this variable are used.
*UnknownRankingOrder* – When the order of the parameters based on prediction power cannot be determined using a group match a set of default parameter ranks are used.  If the objects cannot be categorized as either slow-moving or fast directional the parameter order provided in this variable is used.

## Output Structure Members

*UnassignedIDs* – List of object IDs currently unassigned.
*TrackAssignments* – List of track assignments that have already been completed.
*MatchingGroups* – Matrix of mean values for each parameter in each group.
*GroupIndex* – Indicates the index of the group to which the object has been assigned.
*ExcludedTracks* – List of track assignments that should not be changed.

## Example

```
assign_cell_to_track_function.InstanceName='AssignCellToTrackUsingAll';
assign_cell_to_track_function.FunctionHandle=@assignCellToTrackUsingAll;
assign_cell_to_track_function.FunctionArgs.UnassignedCells.FunctionInstance='
AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.UnassignedCells.InputArg='Unassign
edCells';
assign_cell_to_track_function.FunctionArgs.ExcludedTracks.FunctionInstance='A
ssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.ExcludedTracks.InputArg='ExcludedT
racks';
assign_cell_to_track_function.FunctionArgs.CellsLabel.FunctionInstance='Assig
nCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.CellsLabel.InputArg='CellsLabel';
assign_cell_to_track_function.FunctionArgs.PreviousCellsLabel.FunctionInstanc
e='AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.PreviousCellsLabel.InputArg='Previ
ousCellsLabel';
assign_cell_to_track_function.FunctionArgs.ShapeParameters.FunctionInstance='
AssignCellsToTracksLoop';
```

```
assign_cell_to_track_function.FunctionArgs.ShapeParameters.InputArg='ShapePar
ameters';
assign_cell_to_track_function.FunctionArgs.CellsCentroids.FunctionInstance='A
ssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.CellsCentroids.InputArg='CellsCent
roids';
assign_cell_to_track_function.FunctionArgs.CurrentTracks.FunctionInstance='As
signCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.CurrentTracks.InputArg='CurrentTra
cks';
assign_cell_to_track_function.FunctionArgs.CheckCellPath.Value=true;
assign_cell_to_track_function.FunctionArgs.FrontParams.Value=[];
assign_cell_to_track_function.FunctionArgs.MaxSearchRadius.Value=Inf;
assign_cell_to_track_function.FunctionArgs.MinSearchRadius.Value=0;
assign_cell_to_track_function.FunctionArgs.SearchRadiusPct.Value=1.5;
assign_cell_to_track_function.FunctionArgs.TrackAssignments.FunctionInstance=
'AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.TrackAssignments.InputArg='TrackAs
signments';
assign_cell_to_track_function.FunctionArgs.MaxTrackID.FunctionInstance='Assig
nCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.MaxTrackID.InputArg='MaxTrackID';
assign_cell_to_track_function.FunctionArgs.Tracks.FunctionInstance='AssignCel
lsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.Tracks.InputArg='Tracks';
assign_cell_to_track_function.FunctionArgs.MatchingGroups.FunctionInstance='A
ssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.MatchingGroups.InputArg='MatchingG
roups';
assign_cell_to_track_function.FunctionArgs.MatchingGroupsStats.FunctionInstan
ce='AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.MatchingGroupsStats.InputArg='Matc
hingGroupsStats';
assign_cell_to_track_function.FunctionArgs.ParamsCoeffOfVariation.FunctionIns
tance='AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.ParamsCoeffOfVariation.InputArg='P
aramsCoeffOfVariation';
assign_cell_to_track_function.FunctionArgs.PreviousTracks.FunctionInstance='A
ssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.PreviousTracks.InputArg='PreviousT
racks';
assign_cell_to_track_function.FunctionArgs.TracksLayout.Value=tracks_layout;
assign_cell_to_track_function.FunctionArgs.RelevantParametersIndex.Value=...
    [true true true false true false true true false];
assign_cell_to_track_function.FunctionArgs.NrParamsForSureMatch.Value=TrackSt
ruct.NrParamsForSureMatch;
assign_cell_to_track_function.FunctionArgs.DefaultParamWeights.Value=TrackStr
uct.DefaultParamWeights;
assign_cell_to_track_function.FunctionArgs.UnknownParamWeights.Value=TrackStr
uct.UnknownParamWeights;
assign_cell_to_track_function.FunctionArgs.DistanceRankingOrder.Value=TrackSt
ruct.DistanceRankingOrder;
assign_cell_to_track_function.FunctionArgs.DirectionRankingOrder.Value=TrackS
truct.DirectionRankingOrder;
assign_cell_to_track_function.FunctionArgs.UnknownRankingOrder.Value=TrackStr
uct.UnknownRankingOrder;
```

```
assign_cell_to_track_function.FunctionArgs.MinSecondDistance.Value=TrackStruc
t.MinSecondDistance;
assign_cell_to_track_function.FunctionArgs.MaxDistRatio.Value=TrackStruct.Max
DistRatio;
assign_cell_to_track_function.FunctionArgs.MaxAngleDiff.Value=TrackStruct.Max
AngleDiff;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_fu
nctions,assign_cell_to_track_function);
…
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance='Ma
keUnassignedCellsList';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg='Unassigne
dCellsIDs';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance2='A
ssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg2='Unassign
edIDs';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.FunctionInstance='Mak
eExcludedTracksList';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTr
acks';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.FunctionInstance2='As
signCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.FunctionInstance2='
AssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.OutputArg2='TrackAs
signments';
assign_cells_to_tracks_loop.FunctionArgs.MatchingGroups.FunctionInstance2='As
signCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.MatchingGroups.OutputArg2='MatchingG
roups';
…
set_group_index_function.FunctionArgs.GroupIndex.FunctionInstance='AssignCell
ToTrackUsingAll';
set_group_index_function.FunctionArgs.GroupIndex.OutputArg='GroupIndex';
```

## assignCellToTrackUsingNN

### Usage
This module tracks objects in a time-lapse sequence of label matrices using a nearest-neighbor algorithm.

### Input Structure Members
*CellsCentroids* – Current object centroids.
*CurrentTracks* – Matrix containing the track assignments for the objects in the previous frame.
*MaxTrackID* – Current maximum track ID.
*TrackAssignments* – List of track assignments that have already been completed.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.
*UnassignedCells* – List of object IDs currently unassigned.

### Output Structure Members

*ExcludedTracks* – Returns empty value. Included for compatibility only.
*GroupIndex* – Returns zero. Included for compatibility only.
*MatchingGroups* – Returns empty value. Included for compatibility only.
*TrackAssignments* – List of track assignments that have already been completed.
*UnassignedIDs* – List of object IDs currently unassigned.

### Example

```
assign_cell_to_track_function.InstanceName='AssignCellToTrackUsingNN';
assign_cell_to_track_function.FunctionHandle=@assignCellToTrackUsingNN;
assign_cell_to_track_function.FunctionArgs.UnassignedCells.FunctionInstance='
AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.UnassignedCells.InputArg='Unassign
edCells';
assign_cell_to_track_function.FunctionArgs.CellsCentroids.FunctionInstance='A
ssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.CellsCentroids.InputArg='CellsCent
roids';
assign_cell_to_track_function.FunctionArgs.CurrentTracks.FunctionInstance='As
signCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.CurrentTracks.InputArg='CurrentTra
cks';
assign_cell_to_track_function.FunctionArgs.TrackAssignments.FunctionInstance=
'AssignCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.TrackAssignments.InputArg='TrackAs
signments';
assign_cell_to_track_function.FunctionArgs.MaxTrackID.FunctionInstance='Assig
nCellsToTracksLoop';
assign_cell_to_track_function.FunctionArgs.MaxTrackID.InputArg='MaxTrackID';
assign_cell_to_track_function.FunctionArgs.TracksLayout.Value=tracks_layout;
assign_cells_to_tracks_functions=addToFunctionChain(assign_cells_to_tracks_fu
nctions,assign_cell_to_track_function);
…
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance='Ma
keUnassignedCellsList';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg='Unassigne
dCellsIDs';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.FunctionInstance2='A
ssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.UnassignedCells.OutputArg2='Unassign
edIDs';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.FunctionInstance2='
AssignCellToTrackUsingAll';
assign_cells_to_tracks_loop.FunctionArgs.TrackAssignments.OutputArg2='TrackAs
signments';
```

## clearSmallObjects

### Usage

This module is used to remove objects below a certain area from a binary image.

### Input Structure Members

*Image* – Binary image from which objects will be removed.
*MinObjectArea* – Objects with an area smaller than this value will be removed.

### Output Structure Members

*Image* – Filtered binary image.

### Example

```
clear_small_objects_function.InstanceName='ClearSmallObjects';
clear_small_objects_function.FunctionHandle=@clearSmallObjects;
clear_small_objects_function.FunctionArgs.Image.FunctionInstance='FillHolesIm
age';
clear_small_objects_function.FunctionArgs.Image.OutputArg='Image';
clear_small_objects_function.FunctionArgs.MinObjectArea.Value=30;
functions_list=addToFunctionChain(functions_list,clear_small_objects_function
);
…
display_thresholded_image_function.FunctionArgs.Image.FunctionInstance='Clear
SmallObjects';
display_thresholded_image_function.FunctionArgs.Image.OutputArg='Image';
```

## combineImages

### Usage

This module is used to combine two binary images using logical operations.

### Input Structure Members

*CombineOperation* – String value indicating the logical operation used to combine the images. Currently, only 'AND' and 'OR' are supported.
*Image1* – First binary image.
*Image2* – Second binary image.

### Output Structure Members

*Image* – Binary image resulting from the logical operation.

### Example

```
combine_nucl_plus_cyto_function.InstanceName='CombineNuclearAndCytoplasmImage
s';
combine_nucl_plus_cyto_function.FunctionHandle=@combineImages;
combine_nucl_plus_cyto_function.FunctionArgs.Image1.FunctionInstance='ClearSm
allNuclei';
combine_nucl_plus_cyto_function.FunctionArgs.Image1.OutputArg='Image';
combine_nucl_plus_cyto_function.FunctionArgs.Image2.FunctionInstance='ClearSm
allCells';
combine_nucl_plus_cyto_function.FunctionArgs.Image2.OutputArg='Image';
combine_nucl_plus_cyto_function.FunctionArgs.CombineOperation.Value='AND';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,combin
e_nucl_plus_cyto_function);
…
```

```
reconstruct_cyto_function.FunctionArgs.GuideImage.FunctionInstance='CombineNu
clearAndCytoplasmImages';
reconstruct_cyto_function.FunctionArgs.GuideImage.OutputArg='Image';
```

## compareValues

### Usage

This module is used to compare two numerical values using the specified operation.

### Input Structure Members

*Arg1* – First numerical value.
*Arg2* – Second numerical value.
*Operation* – String representing the mathematical operation to be performed. Currently, '>','<','>=','<=' and '==' are supported.

### Output Structure Members

*BooleanOut* – The result of the operation. Can be either 1 (true) or 0 (false).

