

User's Guide

Konstantin Kozlov

Department of Computational biology,
Center for Advanced Studies, St.Petersburg State Polytechnical University,
St.Petersburg,
Russian Federation,
Polytechnicheskaya ul., 29
kozlov@spbcas.ru

April 28, 2010

Abstract

This guide describes the installation and usage of the ProStack image processing platform on the GNU/Linux operating system. The guide is targeted at the scientists that need to construct complex image processing procedures. It is assumed that a reader is able to run scripts from a command line, set environment variables up and understand the basics of the client-server software architecture. These terms and actions are not explained here.

Table of Contents

1	Team	8
2	License	8
3	Terms, Abbreviations and Notation	8
4	Scheduler	10
5	Installation of a binary packages	10
5.1	Installation on Fedora 11 and later	10
5.2	Installation on generic GNU/Linux	11
5.3	Installation on MS Windows and Mac OS X	11
5.4	System requirements	11
6	Usage	11
6.1	Main window	11
6.2	Graphical scenario builder	11
6.3	Constructing a new workspace	12
6.4	Work with existing workspace	16
6.5	Compiling the workflow	18
6.6	Several workspace or several datasets	19
7	AP format	19
8	List of Workflow Modules	21
8.1	align - Generate rules to correct the embryo orientation	21
8.2	andif - Anisotropic diffusion	21
8.3	apee - Correct orientation of embryo	22
8.4	apro - Filter the table	22
8.5	apron - Filter the data	23
8.6	apsc - Check the orientation of embryo	23
8.7	ar_minus - Subtract the image dimensions	24
8.8	ar_plus - Sum up the image dimensions	24
8.9	ar_x - Multiply the image dimensions by a constant	24
8.10	arcplot - Average the intensities over bins	25
8.11	avg - Average images pixel by pixel	25
8.12	avg2 - Average images pixel by pixel	25

8.13	blo2pol - Convert the coordinates to polar coordinates	26
8.14	blob - Extract the objects from the binary image	26
8.15	bolb - Draw an image of the blobs	26
8.16	bolin - Draw an image of the selected blob	27
8.17	chemar - Check the orientation of the Drosophila embryo	27
8.18	chole - Closing of holes	27
8.19	chole3d - Closing of holes	28
8.20	cmove - Move data using curl	28
8.21	cmove_i - Move data using curl	28
8.22	crop - Crop image	29
8.23	curlup - Move data using curl	29
8.24	cwtsd - Watershed transform	29
8.25	cwtsd3d - Watershed transform	30
8.26	decbsup - Blind deconvolution	30
8.27	decbsup_i - Blind deconvolution	30
8.28	decinv - Inverse filter	31
8.29	decinv_i - Inverse filter	31
8.30	decsup - Richardson-Lucy deconvolution	32
8.31	decsup_i - Richardson-Lucy deconvolution	32
8.32	dectm - Tikhonov-Miller deconvolution	33
8.33	dectm_i - Tikhonov-Miller deconvolution	33
8.34	decwiener - Wiener filter	33
8.35	decwiener_i - Wiener filter	34
8.36	despekle - Reduce speckle noise in the input image	34
8.37	despekle3d - Reduce speckle noise in the input image	35
8.38	diglsm - Distance transform	35
8.39	distance - Distance transform	35
8.40	distance3d - Distance transform	36
8.41	edge - Edge detection	36
8.42	edge3d - Edge detection	37
8.43	expand - Expand image via pixel replication	37
8.44	expand3d - Expand image via pixel replication	37
8.45	fill - Fill the holes	38
8.46	gclose - Morphological closing	38
8.47	gclose3d - Morphological closing	38
8.48	gdilation - Dilation	39
8.49	gdilation3d - Dilation	39
8.50	geometry - Crop the preset number of pixels from the image	39

8.51	geometry3d - Crop the preset number of pixels from the image	40
8.52	gerosion - Erosion	40
8.53	gerosion3d - Erosion	40
8.54	getp - Extract one plane from the stack	41
8.55	gmag - Calculate the magnitude of the gradient	41
8.56	gopen - Morphological opening	41
8.57	gopen3d - Morphological opening	41
8.58	grid - Produce an image with the cartesian grid	42
8.59	grid3d - Produce an image with the grid	42
8.60	grid3d_i - Produce an image with the grid	43
8.61	halfsizes - Print the dimensions of the image	43
8.62	heqm - Equalize histogram using mask	44
8.63	heq - Histogram equalization	44
8.64	hues - Combine color channels into one image	44
8.65	iapee - Visually correct the embryo orientation	44
8.66	invert - Invert the image	45
8.67	invert3d - Invert the image	45
8.68	lev - Extract the pixels with certain intensity	45
8.69	lheq - Local histogram equalization	46
8.70	lheq3d - Local histogram equalization	46
8.71	lv3d - Rotate the stack around the vertical axis	46
8.72	m_ar_plus - Sum up the image dimensions	47
8.73	m_ar_x - Multiply the image dimensions by a constant	47
8.74	match - Find objects matching the template	47
8.75	mask - Mask the image	48
8.76	max - Pixel by pixel maximum	48
8.77	max2 - Pixel by pixel maximum of two images	48
8.78	max3d - Pixel by pixel maximum of the images in the stack	49
8.79	maxlsm - Pixel by pixel maximum of the channels in the stack	49
8.80	mcrop - Crop an image manually	49
8.81	mcrop3d - Crop an image manually	49
8.82	median - Median filter	50
8.83	median3d - Median filter	50
8.84	minusabs - Subtract images	51
8.85	mlsreg - Moving least squares registration of images	51
8.86	movl2 - Combine two grayscale images into one color image	51
8.87	movl3 - Combine three grayscale images into one color image	52
8.88	movl6 - Combine six grayscale images into one color image	52

8.89	mpad - Pad the image with a preset number of pixels	53
8.90	mpad3d - Pad the image with a preset number of pixels	53
8.91	mslice3d - Visualize distance map	53
8.92	mslicegcp3d - Find landmarks or control points for registration	54
8.93	mul - Multiply two images	54
8.94	mul3d - Multiply two images	55
8.95	nplot - Plot 2D graph	55
8.96	nplot2 - Plot 2D graph	56
8.97	pad - Pad the image	56
8.98	pad3d - Pad the image	56
8.99	plot_sp - Print the pixel values to the text file	57
8.100	ppix - Print the pixel values along the line	57
8.101	prutik - Measure two distances visually	58
8.102	pump3d - Convert distance image into volume	58
8.103	pump3d_data - Convert distance image into volume with data	58
8.104	qu3d - Add quantitative data from another channel	58
8.105	qu3d2csv - Convert quantitative information to CSV format .	59
8.106	qu3dinit - Initialize storage for quantitative data	59
8.107	qu3dtrans - Initialize storage for quantitative data	60
8.108	qumap3d - Paint the mask with quantitative data	60
8.109	qumark3d - Paint the mask with quantitative data	61
8.110	qurelabel - Relabel objects after the registration of the mask .	61
8.111	raw - Make tiff from raw	61
8.112	rec3dbp - Morphological reconstruction	62
8.113	reconstruct - Morphological reconstruction	62
8.114	reconstruct3d - Morphological reconstruction	62
8.115	regmax - Regional maxima	63
8.116	regmin - Regional minima	63
8.117	restack3d - Remove planes from the stack	63
8.118	revcol - Reverse columns	64
8.119	revcol3d - Reverse columns	64
8.120	revrow - Reverse rows	64
8.121	revrow3d - Reverse rows	64
8.122	robel - Produce the image of round strip	65
8.123	rogri - Divide the image into sectors	65
8.124	ropol - Print the pixel values	65
8.125	ropri - Print the landmarks of the borders of the round strip .	66
8.126	rotate - Calculate the rotation angle	66

8.127rv3d - Rotate the stack around the vertical axis	66
8.128save - Save file	67
8.129setp - Set one plane in the stack	67
8.130shape - Characterize the shape	67
8.131shape3d - Characterize the shape	67
8.132shrink - Shrink image via pixel subsampling	68
8.133shrink3d - Shrink image via pixel subsampling	68
8.134solo - Extract quantitative information	68
8.135splitism - Write each color in the separate file	69
8.136splitrgb - Write each color in the separate file	69
8.137sselect - Select objects according to the shape	69
8.138sselect3d - Select objects according to the shape	70
8.139strel - Structural element	70
8.140strel3d - Structural element	71
8.141strel3dw - Structural element	71
8.142surf2vol - Draw triangulated surface	71
8.143surf3d - Produce triangulated surface	72
8.144surf3dfull - Produce triangulated surface with full control	72
8.145threshb - Threshold pixel values in each blob	73
8.146threshold - Threshold pixel values	73
8.147threshold3d - Threshold pixel values	73
8.148turn - Rotate image on given angle	74
8.149turn3d - Rotate image on given angle	74
8.150tv3d - Rotate the stack around the horizontal axis	74
8.151vavg - Average images pixel by pixel	75
8.152vmabs - Subtract images	75
8.153vmax - Pixel by pixel maximum of two images	75
8.154vrgb - Combine three grayscale images into one color image	75
8.155vrmbg - Remove background signal	76
8.156vrmbg_1 - Remove background signal	76
8.157vstrel - Convert structural element to the image	77
8.158vstrel3d - Convert structural element to the image	77
8.159vtxt - Convert the stack to text file	77
8.160vvarbc3d - Calculates the ratio of variances	77
8.161zscale3d - Scale the stack in z direction	78