### Example

```
label_to_bw_function.InstanceName='LabelToBW';
label_to_bw_function.FunctionHandle=@compareValues;
label_to_bw_function.FunctionArgs.Operation.Value='>';
label_to_bw_function.FunctionArgs.Arg1.FunctionInstance='ReviewSegmentation';
label_to_bw_function.FunctionArgs.Arg1.OutputArg='LabelMatrix';
label_to_bw_function.FunctionArgs.Arg2.Value=0;
functions_list=addToFunctionChain(functions_list,label_to_bw_function);
…

save_bw_image_function.FunctionArgs.SaveData.FunctionInstance='LabelToBW';
save_bw_image_function.FunctionArgs.SaveData.OutputArg='BooleanOut';
```

## concatenateText

### Usage

This module is used to concatenate all the variables provided in the input structure. All the variables need to be strings.

### Input Structure Members

Any name and any number of text variables may be used in the input structure.

### Output Structure Members

*Text* – The string resulting from the concatenation of all the input arguments.

### Example

```
make_label_file_name_function.InstanceName='MakeLabelFileName';
make_label_file_name_function.FunctionHandle=@concatenateText;
make_label_file_name_function.FunctionArgs.DirName.FunctionInstance='GetFileI
nfo';
```

```
make_label_file_name_function.FunctionArgs.DirName.OutputArg='DirName';
make_label_file_name_function.FunctionArgs.FileName.FunctionInstance='GetFile
Info';
make_label_file_name_function.FunctionArgs.FileName.OutputArg='FileName';
make_label_file_name_function.FunctionArgs.FileExt.Value='.mat';
functions_list=addToFunctionChain(functions_list,make_label_file_name_functio
n);
…

save_label_function.FunctionArgs.FileName.FunctionInstance='MakeLabelFileName
';
save_label_function.FunctionArgs.FileName.OutputArg='Text';
```

## cropImageWithOffset

### Usage

This module is used to crop an image at a specified point to the specified size.

### Input Structure Members

*CropSize* – Two number vector containing the desired dimensions for the cropped image.
*Image* – The image to be processed.
*XYOffset* – Offset point (from the center of the original image) at which the cropped image should be centered.

### Output Structure Members

*Image* – Cropped image.

### Example

```
crop_image.InstanceName='CropImage';
crop_image.FunctionHandle=@cropImageWithOffset;
crop_image.FunctionArgs.Image.FunctionInstance='ReadImagesInProcessingLoop';
crop_image.FunctionArgs.Image.OutputArg='Image';
crop_image.FunctionArgs.XYOffset.FunctionInstance='GetCurrentOffset';
crop_image.FunctionArgs.XYOffset.OutputArg='ArrayVal';
crop_image.FunctionArgs.CropSize.FunctionInstance='IfFirstFrame';
crop_image.FunctionArgs.CropSize.OutputArg='CropSize';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,crop_i
mage);
…

save_image.FunctionArgs.Image.FunctionInstance='CropImage';
save_image.FunctionArgs.Image.OutputArg='Image';
```

## detectMergeCandidatesUsingDistance

### Usage

This module is used to detect tracks that are never further than a small distance apart for possible merging.

### Input Structure Members

*MaxMergeDistance* – Maximum distance between tracks. Any tracks that drift further apart than this distance at any time point will be merged.
*TrackIDs* – Track IDs to be tested for merging.
*Tracks* – Tracks matrix including time stamps and object centroids.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

### Output Structure Members

*TracksToBeMerged* – Paired list of the track IDs to be merged.

### Example

```
detect_merge_candidates_function.InstanceName='DetectMergeCandidates';
detect_merge_candidates_function.FunctionHandle=@detectMergeCandidatesUsingDi
stance;
detect_merge_candidates_function.FunctionArgs.MaxMergeDistance.Value=TrackStr
uct.MaxMergeDist;
detect_merge_candidates_function.FunctionArgs.TrackIDs.FunctionInstance='GetT
rackIDs';
detect_merge_candidates_function.FunctionArgs.TrackIDs.OutputArg='TrackIDs';
detect_merge_candidates_function.FunctionArgs.Tracks.FunctionInstance='Segmen
tationLoop';
detect_merge_candidates_function.FunctionArgs.Tracks.OutputArg='Tracks';
detect_merge_candidates_function.FunctionArgs.TracksLayout.Value=tracks_layou
t;
functions_list=addToFunctionChain(functions_list,detect_merge_candidates_func
tion);
…

merge_tracks_function.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectM
ergeCandidates';
merge_tracks_function.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerg
ed';
```

## detectMitoticEvents

### Usage

This module is used to detect mitotic events in time-lapse movies of cells stained with a nuclear stain.

### Input Structure Members

*MaxSplitArea* – Maximum area a nucleus may have at the time it splits.
*MaxSplitDistance* – Nuclei that are further apart than this distance will not be considered as a potential split pair.
*MaxSplitEccentricity* – Any nucleus with an eccentricity above this value will not be considered a split candidate.
*MinSplitEccentricity* – Any nucleus with an eccentricity below this value will not be considered a split candidate.
*MinTimeForSplit* – A cell needs to have a lifespan above this value to be considered a split candidate.

*Tracks* – Tracks matrix including time stamps and object centroids.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.
*UntestedIDs* - Track IDs to be tested for mitosis.

### Output Structure Members

*SplitCells* – List of mitotic cell pairs.

### Example

```
detect_mitotic_events_function.InstanceName='DetectMitoticEvents';
detect_mitotic_events_function.FunctionHandle=@detectMitoticEvents;
detect_mitotic_events_function.FunctionArgs.Tracks.FunctionInstance='MergeTra
cks';
detect_mitotic_events_function.FunctionArgs.Tracks.OutputArg='Tracks';
detect_mitotic_events_function.FunctionArgs.UntestedIDs.FunctionInstance='Mak
eAncestryForFirstFrameCells';
detect_mitotic_events_function.FunctionArgs.UntestedIDs.OutputArg='UntestedID
s';
detect_mitotic_events_function.FunctionArgs.TracksLayout.Value=tracks_layout;
detect_mitotic_events_function.FunctionArgs.MaxSplitArea.Value=TrackStruct.Ma
xSplitArea;
detect_mitotic_events_function.FunctionArgs.MinSplitEccentricity.Value=TrackS
truct.MinSplitEcc;
detect_mitotic_events_function.FunctionArgs.MaxSplitEccentricity.Value=TrackS
truct.MaxSplitEcc;
detect_mitotic_events_function.FunctionArgs.MaxSplitDistance.Value=TrackStruc
t.MaxSplitDist;
detect_mitotic_events_function.FunctionArgs.MinTimeForSplit.Value=TrackStruct
.MinTimeForSplit;
functions_list=addToFunctionChain(functions_list,detect_mitotic_events_functi
on);
…
make_ancestry_for_cells_entering_frames_function.FunctionArgs.SplitCells.Func
tionInstance='DetectMitoticEvents';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.SplitCells.Outp
utArg='SplitCells';
```

## displayAncestryData

### Usage

This module is used to overlay cell outlines (using different colors to indicate different cell generations) and cell labels on the original images after tracking and save the resulting image.

### Input Structure Members

*AncestryLayout* – Matrix describing the order of the columns in the tracks matrix.
*CellsAncestry* – Matrix containing the ancestry records for the cells in the image.
*CellsLabel* – The label matrix containing the cell outlines for the current image.
*ColorMap* – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used. Afterwards, the colors in the map are recycled.
*CurFrame* – Integer containing the current frame number.
*CurrentTracks* – The list of the tracks for the current image.

*DS* – The directory separator to be used when generating file names (“\” for Windows, “/” for Unix/Linux).
*Image* – The original image which will be used to generate the image with overlayed outlines and labels.
*ImageFileName* – The root of the image file name to be used when generating the image file name for the current image in combination with the current frame number.
*NumberFormat* – A string indicating the number format of the file name to be used when saving the overlayed image.
*ProlDir* – Output directory where the resulting image will be saved.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

### Output Structure Members

None.

### Example

```
display_ancestry_function.InstanceName='DisplayAncestry';
display_ancestry_function.FunctionHandle=@displayAncestryData;
display_ancestry_function.FunctionArgs.Image.FunctionInstance='ReadImagesInOv
erlayLoop';
display_ancestry_function.FunctionArgs.Image.OutputArg='Image';
display_ancestry_function.FunctionArgs.CurrentTracks.FunctionInstance='GetCur
rentTracks2';
display_ancestry_function.FunctionArgs.CurrentTracks.OutputArg='Tracks';
display_ancestry_function.FunctionArgs.CellsLabel.FunctionInstance='LoadCells
Label';
display_ancestry_function.FunctionArgs.CellsLabel.OutputArg='cells_lbl';
display_ancestry_function.FunctionArgs.CellsAncestry.FunctionInstance='ImageO
verlayLoop';
display_ancestry_function.FunctionArgs.CellsAncestry.InputArg='CellsAncestry'
;
display_ancestry_function.FunctionArgs.CurFrame.FunctionInstance='ImageOverla
yLoop';
display_ancestry_function.FunctionArgs.CurFrame.OutputArg='LoopCounter';
display_ancestry_function.FunctionArgs.ColorMap.FunctionInstance='LoadColorma
p';
display_ancestry_function.FunctionArgs.ColorMap.OutputArg='cmap';
display_ancestry_function.FunctionArgs.NumberFormat.Value=TrackStruct.NumberF
ormat;
display_ancestry_function.FunctionArgs.TracksLayout.Value=tracks_layout;
display_ancestry_function.FunctionArgs.ProlDir.Value=TrackStruct.ProlDir;
display_ancestry_function.FunctionArgs.ImageFileName.Value=TrackStruct.ImageF
ileName;
display_ancestry_function.FunctionArgs.DS.Value=ds;
display_ancestry_function.FunctionArgs.AncestryLayout.Value=ancestry_layout;
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,
display_ancestry_function);
```

## displayTracksData

### Usage

This module is used to overlay cell outlines (using different colors to indicate different cell generations) and cell labels on the original images after tracking and save the resulting image.

### Input Structure Members

*CellsLabel* – The label matrix containing the cell outlines for the current image.
*CurFrame* – Integer containing the current frame number.
*CurrentTracks* – The list of the tracks for the current image.
*FileRoot* – The root of the image file name to be used when generating the image file name for the current image in combination with the current frame number.
*Image* – The original image which will be used to generate the image with overlayed outlines and labels.
*NumberFormat* – A string indicating the number format of the file name to be used when saving the overlayed image.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

### Output Structure Members

None.