List of Figures

1	A Workflow is a set of Nodes and Connections	9
2	Main window of the interface	12
3	Graphical scenario builder	13
4	The WM file.	14
5	Set the parameters, timeout and label of new node <code>strel</code> . . .	14
6	Making connection. Step 1.	14
7	Making connection. Step 1. Result.	15
8	Making connection. Step 2.	15
9	Making connection. Step 2. Result	15
10	Running workflow.	17
11	Compiling workspace.	18
12	Run workspace in the background.	19

1 Team

Konstantin N. Kozlov(kozlov@spbcas.ru), Andrei S. Pisarev(pisarev@spbcas.ru).
The leader of the project Maria G. Samsonova(samson@spbcas.ru).

2 License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

<http://www.fsf.org/licensing/licenses/gpl.html>

3 Terms, Abbreviations and Notation

Workflow – a complex scenario or pipeline of operations on some data (see Figure 1). User specifies only inputs. The output of one operation is automatically fed to downstream operations. A workflow can be represented as a Directed Acyclic Graph (DAG). Another synonym is Workspace.

WM – workflow module. The operation that is seen by a software as one command and is displayed to the user as a rectangle. It can be also called as Node or operator.

Each WM has several or zero Input and Output Ports. Input Ports are drawn as small rectangles on the left hand side of a Node and Output Ports – on the right hand side.

WM's are connected with Connections that are displayed as lines in the GUI. Each Input Port can have one and only one Connection. Each Output Port can have any or zero number of Connections.

Consequently, a Workflow is a set of Nodes and Connections. It is stored as a text file in AP format (see Sect.7). This file can be edited manually or

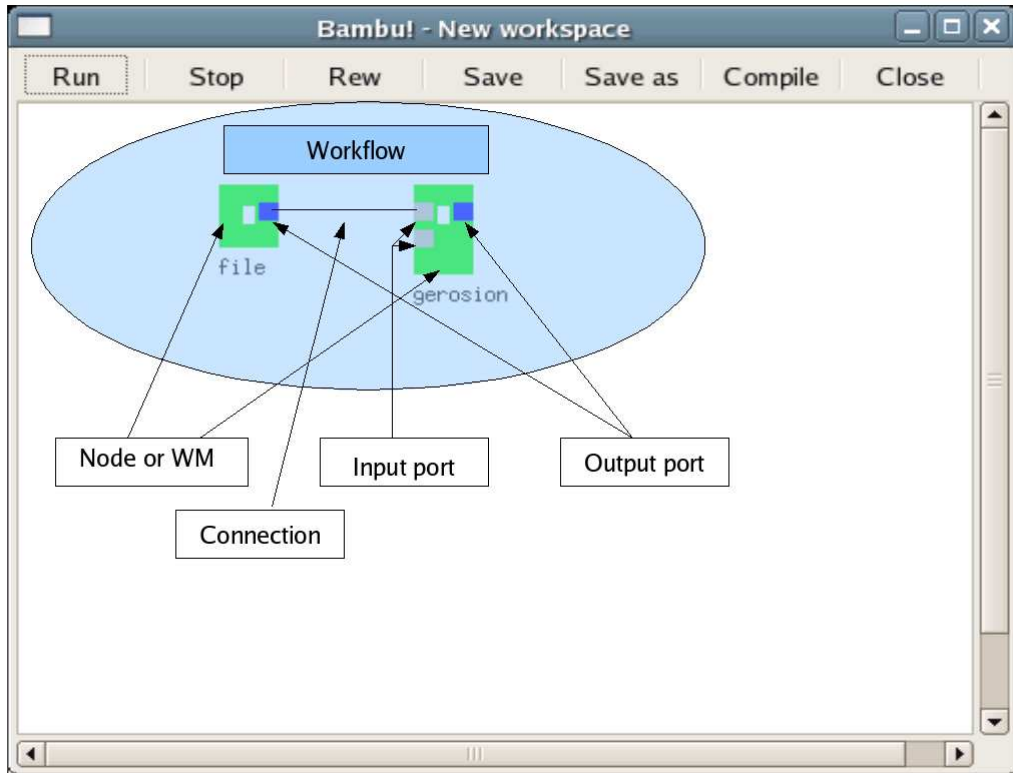


Figure 1: A Workflow is a set of Nodes and Connections

graphically with the **Graphical scenario builder** (see Sect.6.2).