### Example

```
display_tracks_function.InstanceName='DisplayTracks';
display_tracks_function.FunctionHandle=@displayTracksData;
display_tracks_function.FunctionArgs.Image.FunctionInstance='ReadImagesInSegm
entationLoop';
display_tracks_function.FunctionArgs.Image.OutputArg='Image';
display_tracks_function.FunctionArgs.CellsLabel.FunctionInstance='ResizeCytoL
abel';
display_tracks_function.FunctionArgs.CellsLabel.OutputArg='Image';
display_tracks_function.FunctionArgs.CurrentTracks.FunctionInstance='IfIsEmpt
yPreviousCellsLabel';
display_tracks_function.FunctionArgs.CurrentTracks.OutputArg='NewTracks';
display_tracks_function.FunctionArgs.CurFrame.FunctionInstance='SegmentationL
oop';
display_tracks_function.FunctionArgs.CurFrame.OutputArg='LoopCounter';
display_tracks_function.FunctionArgs.TracksLayout.Value=tracks_layout;
display_tracks_function.FunctionArgs.FileRoot.Value=[track_dir ds
TrackStruct.ImageFileName];
display_tracks_function.FunctionArgs.NumberFormat.Value=TrackStruct.NumberFor
mat;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,displa
y_tracks_function);
```

## displayVariable

### Usage

This module is used to display a variable name and its value.

### Input Structure Members

*Variable* – The variable whose value is to be displayed.
*VariableName* – The variable name to be displayed along with the variable value.

### Output Structure Members

None.

### Example

```
display_trackstruct_function.InstanceName='DisplayTrackStruct';
display_trackstruct_function.FunctionHandle=@displayVariable;
display_trackstruct_function.FunctionArgs.Variable.Value=TrackStruct;
display_trackstruct_function.FunctionArgs.VariableName.Value='TrackStruct';
functions_list=addToFunctionChain(functions_list,display_trackstruct_function
);
```

## distanceWatershed

### Usage

This module is used to compute the distance watershed of a binary image.

### Input Structure Members

*Image* – The binary image for which the distance watershed will be computed.
*MedianFilterNhood* – The size of the median filter which will be used to smooth the watershed.

### Output Structure Members

*LabelMatrix* – The result of the distance watershed.

### Example

```
distance_watershed_function.InstanceName='DistanceWatershed';
distance_watershed_function.FunctionHandle=@distanceWatershed;
distance_watershed_function.FunctionArgs.Image.FunctionInstance='ClearSmallOb
jects';
distance_watershed_function.FunctionArgs.Image.OutputArg='Image';
distance_watershed_function.FunctionArgs.MedianFilterNhood.Value=3;
functions_list=addToFunctionChain(functions_list,distance_watershed_function)
;
```

## eccentricityFilterLabel

### Usage

This module is used to remove objects below and/or above a certain eccentricity from a label matrix.

### Input Structure Members

*MaxEccentricity* – Any object with an eccentricity higher than this value will be removed.

*MinEccentricity* – Any object with an eccentricity lower than this value will be removed.

*ObjectsLabel* – The label matrix from which the objects will be removed.

### Output Structure Members

*LabelMatrix* – The filtered label matrix.

### Example

```
eccentricity_filter_function.InstanceName='EccentricityFilter';
eccentricity_filter_function.FunctionHandle=@eccentricityFilterLabel;
eccentricity_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='Segm
entObjectsUsingMarkers';
eccentricity_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix
';
eccentricity_filter_function.FunctionArgs.MaxEccentricity.Value=0.95;
…

resize_cyto_label_function.FunctionArgs.Image.FunctionInstance=’EccentricityF
ilter';

resize_cyto_label_function.FunctionArgs.Image.OutputArg='LabelMatrix';
```

## generateBinImgUsingGlobInt

### Usage

This module is used to convert a grayscale image to binary using a global intensity threshold.

### Input Structure Members

*ClearBorder* – If this value is set to *true* objects that are within *ClearBorderDist* of the image edges will be erased.
*ClearBorderDist* – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.
Pixel intensities greater than the threshold intensity are converted to 1 in the binary image.
*Image* – Grayscale image to be converted.
*IntensityThresholdPct* – This is a percentage of the intensity range of the image. The threshold intensity value is calculated as *IntensityThresholdPct*double(max_pixel-min_pixel)+min_pixel.*

### Output Structure Members

*Image* – Resulting binary image.

### Example

```
cyto_global_int_filter_function.InstanceName='CytoGlobalBrightnessIntensityFi
lter';
cyto_global_int_filter_function.FunctionHandle=@generateBinImgUsingGlobInt;
cyto_global_int_filter_function.FunctionArgs.Image.FunctionInstance='ResizeIm
age';
cyto_global_int_filter_function.FunctionArgs.Image.OutputArg='Image';
cyto_global_int_filter_function.FunctionArgs.IntensityThresholdPct.Value=0.1;
cyto_global_int_filter_function.FunctionArgs.ClearBorder.Value=true;
cyto_global_int_filter_function.FunctionArgs.ClearBorderDist.Value=2;
```

```
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,
cyto_global_int_filter_function);
…
fill_holes_cyto_images_function.FunctionArgs.Image.FunctionInstance='CytoGlob
alBrightnessIntensityFilter';
fill_holes_cyto_images_function.FunctionArgs.Image.OutputArg='Image';
```

## generateBinImgUsingGradient

### Usage

This module is used to convert a grayscale image to a binary image using values of the image gradient.

### Input Structure Members

*ClearBorder* – If this value is set to *true* objects that are within *ClearBorderDist* of the image edges will be erased.
*ClearBorderDist* – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.
*GradientThreshold* – Areas in the gradient image where the gradient is higher than this value will be set to 1 in the binary image.
*Image* – Grayscale image to be converted.

### Output Structure Members

*Image* – Resulting binary image.

### Example

```
cyto_local_avg_filter_function.InstanceName='CytoGradientLocalFilter';
cyto_local_avg_filter_function.FunctionHandle=@generateBinImgUsingGradient;
cyto_local_avg_filter_function.FunctionArgs.Image.FunctionInstance='ResizeIma
ge';
cyto_local_avg_filter_function.FunctionArgs.Image.OutputArg='Image';
cyto_local_avg_filter_function.FunctionArgs.GradientThreshold.Value=1500;
cyto_local_avg_filter_function.FunctionArgs.ClearBorder.Value=true;
cyto_local_avg_filter_function.FunctionArgs.ClearBorderDist.Value=2;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,cyto_l
ocal_avg_filter_function);
…
fill_holes_cyto_images_function.FunctionArgs.Image.FunctionInstance='CytoGrad
ientLocalFilter';
fill_holes_cyto_images_function.FunctionArgs.Image.OutputArg='Image';
```

## generateBinImgUsingLocAvg

### Usage

This module is used to convert a grayscale image to a binary image using local average values.

### Input Structure Members

*BrightnessThresholdPct* – This value indicates what percentage of the local average value will be used to threshold the image. If the pixel intensity is higher than this value times the local average value the corresponding pixel in the binary image will be set to one.
*ClearBorder* – If this value is set to *true* objects that are within *ClearBorderDist* of the image edges will be erased.
*ClearBorderDist* – Objects that are within this distance from the edges of the image will be erased if *ClearBorder* is set to *true*.
*Image* – Grayscale image to be converted.
*Strel* – Filter type used to generate the local average image. Currently *'circular'* is the only value supported.
*StrelSize* – Size of the local neighborhood used to calculate the average for each pixel in the image.

### Output Structure Members

*Image* – Resulting binary image.

### Example

```
local_avg_filter_function.InstanceName='LocalAveragingFilter';
local_avg_filter_function.FunctionHandle=@generateBinImgUsingLocAvg;
local_avg_filter_function.FunctionArgs.Image.FunctionInstance='NegativeImage'
;
local_avg_filter_function.FunctionArgs.Image.OutputArg='Image';
local_avg_filter_function.FunctionArgs.Strel.Value='disk';
local_avg_filter_function.FunctionArgs.StrelSize.Value=10;
local_avg_filter_function.FunctionArgs.BrightnessThresholdPct.Value=1.2;
local_avg_filter_function.FunctionArgs.ClearBorder.Value=false;
local_avg_filter_function.FunctionArgs.ClearBorderDist.Value=0;
functions_list=addToFunctionChain(functions_list,local_avg_filter_function);
…

fill_holes_function.FunctionArgs.Image.FunctionInstance='LocalAveragingFilter
';
fill_holes_function.FunctionArgs.Image.OutputArg='Image';
```

## getArrayVal

### Usage

This module returns a value or set of values from an array.

### Input Structure Members

*Array* – Array from which the value is to be extracted.
*Index* – The index of the values to be returned.

### Output Structure Members

*ArrayVal* – Set of values extracted from the array.

### Example

```
get_current_offset.InstanceName='GetCurrentOffset';
get_current_offset.FunctionHandle=@getArrayVal;
get_current_offset.FunctionArgs.Array.Value=frame_offsets;
get_current_offset.FunctionArgs.Index.FunctionInstance='ProcessingLoop';
get_current_offset.FunctionArgs.Index.OutputArg='LoopCounter';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,get_cu
rrent_offset);
…

crop_image.FunctionArgs.XYOffset.FunctionInstance='GetCurrentOffset';
crop_image.FunctionArgs.XYOffset.OutputArg='ArrayVal';
```

## getConvexObjects

### Usage

This module is used to find and return the index of convex objects in a binary image. The object outlines are simplified using a Douglas-Pecker algorithm to prevent detection of insignificant convexities.

### Input Structure Members

*ApproximationDistance* – This value represents the minimum distance between the approximated outline and the real one. Increasing this value makes the contours simpler but less like the original outlines.
*Image* – Binary image to be processed.

### Output Structure Members

*ConvexObjectsIndex* – List containing the index of the convex objects. The index of each object is based on performing a *bwlabeln* operation on the binary image.

### Example

```
get_convex_objects_function.InstanceName='GetConvexObjects';
get_convex_objects_function.FunctionHandle=@getConvexObjects;
get_convex_objects_function.FunctionArgs.Image.FunctionInstance='ClearSmallNu
clei';
get_convex_objects_function.FunctionArgs.Image.OutputArg='Image';
get_convex_objects_function.FunctionArgs.ApproximationDistance.Value=TrackStr
uct.ApproxDist;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,get_co
nvex_objects_function);
…

polygonal_assisted_watershed_function.FunctionArgs.ConvexObjectsIndex.Functio
nInstance='GetConvexObjects';
polygonal_assisted_watershed_function.FunctionArgs.ConvexObjectsIndex.OutputA
rg='ConvexObjectsIndex';
```

## getCurrentTracks

### Usage

Module to extract a subset of tracks out of the tracks matrix starting with the current frame.

### Input Structure Members

*CurFrame* – The current frame index.
*FrameStep* – How many frames to skip when reading the track subset. If every frame is to be read set this value to 1.
*MaxMissingFrames* – This value indicates if tracks not present in the current frame should be included in the track subset and if so how many frames away from the current frame a track is allowed to be and still be included in the subset. Setting this value to zero ensures that only tracks present in the current frame are included.
*OffsetFrame* – The number of frames from the current frame the subset should include. This value can be a positive or negative integer.
*TimeCol* – Index of the time column in the tracks matrix.
*TimeFrame* – The amount of time elapsed between each frame.
*TrackIDCol* – Index of the track ID column in the tracks matrix.
*Tracks* – Matrix containing the set of tracks from which the subset is to be extracted.