In the following sections the **LEFT CLICK** denotes an event when the user presses and releases the left button of a mouse. **MIDDLE CLICK** corresponds to the middle button and **RIGHT CLICK** corresponds to the right button. If there are only two buttons on the mouse, the event of pressing the middle one is usually emulated by the simultaneous pressing of both left and right buttons. The emulation is done on the system level and thus is outside the scope of this manual. The **SHIFT+MIDDLE CLICK** denotes an event that happens when the user presses and releases the middle button of a mouse while holding the **SHIFT** key pressed. The **CTRL+MIDDLE CLICK** denotes an event that happens when the user presses and releases the middle button of a mouse while holding the **CTRL** key pressed.

4 Scheduler

The workflow is represented as a set of Nodes and Connections. Each Node has an associated operation, input and output ports, status and timeout. Timeout is defined as the maximal time that the software is allowed to spend on the requested operation. The status can be one of the following:

1. waiting
2. running
3. completed
4. failed

On loading from file or on the construction of a workflow all nodes have 'waiting' status. When the node is being executed its status is changed to 'running'. The node gets given 'completed' status if no errors occurred during its execution otherwise it gets 'failed' status. Each node is executed in a separate thread and several nodes can be simultaneously executed. A Node can be executed if its status is 'waiting' and as soon as data is available on all its input ports. Once execution is completed results are sent through output connections to their destination nodes.

5 Installation of a binary packages

The software is developed on Fedora 11 GNU/Linux and distributed as RPM packages for this OS and x86_64 architecture. The binary bundles for Microsoft Windows XP (x86) and Mac OS X 10.5 Leopard (x86) are also provided.

5.1 Installation on Fedora 11 and later

For Fedora 11 x86_64 download the latest binary RPM package and install with

```
$ yum --nogpgcheck localinstall <binary rpm>
```

For Fedora 11 x86 and later Fedora versions both x86 and x86_64 download source rpm and rebuild it. This may require installation of many development packages. If you don't want to install those packages you may jump to the next section.

5.2 Installation on generic GNU/Linux

Download the binary bundle and unpack it to any directory you have write access to. Run the software as

```
$ /path/to/your/directory/ProStack/bin/ProStack
```

5.3 Installation on MS Windows and Mac OS X

Download the binary bundle for your OS and install it the standard way. On Windows run the downloaded setup program, on Mac copy the application from the mounted disk image to the Applications folder.

5.4 System requirements

The RPM distribution should resolve all dependencies using yum. Bundles should contain everything that is not standard for the underlying platform. To reduce the download size GNU/Linux bundle doesn't contain things like Perl, Gnuplot and ImageMagick. If you do not have or do not want to install them, some not essential modules will not work.

6 Usage

6.1 Main window

The main window of the interface is presented in [Figure 2](#).

6.2 Graphical scenario builder

The window of the Graphical scenario builder is presented in [Figure 3](#).

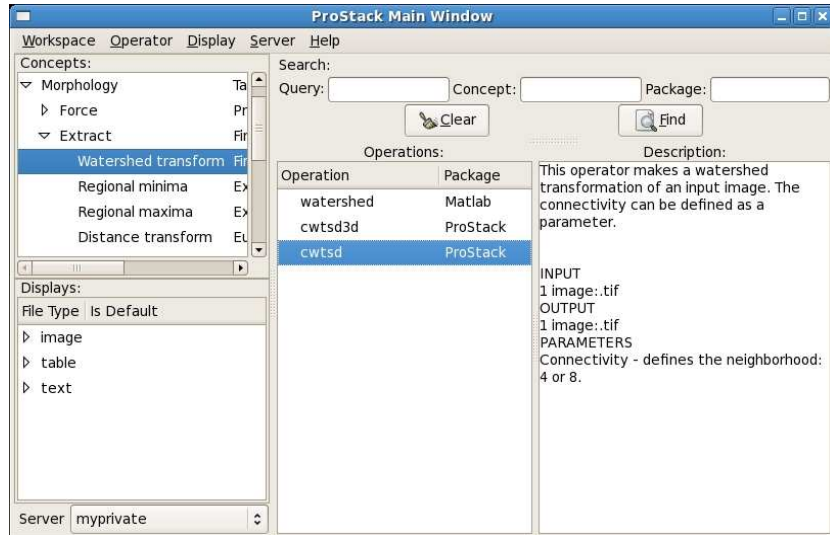


Figure 2: Main window of the interface

6.3 Constructing a new workspace

1. MIDDLE CLICK in BUILDER WINDOW places the *file* WM (Figure 4).
2. SHIFT+MIDDLE CLICK in BUILDER WINDOW places the *from_file* WM. RIGHT CLICK on the WM opens the pop-up menu. Options opens the File Selection dialog. The *from_file* WM lets the user to process data stored on local machine.
3. Other pop-up menu items:
 - (a) Copy – Copy node to clipboard,
 - (b) Paste – Paste the previously copied node at the location of mouse pointer,
 - (c) Delete – Delete node,
 - (d) Help – Open the short description of the selected WM,
 - (e) View output – If a node was executed and the mouse pointer is placed on the node output port this item will display the port output.

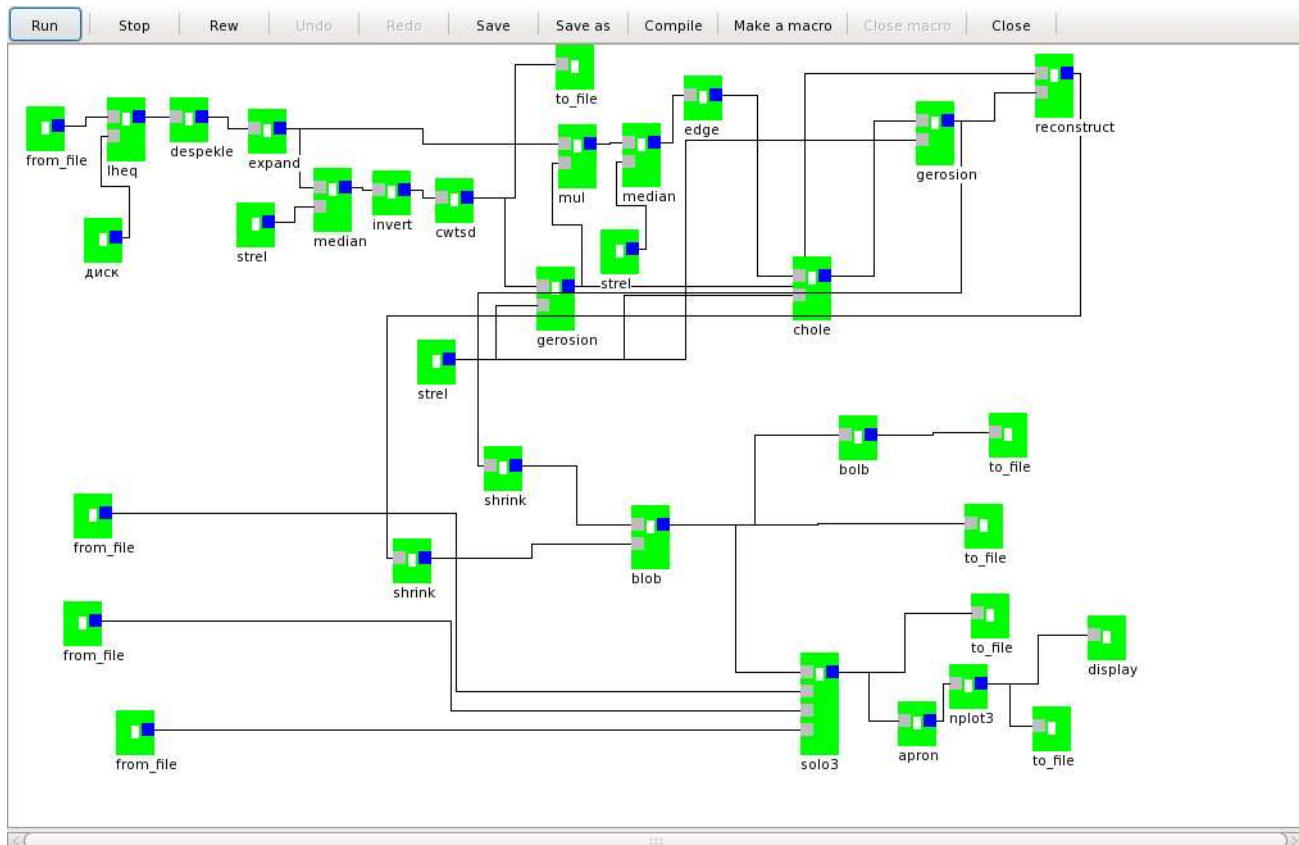


Figure 3: Graphical scenario builder

4. Select a WM from the list displayed in the MAIN WINDOW by LEFT CLICK on its name. The MIDDLE CLICK in the BUILDER WINDOW window will place the WM at the location of mouse pointer. Use **Options** item of the pop-up menu to set the parameters, timeout and label of new node. The DOUBLE CLICK opens the same dialog, e.g. for WM `strel` 5.
5. Use LEFT CLICK to select the `display` node from the list shown in the MAIN WINDOW. The MIDDLE CLICK in the BUILDER WINDOW will place this node at the location of mouse pointer. Use **Options** item of the pop-up menu to set the parameters, timeout and label of new node.
6. To connect nodes LEFT CLICK on the output port of the first node –