### Output Structure Members

*Tracks* – The subset of tracks extracted from the tracks matrix.

### Example

```
get_cur_tracks_function.InstanceName='GetCurrentTracks';
get_cur_tracks_function.FunctionHandle=@getCurrentTracks;
get_cur_tracks_function.FunctionArgs.Tracks.FunctionInstance='IfIsEmptyPrevio
usCellsLabel';
get_cur_tracks_function.FunctionArgs.Tracks.InputArg='Tracks';
get_cur_tracks_function.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
get_cur_tracks_function.FunctionArgs.CurFrame.InputArg='CurFrame';
get_cur_tracks_function.FunctionArgs.OffsetFrame.Value=-1;
get_cur_tracks_function.FunctionArgs.TimeFrame.Value=TrackStruct.TimeFrame;
get_cur_tracks_function.FunctionArgs.TimeCol.Value=tracks_layout.TimeCol;
get_cur_tracks_function.FunctionArgs.TrackIDCol.Value=tracks_layout.TrackIDCo
l;
get_cur_tracks_function.FunctionArgs.MaxMissingFrames.Value=TrackStruct.MaxFr
amesMissing;
get_cur_tracks_function.FunctionArgs.FrameStep.Value=TrackStruct.FrameStep;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,get_cur_tracks_function);
…

get_mean_displacement_function.FunctionArgs.CurrentTracks.FunctionInstance='G
etCurrentTracks';
get_mean_displacement_function.FunctionArgs.CurrentTracks.OutputArg='Tracks';
```

## getFileInfo

### Usage

This module is used to extract the file name, extension and directory from the absolute path.

### Input Structure Members

*DirSep* – Directory separator ("\" for Windows, "/" for Linux/Unix).
*PathName* – Absolute path name from which file name, extension and directory will be extracted.

### Output Structure Members

*DirName* – The extracted directory name.
*FileName* – The extracted file name.
*ExtName* – The extracted extension name.

### Example

```
get_file_function.InstanceName='GetFileInfo';
get_file_function.FunctionHandle=@getFileInfo;
get_file_function.FunctionArgs.DirSep.Value='\';
get_file_function.FunctionArgs.PathName.Value=path_name;
functions_list=addToFunctionChain(functions_list,get_file_function);
…

make_spreadsheet_file_name_function.FunctionArgs.DirName.FunctionInstance='Ge
tFileInfo';
make_spreadsheet_file_name_function.FunctionArgs.DirName.OutputArg='DirName';
```

## getMaxTrackID

### Usage

This module is used to return the current maximum track ID from a track matrix.

### Input Structure Members

*TrackIDCol* – Index of the track ID column in the tracks matrix.
*Tracks* – Matrix containing the set of tracks.

### Output Structure Members

*MaxTrackID* – The maximum track ID.

### Example

```
get_max_track_id_function.InstanceName='GetMaxTrackID';
get_max_track_id_function.FunctionHandle=@getMaxTrackID;
get_max_track_id_function.FunctionArgs.Tracks.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
get_max_track_id_function.FunctionArgs.Tracks.OutputArg='Tracks';
get_max_track_id_function.FunctionArgs.TrackIDCol.Value=tracks_layout.TrackID
Col;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,get_max_track_id_function);
…
```

```
assign_cells_to_tracks_loop.FunctionArgs.MaxTrackID.FunctionInstance='GetMaxT
rackID';
assign_cells_to_tracks_loop.FunctionArgs.MaxTrackID.OutputArg='MaxTrackID';
```

## getObjectsMeanDisplacement

### Usage

This module estimates the average displacement of the cells in the frame using the centroids from the previous and current frame.

### Input Structure Members

*Centroid1Col* – The index of the first coordinate of the object centroids in the tracks matrix.
*Centroid2Col* – The index of the second coordinate of the object centroids in the tracks matrix.
*CurrentTracks* – Set of tracks belonging to previous frame.
*ObjectCentroids* – The list of centroids in the current frame.

### Output Structure Members

*MeanDisplacement* – The estimated mean displacement of the objects in the frame.
*SDDisplacement* – The estimated standard deviation of the objects in the frame.

### Example

```
get_mean_displacement_function.InstanceName='GetCellsMeanDisplacement';
get_mean_displacement_function.FunctionHandle=@getObjectsMeanDisplacement;
get_mean_displacement_function.FunctionArgs.ObjectCentroids.FunctionInstance=
'GetShapeParameters';
get_mean_displacement_function.FunctionArgs.ObjectCentroids.OutputArg='Centro
ids';
get_mean_displacement_function.FunctionArgs.CurrentTracks.FunctionInstance='G
etCurrentTracks';
get_mean_displacement_function.FunctionArgs.CurrentTracks.OutputArg='Tracks';
get_mean_displacement_function.FunctionArgs.Centroid1Col.Value=tracks_layout.
Centroid1Col;
get_mean_displacement_function.FunctionArgs.Centroid2Col.Value=tracks_layout.
Centroid2Col;
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,get_mean_displacement_function);
…

assign_cells_to_tracks_loop.FunctionArgs.MeanDisplacement.FunctionInstance='G
etMeanDisplacement';
assign_cells_to_tracks_loop.FunctionArgs.MeanDisplacement.OutputArg='MeanDisp
lacement';
```

## getShapeParams

### Usage

This module is used to extract the shape parameters (Area,Eccentricity,etc.) of objects in a label matrix.  Mostly, a wrapper for the MATLAB *regionprops* function. Adds a blob id to the output so that objects that belong to the same blob may be identified at a later time.

### Input Structure Members
*LabelMatrix* – The label matrix from which the shape parameters will be extracted.

### Output Structure Members
*Centroids* – The centroids of the objects in the label matrix.
*ShapeParameters* – The shape parameters of the objects in the label matrix.

### Example
```
get_shape_params_function.InstanceName='GetShapeParameters';
get_shape_params_function.FunctionHandle=@getShapeParams;
get_shape_params_function.FunctionArgs.LabelMatrix.FunctionInstance='IfIsEmpt
yPreviousCellsLabel';
get_shape_params_function.FunctionArgs.LabelMatrix.InputArg='CellsLabel';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_
functions,get_shape_params_function);
…

start_tracks_function.FunctionArgs.ShapeParameters.FunctionInstance='GetShape
Parameters';
start_tracks_function.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters
';
```

## getShapeParamsWithDisconnects

### Usage
This module is used to extract the shape parameters (Area,Eccentricity,etc.) of objects in a label matrix. The difference between this module and *getShapeParams* is that *getShapeParamsWithDisconnects* supports objects which are disjoint (spread across multiple blobs) at the cost of execution speed.

### Input Structure Members
*LabelMatrix* – The label matrix from which the shape parameters will be extracted.

### Output Structure Members
*Centroids* – The centroids of the objects in the label matrix.
*ShapeParameters* – The shape parameters of the objects in the label matrix.

### Example
```
get_shape_params_function.InstanceName='GetShapeParameters';
get_shape_params_function.FunctionHandle=@ getShapeParamsWithDisconnects
;
get_shape_params_function.FunctionArgs.LabelMatrix.FunctionInstance='IfIsEmpt
yPreviousCellsLabel';
get_shape_params_function.FunctionArgs.LabelMatrix.InputArg='CellsLabel';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_
functions,get_shape_params_function);
…

start_tracks_function.FunctionArgs.ShapeParameters.FunctionInstance='GetShape
Parameters';
```

```
start_tracks_function.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters
';
```

## getTrackIDs

### Usage

This module is used to retrieve the list of track IDs from the track matrix.

### Input Structure Members

*TrackIDCol* – Index of the track ID column in the tracks matrix.

*Tracks* – Matrix containing the set of tracks from which IDs will be extracted.

### Output Structure Members

*TrackIDs* – The IDs extracted from the tracks matrix.

## imNorm

### Usage

This module is used to normalize an image so that the lowest pixel value is zero and the highest pixel value is the maximum allowed value for the specified integer class.

### Input Structure Members

*IntegerClass* – The integer class of the image.  Has to be an integer class supported by MATLAB such as 'int8' or 'uint8'.
*RawImage* – The image to be processed.

### Output Structure Members

*Image* – The normalized image.

### Example

```
normalize_image_to_16bit_function.InstanceName='NormalizeImageTo16Bit';
normalize_image_to_16bit_function.FunctionHandle=@imNorm;
normalize_image_to_16bit_function.FunctionArgs.RawImage.FunctionInstance='Rea
dImagesInSegmentationLoop';
normalize_image_to_16bit_function.FunctionArgs.RawImage.OutputArg='Image';
normalize_image_to_16bit_function.FunctionArgs.IntegerClass.Value='uint16';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,normal
ize_image_to_16bit_function);
…

resize_image_function.FunctionArgs.Image.FunctionInstance='NormalizeImageTo16
Bit';
resize_image_function.FunctionArgs.Image.OutputArg='Image';
```

## loadCellsLabel

### Usage

This module is used to load cells/nuclei labels. It looks for a variable named cells_lbl in the .mat data file.

### Input Structure Members

*MatFileName* – The name of the data file.

### Output Structure Members

*LabelMatrix* – The label matrix loaded from the file.

### Example

```
load_cells_label_function.InstanceName='LoadCellsLabel';
load_cells_label_function.FunctionHandle=@loadMatFile;
load_cells_label_function.FunctionArgs.MatFileName.FunctionInstance='MakeMatN
amesInOverlayLoop';
load_cells_label_function.FunctionArgs.MatFileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,
load_cells_label_function);
…

display_ancestry_function.FunctionArgs.CellsLabel.FunctionInstance='LoadCells
Label';
display_ancestry_function.FunctionArgs.CellsLabel.OutputArg='cells_lbl';
```

## loadMatFile

### Usage

This module is used to load all the variables in a MATLAB .mat file.

### Input Structure Members

*MatFileName* – The name of the .mat file.

### Output Structure Members

Each of the variable names and values found in the .mat file will be replicated in the output structure.

### Example

```
load_cells_label_function.InstanceName='LoadCellsLabel';
load_cells_label_function.FunctionHandle=@loadMatFile;
load_cells_label_function.FunctionArgs.MatFileName.FunctionInstance='MakeMatN
amesInOverlayLoop';
load_cells_label_function.FunctionArgs.MatFileName.OutputArg='FileName';
image_overlay_loop_functions=addToFunctionChain(image_overlay_loop_functions,
load_cells_label_function);
…
```

```
display_ancestry_function.FunctionArgs.CellsLabel.FunctionInstance='LoadCells
Label';
display_ancestry_function.FunctionArgs.CellsLabel.OutputArg='cells_lbl';
```

## makeAncestryForCellsEnteringFrames

### Usage

This module is used to add ancestry records for cells entering the field of view after the first frame (not the result of a mitotic event in the field of view).