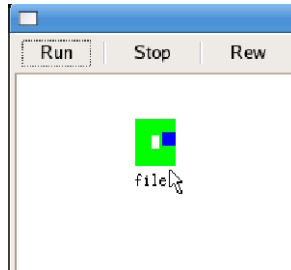


Figure 4: The WM file.

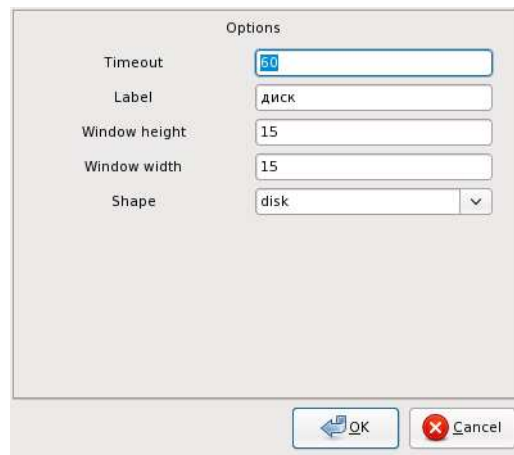


Figure 5: Set the parameters, timeout and label of new node `strel`.

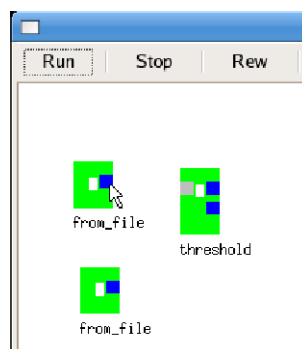


Figure 6: Making connection. Step 1.

Figure 6. The port and all outgoing connections will be colored red –

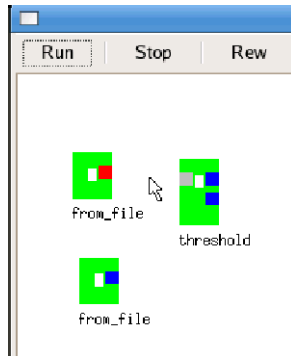


Figure 7: Making connection. Step 1. Result.

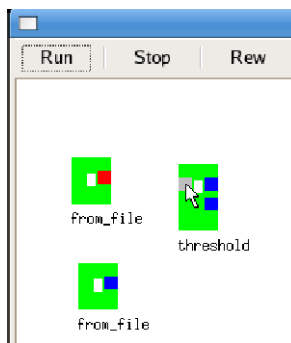


Figure 8: Making connection. Step 2.

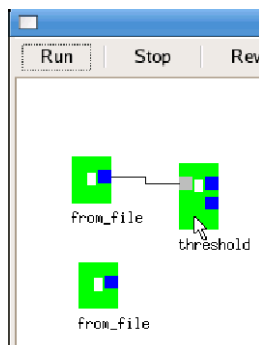


Figure 9: Making connection. Step 2. Result

Figure 7. LEFT CLICK on the input port of the target node – Figure 8.
Connection will appear – Figure 9.

7. The processing results can be saved on the local machine. To do so CTRL+MIDDLE CLICK in the BUILDER WINDOW window. This places the *to_file* node. Go to the Options menu item and specify the output file in the File Save dialog window.
8. To delete an existing connection repeat the operations listed in 9.
9. The addition/deletion of node or connection can be undone and redone by using the Undo and Redo buttons of the BUILDER WINDOW toolbar (1).
10. Other buttons of the BUILDER WINDOW toolbar:
 - (a) Save As – open the File Save dialog window to specify the location and name of the file to which workflow will be saved,
 - (b) Save – save workflow,
 - (c) Close – close application,
 - (d) Run – run workflow,
 - (e) Stop – stop the execution of workflow,
 - (f) Rew – rewind workflow to the start,
 - (g) Compile – compile workflow to WM.

6.4 Work with existing workspace

1. The execution is started by pressing the Run button of the BUILDER WINDOW toolbar, see Figure 10, where the status of the node is displayed with a color.
2. The execution stops, if the Stop button is pressed. The run will continue from the break point if the Run button is pressed again. The Rew button (rewind) clears all intermediate results.
3. If the options of a node were changed by using the Options dialog then the execution of workflow stops and all downstream nodes are cleared starting from this node. If the options of a node were changed by using the Options dialog box then the execution of workflow stops and all nodes located downstream from the modified node are cleared.