### Input Structure Members

*CellsAncestry* – Current cell ancestry records.
*FirstFrameIDs* – The IDs of tracks starting in the first frame.
*SplitCells* – The IDs of tracks that are the result of mitosis.
*TimeCol* – The index of the time column in the tracks matrix.
*TrackIDCol* – The index of the track ID column in the tracks matrix.
*TrackIDs* – The IDs of all the tracks.
*Tracks* – The matrix containing all the tracks.

### Output Structure Members

*CellsAncestry* – The current cell ancestry record with ancestry of cells entering the field of view after the first frame appended to the end.

### Example

```
make_ancestry_for_cells_entering_frames_function.InstanceName='MakeAncestryFo
rCellsEnteringFrames';
make_ancestry_for_cells_entering_frames_function.FunctionHandle=@makeAncestry
ForCellsEnteringFrames;
make_ancestry_for_cells_entering_frames_function.FunctionArgs.SplitCells.Func
tionInstance='DetectMitoticEvents';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.SplitCells.Outp
utArg='SplitCells';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.TrackIDs.Functi
onInstance='GetTrackIDsAfterMerge';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.TrackIDs.Output
Arg='TrackIDs';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.FirstFrameIDs.F
unctionInstance='MakeAncestryForFirstFrameCells';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.FirstFrameIDs.O
utputArg='FirstFrameIDs';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.CellsAncestry.F
unctionInstance='MakeAncestryForFirstFrameCells';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.CellsAncestry.O
utputArg='CellsAncestry';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.Tracks.Function
Instance='MergeTracks';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.Tracks.OutputAr
g='Tracks';
make_ancestry_for_cells_entering_frames_function.FunctionArgs.TimeCol.Value=t
racks_layout.TimeCol;
```

```
make_ancestry_for_cells_entering_frames_function.FunctionArgs.TrackIDCol.Valu
e=tracks_layout.TrackIDCol;
functions_list=addToFunctionChain(functions_list,make_ancestry_for_cells_ente
ring_frames_function);
…

split_tracks_function.FunctionArgs.CellsAncestry.FunctionInstance='MakeAncest
ryForCellsEnteringFrames';
split_tracks_function.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
```

## makeAncestryForFirstFrameCells

### Usage

This module is used to create ancestry records for cells present in the first frame of a time-lapse movie.  These cells are generation zero.

### Input Structure Members

*TimeCol* – The index of the time column in the tracks matrix.
*TrackIDCol* – The index of the track ID column in the tracks matrix.
*TrackIDs* – The IDs of all the tracks.
*Tracks* – The matrix containing all the tracks.

### Output Structure Members

*CellsAncestry* – The ancestry records for the cells in the first frame.
*FirstFrameIDs* – IDs of cells in the first frame.
*UntestedIDs* – IDs of cells in the movie that were not present in the first frame.

### Example

```
make_ancestry_for_first_frame_cells_function.InstanceName='MakeAncestryForFir
stFrameCells';
make_ancestry_for_first_frame_cells_function.FunctionHandle=@makeAncestryForF
irstFrameCells;
make_ancestry_for_first_frame_cells_function.FunctionArgs.Tracks.FunctionInst
ance='MergeTracks';
make_ancestry_for_first_frame_cells_function.FunctionArgs.Tracks.OutputArg='T
racks';
make_ancestry_for_first_frame_cells_function.FunctionArgs.TrackIDs.FunctionIn
stance='GetTrackIDsAfterMerge';
make_ancestry_for_first_frame_cells_function.FunctionArgs.TrackIDs.OutputArg=
'TrackIDs';
make_ancestry_for_first_frame_cells_function.FunctionArgs.TimeCol.Value=track
s_layout.TimeCol;
make_ancestry_for_first_frame_cells_function.FunctionArgs.TrackIDCol.Value=tr
acks_layout.TrackIDCol;
functions_list=addToFunctionChain(functions_list,make_ancestry_for_first_fram
e_cells_function);
…

detect_mitotic_events_function.FunctionArgs.UntestedIDs.FunctionInstance='Mak
eAncestryForFirstFrameCells';
```

```
detect_mitotic_events_function.FunctionArgs.UntestedIDs.OutputArg='UntestedID
s';
```

## makeExcludedTracksList

### Usage

This module wraps its input, a list of cell IDs, in a MATLAB cell array.

### Input Structure Members

*UnassignedCellsIDs* – The list of cell IDs.

### Output Structure Members

*ExcludedTracks* – The MATLAB cell array.

### Example

```
make_excluded_tracks_list_function.InstanceName='MakeExcludedTracksList';
make_excluded_tracks_list_function.FunctionHandle=@makeExcludedTracksList;
make_excluded_tracks_list_function.FunctionArgs.UnassignedCellsIDs.FunctionIn
stance='MakeUnassignedCellsList';
make_excluded_tracks_list_function.FunctionArgs.UnassignedCellsIDs.OutputArg=
'UnassignedCellsIDs';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,make_excluded_tracks_list_function);
…

assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.FunctionInstance='Mak
eExcludedTracksList';
assign_cells_to_tracks_loop.FunctionArgs.ExcludedTracks.OutputArg='ExcludedTr
acks';
```

## makeImgFileName

### Usage

This module is used to build the filename of a frame from a time-lapse movie using the root file name, current frame number and a specified number format and file extension.

### Input Structure Members

*CurFrame* – The index of the current frame.
*FileBase* – The root of the image file name.
*FileExt* – The file extension.
*NumberFmt* – The number format to be used. See MATLAB *sprintf* documentation for format documentation.

### Output Structure Members

*FileName* – String containing the resulting file name.

### Example

```
make_file_name_function.InstanceName='MakeImageNamesInSegmentationLoop';
make_file_name_function.FunctionHandle=@makeImgFileName;
make_file_name_function.FunctionArgs.FileBase.Value=TrackStruct.ImageFileBase
;
make_file_name_function.FunctionArgs.CurFrame.FunctionInstance='SegmentationL
oop';
make_file_name_function.FunctionArgs.CurFrame.OutputArg='LoopCounter';
make_file_name_function.FunctionArgs.NumberFmt.Value=TrackStruct.NumberFormat
;
make_file_name_function.FunctionArgs.FileExt.Value=TrackStruct.ImgExt;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,make_f
ile_name_function);
…

read_image_function.FunctionArgs.ImageName.FunctionInstance='MakeImageNamesIn
SegmentationLoop';
read_image_function.FunctionArgs.ImageName.OutputArg='FileName';
```

## makeUnassignedCellsList

### Usage

This module is used to create a list of IDs for each cell centroid in a list.

### Input Structure Members

*CellsCentroids* – List of cell centroids.

### Output Structure Members

*UnassignedCellsIDs* – List of IDs.

### Example

```
make_unassigned_cells_list_function.InstanceName='MakeUnassignedCellsList';
make_unassigned_cells_list_function.FunctionHandle=@makeUnassignedCellsList;
make_unassigned_cells_list_function.FunctionArgs.CellsCentroids.FunctionInsta
nce='GetShapeParameters';
make_unassigned_cells_list_function.FunctionArgs.CellsCentroids.OutputArg='Ce
ntroids';
else_is_empty_cells_label_functions=addToFunctionChain(else_is_empty_cells_la
bel_functions,make_unassigned_cells_list_function);
…

make_excluded_tracks_list_function.FunctionArgs.UnassignedCellsIDs.FunctionIn
stance='MakeUnassignedCellsList';
make_excluded_tracks_list_function.FunctionArgs.UnassignedCellsIDs.OutputArg=
'UnassignedCellsIDs';
```

## manualSegmentationReview

### Usage

This module is used to manually correct errors in automatic segmentation. The module loads a GUI for each frame with all the available segmentation corrections.  To start the user has to select between blob correction and object correction by clicking on the “Select Blob” or “Select Object” button.  A blob is a contiguous region as defined by the background pixels and it may contain one or more objects. An object is the set of pixels with the same non-zero ID in the label matrix. An object may span multiple blobs.

In general (with the exception of the “Restore Blob” operation), an object or blob must be selected before an operation can be performed on it. Selection is performed by clicking on the object or blob of interest. A selected item is indicated by a checkerboard pattern.  If the “Select Multiple” box is checked clicking on an unselected item adds it to the selection.

The types of operations that may be performed on a blob are resegmentation, deletion and restoration. To resegment a blob one needs to indicate how many objects the new blob will contain and their approximate boundaries.  Clicking on the selected blob after the “Resegment Blob” button has been pressed indicates that the pixels at those locations belong to the first object in the blob. To move to the next object press the letter “n” on the keyboard and click within the blob to indicate rough boundaries.  The boundaries do not have to be specified precisely and a blob may be separated into two objects in as few as two clicks.  Once all the objects have been defined press the letter “d” on the keyboard and the blob will be resegmented.  During resegmentaion all the pixels in the blob are assigned to objects using a nearest-neighbor classifier based on the pixels selected by the user.  A blob may be deleted by selecting it and then clicking the “Remove Blob” button.  To restore a blob removed by mistake click on the “Restore Blob” button.  The GUI will display the “Raw Label” image which shows all the blobs present after thresholding before any removal by filters or manual deletions.  Choose the blob to restore by clicking on it.

Two types of operations can be performed on objects: joining and deletion. To join a number of objects into a single object click on the “Join Objects” button then select the objects you want to join by clicking on them. When you are done with object selection and want to join the objects press the “d” letter on the keyboard.  To delete an object select it then click on the “Remove Object” button.

Once all the corrections have been performed click on the “Save Changes & Continue” button to save your changes and move on to the next frame.

### Input Structure Members

*Image* – The microscopy image for the current *ObjectsLabel*.
*ObjectsLabel* – The label matrix containing the objects for which the automatic segmentation will be evaluated or corrected.
*PreviousLabel* – The label matrix containing objects from the previous time step.
*RawLabel* – The label matrix containing the objects before filtering.

### Output Structure Members

*LabelMatrix* – The label matrix containing manual corrections if any.

### Example

```
review_segmentation_function.InstanceName='ReviewSegmentation';
review_segmentation_function.FunctionHandle=@manualSegmentationReview;
review_segmentation_function.FunctionArgs.ObjectsLabel.FunctionInstance='Segm
entObjectsUsingMarkers';
```

```
review_segmentation_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix
';
review_segmentation_function.FunctionArgs.RawLabel.FunctionInstance='LabelObj
ects';
review_segmentation_function.FunctionArgs.RawLabel.OutputArg='LabelMatrix';
review_segmentation_function.FunctionArgs.PreviousLabel.Value=[];
review_segmentation_function.FunctionArgs.Image.FunctionInstance='NormalizeIm
ageTo16Bit';
review_segmentation_function.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,review_segmentation_function
);
…

get_region_props_function.FunctionArgs.LabelMatrix.FunctionInstance='ReviewSe
gmentation';
get_region_props_function.FunctionArgs.LabelMatrix.OutputArg='LabelMatrix';
```

## manualTrackingReview

### Usage

The manual tracking review module can be used to correct errors in automatic tracking and as a visualization module to select subsets of tracks based on speed, motility and ancestry parameters. For track correction there are six actions that can be performed: continue a track, switch tracks, delete a track, break a track, add a split and remove a split. The GUI displays population statistics at the top under the menu bar and individual cell statistics (if a cell is selected) in a text box on the right side.  Detected cell outlines and cell IDs may be displayed by checking the “Outlines” and/or “Labels” checkboxes.

Track continuation can be used when automatic tracking loses a cell it is tracking and starts a new track for that same cell. To continue the pre-existing track with the new track, select the pre-existing track. Selecting a track is done by clicking the on the cell body. After selecting the pre-existing track, click on the “Continue Track” button, navigate to a frame where the new track exists and click on the cell body again to select it. The new track is then appended to the pre-existing track. To see the updated track navigate to another frame.

Switching tracks can be used to correct errors resulting from tracks being switched from one cell to the other during automatic tracking. To switch the tracks navigate to the frame where the error occurs click on the first cell, click on the “Switch tracks” button and finally click on the second cell. To see the updated tracks navigate to another frame.

To erase a track select it then click on the “Delete Track” button.

Breaking a track splits the track in two at the current frame. The newly created track starts at the current frame and the old track ends at the frame before that. This can be used to correct complex errors by separating a track into smaller segments than using the other actions to fix the automatic tracking errors.

Errors in mitotic event detection can be corrected using the “Add Split” and “Remove Split” buttons. To correct a missed mitotic event select click on the parent cell (the cell with the older track), click on “Add Split” button then click on the second cell that is part of the mitotic event.

The older track is broken at the current frame, a new track is created and the ancestry records are updated for all three tracks.  To remove a spurious mitotic event select the track you want to use to continue the parent track, and then click on the “Remove Split” button.  The new track is merged with the parent track and the ancestry records are updated for both the parent track and the other remaining track.

The visualization part of the GUI is controlled using the “Manage Selection Layers” button. A selection layer is a transparency overlaid on the original image that highlights certain cells based on a user-defined criterion. The criterion for comparison may be an exact value (such as all cells with an area larger than 500 square pixels) or a percentage (cells with an area in the top 20%). Any combination of shape, motility and ancestry parameters may be used alone or in combination to define a layer. This allows the user to define layers that are either very broad in scope, such as all cells that are larger than average in a movie, or extremely tailored, such as selection for small, rounded, fast cells with a specific parent ID. Multiple layers may be present at one time and, due to the use of transparencies and a broad selection of layer colors, cells that are part of multiple layers can be detected. The layers are automatically updated as the user moves backward or forward through the timelapse sequence, and the resulting images themselves may be saved.

To define a selection layer click on the “Manage Selection Layers” button then click the “Add Layer” button. Type in the name of the layer, select a layer color from the dropdown box, then add a number of conditions. Conditions may be combined using “AND” and “OR” logical connectors. A condition consists of a property such as “Area” or “Cell ID” an operation (“=”,”<”,”>” are supported) and a value. The value can be either an absolute number or a percentage. Once all the conditions have been set click the “Save Layer” button, then close the selection layers GUI to apply the layer. To delete a layer click on “Manage Selection Layers” then select the layer to be removed and click on the “Remove Layer” button.

## Input Structure Members

*AncestryLayout* – Matrix describing the order of the columns in the tracks matrix.
*CellsAncestry* – Matrix containing cell ancestry records.
*ColorMap* – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used.  Afterwards, the colors in the map are recycled.
*FirstFrameIDs* – The IDs of tracks starting in the first frame.
*FrameCount* – The number of frames to track.
*FrameStep* – Read one out of every x frames when reading the image set. Default value is one meaning every frame will be read.
*ImageFileBase* – The root file name of the images in the sequence. For example, if the image names in the time-lapse sequence are “Experiment-0002_Position(8)_t001.jpg”,”Experiment-0002_Position(8)_t002.jpg”, etc., the root image file name is “Experiment-0002_Position(8)_t”.
*ImgExt* – String indicating the image file extension. Usually, “.jpg” or “.tif”.
*MaxMissingFrames* – This value indicates if tracks not present in the current frame should be included in the track subset and if so how many frames away from the current frame a track is allowed to be and still be included in the subset.
*NumberFormat* – A string indicating the number format of the file name to be used when saving the overlayed image.

*SegFileRoot* – The root of the data file name containing the segmented objects.
*StartFrame* – Integer indicating at which frame the tracking should start.
*SplitCells* – The IDs of tracks that are the result of mitosis.
*TimeCol* – The index of the time column in the tracks matrix.
*TimeFrame* – Time interval between consecutive frames.
*TrackIDCol* – The index of the track ID column in the tracks matrix.
*TrackIDs* – The IDs of all the tracks.
*Tracks* – The matrix containing all the tracks (track IDs and shape parameters for every cell at every time point).
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

## Output Structure Members

*CellsAncestry* – Matrix containing corrected cell ancestry records.
*Tracks* – The matrix containing all the corrected tracks (track IDs and shape parameters for every cell at every time point).

## Example

```
manual_tracks_review_function.InstanceName='ManualTracksReview';
manual_tracks_review_function.FunctionHandle=@manualTrackingReview;
manual_tracks_review_function.FunctionArgs.Tracks.FunctionInstance='LoadTrack
s';
manual_tracks_review_function.FunctionArgs.Tracks.OutputArg='tracks';
manual_tracks_review_function.FunctionArgs.CellsAncestry.FunctionInstance='Lo
adAncestry';
manual_tracks_review_function.FunctionArgs.CellsAncestry.OutputArg='cells_anc
estry';
manual_tracks_review_function.FunctionArgs.ColorMap.FunctionInstance='LoadCol
ormap';
manual_tracks_review_function.FunctionArgs.ColorMap.OutputArg='cmap';
manual_tracks_review_function.FunctionArgs.ImageFileBase.Value=TrackStruct.Im
ageFileBase;
manual_tracks_review_function.FunctionArgs.NumberFormat.Value=TrackStruct.Num
berFormat;
manual_tracks_review_function.FunctionArgs.ImgExt.Value=TrackStruct.ImgExt;
manual_tracks_review_function.FunctionArgs.TimeFrame.Value=TrackStruct.TimeFr
ame;
manual_tracks_review_function.FunctionArgs.TimeCol.Value=tracks_layout.TimeCo
l;
manual_tracks_review_function.FunctionArgs.TrackIDCol.Value=tracks_layout.Tra
ckIDCol;
manual_tracks_review_function.FunctionArgs.MaxMissingFrames.Value=TrackStruct
.MaxFramesMissing;
manual_tracks_review_function.FunctionArgs.FrameStep.Value=TrackStruct.FrameS
tep;
manual_tracks_review_function.FunctionArgs.TracksLayout.Value=tracks_layout;
manual_tracks_review_function.FunctionArgs.SegFileRoot.Value=TrackStruct.SegF
ileRoot;
manual_tracks_review_function.FunctionArgs.AncestryLayout.Value=ancestry_layo
ut;
manual_tracks_review_function.FunctionArgs.FrameCount.Value=TrackStruct.Frame
Count;
```

```
manual_tracks_review_function.FunctionArgs.StartFrame.Value=TrackStruct.Start
Frame;
…

save_updated_tracks_function.InstanceName='SaveUpdatedTracks';
save_updated_tracks_function.FunctionHandle=@saveTracks;
save_updated_tracks_function.FunctionArgs.Tracks.FunctionInstance='ManualTrac
ksReview';
save_updated_tracks_function.FunctionArgs.Tracks.OutputArg='Tracks';
save_updated_tracks_function.FunctionArgs.TracksFileName.Value=[TrackStruct.P
rolDir ds 'tracks.mat'];

save_ancestry_function.InstanceName='SaveAncestry';
save_ancestry_function.FunctionHandle=@saveAncestry;
save_ancestry_function.FunctionArgs.CellsAncestry.FunctionInstance='ManualTra
cksReview';
save_ancestry_function.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
save_ancestry_function.FunctionArgs.AncestryFileName.Value=[TrackStruct.ProlD
ir ds 'ancestry.mat'];
```

## mergeTracks

### Usage
This module is used to merge a list of track pairs. This module is not used to determine whether a set of tracks *should* be merged. Other modules are provided for that purpose such as *detectMergeCandidatesUsingDistance*.

### Input Structure Members
*FrameCount* – The number of frames that are being processed.
*FrameStep* – How many frames to skip when reading frames. Setting this value to one will cause all the frames to be read.
*NumberFormat* – A string indicating the number format of the file name to be used when saving the overlayed image.
*SegFileRoot* – The root of the data file name containing the segmented objects.
*StartFrame* – The first frame in the processed set.
*TimeFrame* – Time interval between consecutive frames.
*Tracks* – The current set of tracks.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.
*TracksToBeMerged* – The matrix of track IDs to be merged. Primary IDs (the IDs which will remain after the merge) are listed in the first column and secondary IDs (the IDs which will be removed after the merge) are listed in the second column of the matrix.

### Output Structure Members
*Tracks* – The new track set with the results from the merge incorporated.

### Example
```
merge_tracks_function.InstanceName='MergeTracks';
merge_tracks_function.FunctionHandle=@mergeTracks;
merge_tracks_function.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop'
;
merge_tracks_function.FunctionArgs.Tracks.OutputArg='Tracks';
merge_tracks_function.FunctionArgs.TracksToBeMerged.FunctionInstance='DetectM
ergeCandidates';
merge_tracks_function.FunctionArgs.TracksToBeMerged.OutputArg='TracksToBeMerg
ed';
merge_tracks_function.FunctionArgs.TracksLayout.Value=tracks_layout;
merge_tracks_function.FunctionArgs.FrameCount.Value=TrackStruct.FrameCount;
merge_tracks_function.FunctionArgs.StartFrame.Value=TrackStruct.StartFrame;
merge_tracks_function.FunctionArgs.TimeFrame.Value=TrackStruct.TimeFrame;
merge_tracks_function.FunctionArgs.SegFileRoot.Value=TrackStruct.SegFileRoot;
merge_tracks_function.FunctionArgs.FrameStep.Value=TrackStruct.FrameStep;
merge_tracks_function.FunctionArgs.NumberFormat.Value=TrackStruct.NumberForma
t;
functions_list=addToFunctionChain(functions_list,merge_tracks_function);
…
get_track_ids_after_merge_function.FunctionArgs.Tracks.FunctionInstance='Merg
eTracks';
get_track_ids_after_merge_function.FunctionArgs.Tracks.OutputArg='Tracks';
```

## negativeImage

### Usage
This module returns the negative of the image provided as an argument.

### Input Structure Members
*Image* – Image to be processed.

### Output Structure Members
*Image* – Negative image.

### Example
```
negative_image_function.InstanceName='NegativeImage';
negative_image_function.FunctionHandle=@negativeImage;
negative_image_function.FunctionArgs.Image.FunctionInstance='ImageToBW';
negative_image_function.FunctionArgs.Image.OutputArg='Image';
functions_list=addToFunctionChain(functions_list,negative_image_function);
…

display_negative_image_function.FunctionArgs.Image.FunctionInstance='Negative
Image';
display_negative_image_function.FunctionArgs.Image.OutputArg='Image';
```

## overlayAncestry

### Usage
This module is used to overlay the cell outlines (color-coded according to generation number) and the track ids on top of the original cell image.