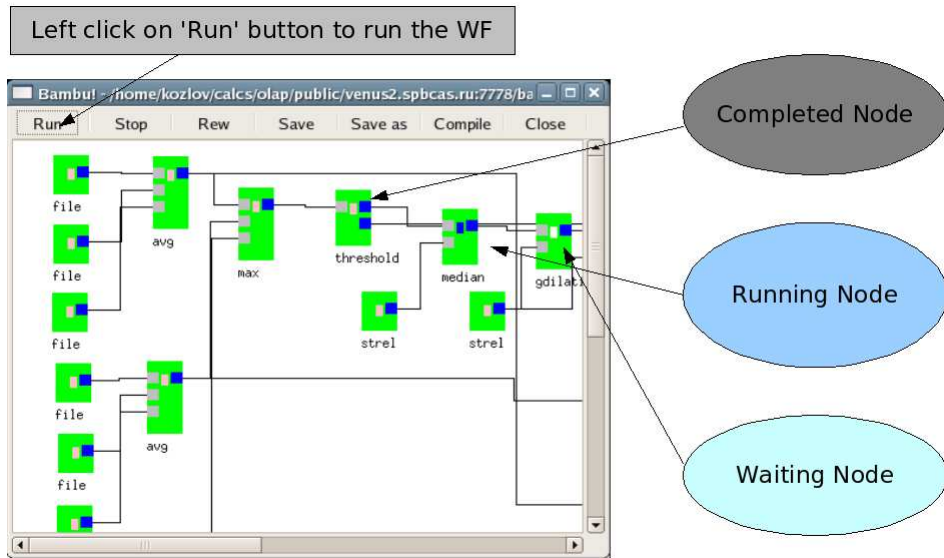


Figure 10: Running workflow.

4. If a connection is removed or changed then the execution of workflow stops and all downstream nodes are cleared starting from this connection.
5. The temporary storage is freed on exit and rewind.

6.5 Compiling the workflow

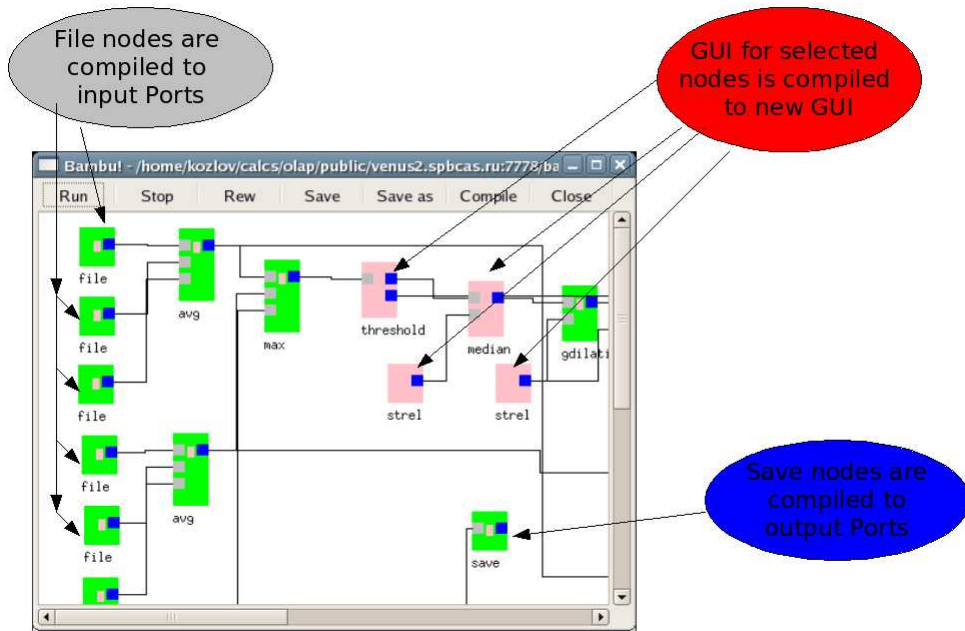


Figure 11: Compiling workspace.

The compilation is illustrated in Figure 11. Compilation results in four files placed in a new directory specified in the File selection dialog opened by pressing the **Compile** button. The name of the directory is the same as the name of the new WM.

The following files are generated:

<name>.p1 – the Perl script that serves as the WM executable and runs workflow in the **komet** interpreter,

<name>.in – the input Ports of new WM,

<name>.out – the output Ports of new WM,

<name>.ui2 – the user interface displayed under the **Options** menu in **bambu**.