### Input Structure Members

*AncestryLayout* – Matrix describing the order of the columns in the ancestry matrix.
*CurrentTracks* – The set of tracks for the current image.
*CellsLabel* – The label matrix containing the detected cell shapes for the current image.
*CellsAncestry* – Matrix containing the ancestry records for the cells in the time-lapse movie.
*ColorMap* – Color map to be used in drawing the cell outlines for each generation. Each generation will use the next color in the color map until all colors have been used. Afterwards, the colors in the map are recycled.
*Image* – This is the original cell image.
*ShowLabels* – Boolean value. If set to false the cell IDs will not be overlayed.
*ShowOutlines* – Boolean value. If set to false the cell outlines will not be overlayed.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

### Output Structure Members

*Image* – The overlayed image.

### Example

```
overlay_ancestry_args.Image.Value=img;
overlay_ancestry_args.CurrentTracks.Value=cur_tracks_struct.Tracks;
overlay_ancestry_args.CellsLabel.Value=label_struct.cells_lbl;
overlay_ancestry_args.CellsAncestry.Value=mtr_gui_struct.CellsAncestry;
overlay_ancestry_args.CurFrame.Value=frame_nr;
overlay_ancestry_args.ColorMap.Value=mtr_gui_struct.ColorMap;
overlay_ancestry_args.TracksLayout.Value=mtr_gui_struct.TracksLayout;
overlay_ancestry_args.AncestryLayout.Value=mtr_gui_struct.AncestryLayout;
overlay_ancestry_args.ShowLabels.Value=b_show_labels;
overlay_ancestry_args.ShowOutlines.Value=b_show_outlines;
overlay_ancestry_struct=overlayAncestry(overlay_ancestry_args);
```

## percentageForeground

### Usage

This module calculates the percentage of foreground pixels in a binary image.

### Input Structure Members

*Image* – Binary image for which the percentage of foreground pixels is to be calculated.

### Output Structure Members

*PercentageForeground* – The percentage of foreground pixels.

### Example

```
percentage_foreground_function.InstanceName='PercentageForeground';
percentage_foreground_function.FunctionHandle=@percentageForeground;
percentage_foreground_function.FunctionArgs.Image.FunctionInstance='LabelToBW
';
percentage_foreground_function.FunctionArgs.Image.OutputArg='BooleanOut';
functions_list=addToFunctionChain(functions_list,percentage_foreground_functi
on);
…
```

```
display_cell_coverage_function.FunctionArgs.Variable.FunctionInstance='Percen
tageForeground';
display_cell_coverage_function.FunctionArgs.Variable.OutputArg='PercentageFor
eground';
```

## polygonalAssistedWatershed

### Usage

This module is used to prevent a watershed segmentation module (can be another type of segmentation module) from splitting convex objects. The assumption is that convex objects are atomic and should not be split. This assumption works well for nuclear stains.

### Input Structure Members

*ConvexObjectsIndex* – List containing the index of convex objects. This list may be generated using the *getConvexObjects* module.
*ImageLabel* – Label matrix containing the original objects before segmentation.
*MinBlobArea* – Objects resulting from segmentation that have an area smaller than this value will be unsegmented.
*WatershedLabel* – Label matrix containing the objects after segmentation.

### Output Structure Members

*LabelMatrix* – A label matrix containing the objects segmented according to the segmentation in *WatershedLabel* with the exception of convex objects which are left unaltered.

### Example

```
polygonal_assisted_watershed_function.InstanceName='PolygonalAssistedWatershe
d';
polygonal_assisted_watershed_function.FunctionHandle=@polygonalAssistedWaters
hed;
polygonal_assisted_watershed_function.FunctionArgs.ImageLabel.FunctionInstanc
e='LabelNuclei';
polygonal_assisted_watershed_function.FunctionArgs.ImageLabel.OutputArg='Labe
lMatrix';
polygonal_assisted_watershed_function.FunctionArgs.WatershedLabel.FunctionIns
tance='DistanceWatershed';
polygonal_assisted_watershed_function.FunctionArgs.WatershedLabel.OutputArg='
LabelMatrix';
polygonal_assisted_watershed_function.FunctionArgs.ConvexObjectsIndex.Functio
nInstance='GetConvexObjects';
polygonal_assisted_watershed_function.FunctionArgs.ConvexObjectsIndex.OutputA
rg='ConvexObjectsIndex';
polygonal_assisted_watershed_function.FunctionArgs.MinBlobArea.Value=TrackStr
uct.MinNuclArea;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,polygo
nal_assisted_watershed_function);
…

segment_objects_using_markers_function.FunctionArgs.MarkersLabel.FunctionInst
ance='PolygonalAssistedWatershed';
segment_objects_using_markers_function.FunctionArgs.MarkersLabel.OutputArg='L
abelMatrix';
```

## refineSegmentation

### Usage

This module is used to retain only objects in a label matrix that are nearest to objects in another matrix.

### Input Structure Members

*CurrentLabel* – The label matrix from which objects may be removed if they don't have an object to which they are nearest in the *PreviousLabel* matrix.
*PreviousLabel* – The objects in this label will determine the objects that will be retain in the current label.

### Output Structure Members

*LabelMatrix* – The filtered label matrix.

### Example

```
refine_segmentation_function.InstanceName='RefineSegmentation';
refine_segmentation_function.FunctionHandle=@refineSegmentation;
refine_segmentation_function.FunctionArgs.CurrentLabel.FunctionInstance='Segm
entObjectsUsingMarkers';
refine_segmentation_function.FunctionArgs.CurrentLabel.OutputArg='LabelMatrix
';
refine_segmentation_function.FunctionArgs.PreviousLabel.FunctionInstance='Res
izePreviousLabel';
refine_segmentation_function.FunctionArgs.PreviousLabel.OutputArg='Image';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,refine
_segmentation_function);
…

review_segmentation_function.FunctionArgs.ObjectsLabel.FunctionInstance='Refi
neSegmentation';
review_segmentation_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix
';
```

## removeShortTracks

### Usage

This module is used to erase tracks with a lifespan shorter than a specified period of time.

### Input Structure Members

*AncestryLayout* – Matrix describing the order of the columns in the ancestry matrix.
*Tracks* – The tracks matrix to be processed.
*CellsAncestry* – Matrix containing the ancestry records for the cells in the time-lapse movie.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.
*MinLifespan* – Tracks with a lifespan shorter than this value will be erased.

### Output Structure Members

*Tracks* – The filtered tracks matrix.

### Example

```
remove_short_tracks_function.InstanceName='RemoveShortTracks';
remove_short_tracks_function.FunctionHandle=@removeShortTracks;
remove_short_tracks_function.FunctionArgs.Tracks.FunctionInstance='SplitTrack
s';
remove_short_tracks_function.FunctionArgs.Tracks.OutputArg='Tracks';
remove_short_tracks_function.FunctionArgs.CellsAncestry.FunctionInstance='Spl
itTracks';
remove_short_tracks_function.FunctionArgs.CellsAncestry.OutputArg='CellsAnces
try';
remove_short_tracks_function.FunctionArgs.TracksLayout.Value=tracks_layout;
remove_short_tracks_function.FunctionArgs.AncestryLayout.Value=ancestry_layou
t;
remove_short_tracks_function.FunctionArgs.MinLifespan.Value=30; %minutes
functions_list=addToFunctionChain(functions_list,remove_short_tracks_function
);
…
save_updated_tracks_function.FunctionArgs.Tracks.FunctionInstance='RemoveShor
tTracks';
save_updated_tracks_function.FunctionArgs.Tracks.OutputArg='Tracks';
```

## saveAncestry

### Usage

This module is used to save the cells ancestry matrix. The matrix is saved with the variable name *cells_ancestry*.

### Input Structure Members

*AncestryFileName* – The file name to which the cell ancestry matrix should be saved.
*CellsAncestry* – The matrix containing the cells ancestry records.

### Output Structure Members

None

### Example

```
save_ancestry_function.InstanceName='SaveAncestry';
save_ancestry_function.FunctionHandle=@saveAncestry;
save_ancestry_function.FunctionArgs.CellsAncestry.FunctionInstance='RemoveSho
rtTracks';
save_ancestry_function.FunctionArgs.CellsAncestry.OutputArg='CellsAncestry';
save_ancestry_function.FunctionArgs.AncestryFileName.Value=[TrackStruct.ProlD
ir ds 'ancestry.mat'];
functions_list=addToFunctionChain(functions_list,save_ancestry_function);
```

## saveAncestrySpreadsheets

### Usage

This module is used to save the tracks and ancestry records spreadsheets.

### Input Structure Members

*CellsAncestry* – Matrix containing the ancestry records for the cells in the time-lapse movie.
*ProlXlsFile* – The desired file name for the spreadsheet containing the ancestry records.
*ShapesXlsFile* – The desired file name for the spreadsheet containing the tracks and shape parameters data.
*Tracks* – The tracks matrix to be processed.
*TracksLayout* – Matrix describing the order of the columns in the tracks matrix.

### Output Structure Members

None.

### Example

```
save_ancestry_spreadsheets.InstanceName='SaveAncestrySpreadsheets';
save_ancestry_spreadsheets.FunctionHandle=@saveAncestrySpreadsheets;
save_ancestry_spreadsheets.FunctionArgs.Tracks.FunctionInstance='RemoveShortT
racks';
save_ancestry_spreadsheets.FunctionArgs.Tracks.OutputArg='Tracks';
save_ancestry_spreadsheets.FunctionArgs.CellsAncestry.FunctionInstance='Remov
eShortTracks';
save_ancestry_spreadsheets.FunctionArgs.CellsAncestry.OutputArg='CellsAncestr
y';
save_ancestry_spreadsheets.FunctionArgs.TracksLayout.Value=tracks_layout;
save_ancestry_spreadsheets.FunctionArgs.ShapesXlsFile.Value=TrackStruct.Shape
sXlsFile;
save_ancestry_spreadsheets.FunctionArgs.ProlXlsFile.Value=TrackStruct.ProlXls
File;
functions_list=addToFunctionChain(functions_list,save_ancestry_spreadsheets);
```

## saveCellsLabel

### Usage

This module is used to save a MATLAB label matrix containing cell objects.

### Input Structure Members

*CellsLabel* – The label matrix containing cell objects.
*CurFrame* – The index of the frame to which the label matrix corresponds.
*FileRoot* – String containing the root of the file name to be used when saving the label matrix.
*NumberFormat* – String indicating the number format to be used when formatting the current frame number to be concatenated to the file root string. See the MATLAB sprintf help file for example number format strings.

### Output Structure Members

*CellsLabel* – The label matrix containing cell objects.

### Example

```
save_cells_label_function.InstanceName='SaveCellsLabel';
save_cells_label_function.FunctionHandle=@saveCellsLabel;
save_cells_label_function.FunctionArgs.CellsLabel.FunctionInstance='ResizeCyt
oLabel';
save_cells_label_function.FunctionArgs.CellsLabel.OutputArg='Image';
save_cells_label_function.FunctionArgs.CurFrame.FunctionInstance='Segmentatio
nLoop';
save_cells_label_function.FunctionArgs.CurFrame.OutputArg='LoopCounter';
save_cells_label_function.FunctionArgs.FileRoot.Value=TrackStruct.SegFileRoot
;
save_cells_label_function.FunctionArgs.NumberFormat.Value=TrackStruct.NumberF
ormat;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,save_c
ells_label_function);
```

## saveRegionPropsSpreadsheets

### Usage

This module is used to save the shape parameters extracted using the *getRegionProps* wrapper module.

### Input Structure Members

*RegionProps* – The matrix containing the shape parameters.
*SpreadsheetFileName* – The desired file name for the saved file.

### Output Structure Members

None.

### Example

```
save_region_props_function.InstanceName='SaveRegionProps';
save_region_props_function.FunctionHandle=@saveRegionPropsSpreadsheets;
save_region_props_function.FunctionArgs.RegionProps.FunctionInstance='GetRegi
onProps';
save_region_props_function.FunctionArgs.RegionProps.OutputArg='RegionProps';
save_region_props_function.FunctionArgs.SpreadsheetFileName.FunctionInstance=
'MakeSpreadsheetFileName';
save_region_props_function.FunctionArgs.SpreadsheetFileName.OutputArg='Text';
functions_list=addToFunctionChain(functions_list,save_region_props_function);
```

## saveTracks

### Usage

This module is used to save the tracks matrix.

### Input Structure Members

*Tracks* – Matrix containing the tracks to be saved.
*TracksFileName* – The desired file name for the saved tracks data.

### Output Structure Members
None.

### Example
```
save_tracks_function.InstanceName='SaveTracks';
save_tracks_function.FunctionHandle=@saveTracks;
save_tracks_function.FunctionArgs.Tracks.FunctionInstance='SegmentationLoop';
save_tracks_function.FunctionArgs.Tracks.OutputArg='Tracks';
save_tracks_function.FunctionArgs.TracksFileName.Value=TrackStruct.TracksFile
;
functions_list=addToFunctionChain(functions_list,save_tracks_function);
```

## segmentObjectsUsingClusters

### Usage
This module is used to segment objects in a label matrix using hierarchical clustering.

### Input Structure Members
*ClusterDistance* – The height threshold for the cluster tree. All leaves below this value will be grouped in a cluster. See documentation for the MATLAB function *cluster* for more details.
*MinimumObjectArea* – Objects with an area smaller than this value will be unsegmented and distributed between the neighboring objects.
*ObjectsLabel* – The label matrix containing the objects to be segmented.
*ObjectReduce* – Used to reduce the size of the objects. If the objects are too large the clustering function will run out of memory. When this happens set *ObjectReduce* to a value lower than one.

### Output Structure Members
*LabelMatrix* – The label matrix containing the segmented objects.

### Example
```
segment_objects_using_clusters_function.InstanceName='SegmentObjectsUsingClus
ters';
segment_objects_using_clusters_function.FunctionHandle=@segmentObjectsUsingCl
usters;
segment_objects_using_clusters_function.FunctionArgs.ObjectsLabel.FunctionIns
tance='LabelNuclei';
segment_objects_using_clusters_function.FunctionArgs.ObjectsLabel.OutputArg='
LabelMatrix';
segment_objects_using_clusters_function.FunctionArgs.ObjectReduce.Value=Track
Struct.ObjectReduce;
segment_objects_using_clusters_function.FunctionArgs.MinimumObjectArea.Value=
TrackStruct.MinNuclArea;
segment_objects_using_clusters_function.FunctionArgs.ClusterDistance.Value=Tr
ackStruct.ClusterDist;
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmen
t_objects_using_clusters_function);
…

segment_objects_using_markers_function.FunctionArgs.MarkersLabel.FunctionInst
ance='SegmentObjectsUsingClusters';
```

```
segment_objects_using_markers_function.FunctionArgs.MarkersLabel.OutputArg='L
abelMatrix';
```

## segmentObjectsUsingMarkers

### Usage

This module is used to segment objects in a label matrix using markers from another label matrix.

### Input Structure Members

*MarkersLabel* – The label matrix containing the marker objects.
*ObjectsLabel* – The label matrix containing the objects to be segmented.

### Output Structure Members

*LabelMatrix* – Label matrix containing the segmented objects.

### Example

```
segment_objects_using_markers_function.InstanceName='SegmentObjectsUsingMarke
rs';
segment_objects_using_markers_function.FunctionHandle=@segmentObjectsUsingMar
kers;
segment_objects_using_markers_function.FunctionArgs.MarkersLabel.FunctionInst
ance='SegmentObjectsUsingClusters';
segment_objects_using_markers_function.FunctionArgs.MarkersLabel.OutputArg='L
abelMatrix';
segment_objects_using_markers_function.FunctionArgs.ObjectsLabel.FunctionInst
ance='LabelCytoplasm';
segment_objects_using_markers_function.FunctionArgs.ObjectsLabel.OutputArg='L
abelMatrix';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,segmen
t_objects_using_markers_function);
…

area_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SegmentObjec
tsUsingMarkers';
area_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
```

## setArrayVar

### Usage

This module is used to set a set of values in an array.

### Input Structure Members

*Array* – The array where the values will be entered.
*Index* – The index in the array where the values will be entered.
*Var* – The set of values that will be entered in the array.

### Output Structure Members

*Array* – The array with the new set of values.

### Example

```
set_offset_function.InstanceName='SetOffset';
set_offset_function.FunctionHandle=@setArrayVar;
set_offset_function.FunctionArgs.Array.FunctionInstance='SegmentationLoop';
set_offset_function.FunctionArgs.Array.InputArg='OffsetArray';
set_offset_function.FunctionArgs.Index.FunctionInstance='SegmentationLoop';
set_offset_function.FunctionArgs.Index.OutputArg='LoopCounter';
set_offset_function.FunctionArgs.Var.FunctionInstance='GetXYCoordinates';
set_offset_function.FunctionArgs.Var.OutputArg='XYCoords';
image_read_loop_functions=addToFunctionChain(image_read_loop_functions,set_of
fset_function);
…

image_read_loop.FunctionArgs.OffsetArray.FunctionInstance='SetOffset';
image_read_loop.FunctionArgs.OffsetArray.OutputArg='Array';
```

## showImageAndPause

### Usage

This module is used to show an image and pause execution.

### Input Structure Members

*FigureNr* – The handle number of the MATLAB figure. If it doesn’t exist it will be created.

*Image* – Matrix containing the image to be shown.

### Output Structure Members

None.

### Example

```
show_image_function.InstanceName=’ShowImage’;
show_image_function.FunctionHandle=@showImageAndPause;
show_image_function.FunctionArgs.FigureNr.Value=1;
show_image_function.FunctionArgs.Image.FunctionInstance='AreaFilter';
show_image_function.FunctionArgs.Image.OutputArg='LabelMatrix';
```

## showLabelMatrixAndPause

### Usage

This module is used to show a MATLAB label matrix and pause execution.

### Input Structure Members

*FigureNr* – The handle number of the MATLAB figure. If it doesn’t exist it will be created.

*LabelMatrix* – The label matrix to be displayed.

### Output Structure Members

None.

### Example

```
show_image_function.InstanceName=’ShowLabel’;
show_image_function.FunctionHandle=@ showLabelMatrixAndPause;
show_image_function.FunctionArgs.FigureNr.Value=1;
show_image_function.FunctionArgs. LabelMatrix.FunctionInstance='AreaFilter';
show_image_function.FunctionArgs. LabelMatrix.OutputArg='LabelMatrix';
```

## solidityFilter

### Usage
This module is used to remove objects below or above a threshold solidity from a binary image.

### Input Structure Members
*Image* – The binary image from which objects will be removed.
*MaxSolidity* – Objects whose solidity is above this value will be removed from the image.
*MinSolidity* - Objects whose solidity is below this value will be removed from the image.

### Output Structure Members
*Image* – The filtered binary image.

### Example
```
solidity_filter_function.InstanceName='SolidityFilter';
solidity_filter_function.FunctionHandle=@solidityFilterLabel;
solidity_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilt
er';
solidity_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
solidity_filter_function.FunctionArgs.MinSolidity.Value=0.69;
…

ap_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter
';
ap_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
```

## solidityFilterLabel

### Usage
This module is used to remove objects below or above a threshold solidity value from a MATLAB label matrix.

### Input Structure Members
*ObjectsLabel* – The label matrix from which objects will be removed.
*MaxSolidity* – Objects whose solidity is above this value will be removed from the image.
*MinSolidity* - Objects whose solidity is below this value will be removed from the image.

### Output Structure Members
*LabelMatrix* – The filtered label matrix.

### Example
```
solidity_filter_function.InstanceName='SolidityFilter';
solidity_filter_function.FunctionHandle=@solidityFilterLabel;
solidity_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='AreaFilt
er';
solidity_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
solidity_filter_function.FunctionArgs.MinSolidity.Value=0.69;

…

ap_filter_function.FunctionArgs.ObjectsLabel.FunctionInstance='SolidityFilter
';
ap_filter_function.FunctionArgs.ObjectsLabel.OutputArg='LabelMatrix';
```

## startTracks

### Usage

This module is used to start a tracks matrix.

### Input Structure Members

*CellsLabel* – The label matrix identifying the objects in the first frame of the time-lapse.
*CurFrame* – The index value of the current frame.
*ShapeParameters* – The shape parameters for the objects in the first frame (Area, Eccentricity, etc.).
*TimeFrame* – The time interval between consecutive frames in the time-lapse.

### Output Structure Members

*Tracks* – The new tracks matrix.

### Example

```
start_tracks_function.InstanceName='StartTracks';
start_tracks_function.FunctionHandle=@startTracks;
start_tracks_function.FunctionArgs.CellsLabel.FunctionInstance='IfIsEmptyPrev
iousCellsLabel';
start_tracks_function.FunctionArgs.CellsLabel.InputArg='CellsLabel'; %only
works for subfunctions
start_tracks_function.FunctionArgs.CurFrame.FunctionInstance='IfIsEmptyPrevio
usCellsLabel';
start_tracks_function.FunctionArgs.CurFrame.InputArg='CurFrame'; %only works
for subfunctions
start_tracks_function.FunctionArgs.TimeFrame.Value=TrackStruct.TimeFrame;
start_tracks_function.FunctionArgs.ShapeParameters.FunctionInstance='GetShape
Parameters';
start_tracks_function.FunctionArgs.ShapeParameters.OutputArg='ShapeParameters
';
if_is_empty_cells_label_functions=addToFunctionChain(if_is_empty_cells_label_
functions,start_tracks_function);
…

if_is_empty_cells_label_function.KeepValues.Tracks.FunctionInstance='StartTra
cks';
if_is_empty_cells_label_function.KeepValues.Tracks.OutputArg='Tracks';
```