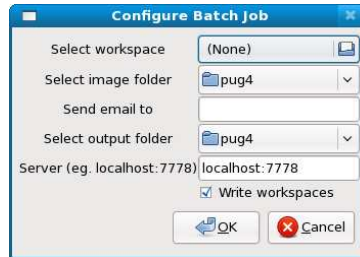


Figure 12: Run workspace in the background.

6.6 Several workspace or several datasets

To merge two workspaces use the following commands:

```
$ komet -m <workspace 1> <workspace 2> <workspace merged>
```

To run workspace in the background open the workspace and look through all *to_file* and *from_file* nodes. They should contain not the real filenames but only the "suffixes" to identify correct inputs and produce correct outputs based on the embryoname. The paths will be corrected at runtime only suffixes matter. Go to **Workspace/Apply** menu 12. Select workspace, the folder that contains input images, the output folder, the server can be left unchanged (localhost:7778) and type your email. If the check box is checked the workspaces for each embryo in the output folder with correct paths. When you click OK the separate process is started that does the job so you can close the main window. It writes logs and states to the output folder.

Another option is to generate workspaces for all embryos and then process them. To generate them to the following:

```
$ komet -w -n <email>,<output dir>,<input dir>,localhost:7778 <workspace>
```

-w - means write workspaces

-n - don't run them

7 AP format

The scenarios are stored as text files that use so called *.ini*-style. The following is a sample scenario.

8 List of Workflow Modules

Each section below describes one workflow module. The section name consists of the name of a module as it is presented in the C window of the bambu GUI and the full name of operation. The contents of the section presents the short explanation of how this module works. For detailed explanation and general information on image processing consult [2]. The number and type of input and output ports are listed under INPUT and OUTPUT entries, respectively. The parameters that can be adjusted for this operation are listed under PARAMETERS entry.

8.1 align - Generate rules to correct the embryo orientation

This operator generates the rules to put the image of Drosophila embryo into standard orientation. It checks whether the posterior part of an embryo is larger than the anterior one, and whether the ventral part is larger than the dorsal one. The output text file contains the list of needed operations.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

The format of an output file:

<align>

transpose or - if not needed

reversecolumn or - if not needed

reverserows or - if not needed

</align>

8.2 andif - Anisotropic diffusion

Diffusion algorithms remove noise from an image by modifying the image via a partial differential equation (PDE). Anisotropic diffusion lets the diffusion coefficient to vary spatially so as to enhance the intra-region smoothing in preference to the inter-region one. Region boundaries remain sharp. The details can be found in [4]. The PDE is discretized in space using the pixel locations as knots of the grid. The original image is used as the initial condition and the PDE is numerically integrated over time with the integration

("time") step and the number of iterations supplied as parameters. Two edge-stopping functions were proposed in the original paper. The additional parameter called **Threshold** controls the "sensitivity" of the edge-stopping function. The parameters are to be determined experimentally to achieve the desired result.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Number of iterations

Time step(<=1)

Threshold(<1) for edge stopping function

The type of edge stopping function

8.3 apee - Correct orientation of embryo

This operator applies the rules to put the image of Drosophila embryo into standard orientation. These rules are generated by the **align** (see 8.1) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.4 apro - Filter the table

This operator filters the input table of numerical values. The table can consist of an arbitrary number of rows greater than one. The number of columns should be greater than two. The output consists of two columns. Parameters **X** and **F** define the numbers of columns in input table that are printed in the output. The rows are selected by the following rule: $Y_{min} < Y < Y_{max}$, where **Y** represents the value of the current row in the third column of the input table. The rows of the output are sorted with respect to the first column. Consequently, the output can be directly fed to the **gnuplot**.

INPUT

1 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

X – number of the column for the argument,

F – number of the column for the function value,

Y min – lower bound for y,

Y max – upper bound for y.

8.5 apron - Filter the data

This operator filters the input table of numerical values. The table can consist of an arbitrary number of rows greater than one. The number of columns should be greater than two. The output table consists of all columns from the input. The rows are selected by the following rule: $Y_{min} < Y < Y_{max}$, where Y represents the value of the current row in the third column of the input table. The rows of the output are sorted with respect to the second column. Consequently, the output can be directly fed to the `gnuplot`.

INPUT

1 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Y min – lower bound for y,

Y max – upper bound for y.

8.6 apsc - Check the orientation of embryo

This operator checks the rules generated by `align` (see 8.1) operator using the result of `chemar` (see 8.17) operator. The resultant rules can be used to correct the orientation.

INPUT

1 text:.txt

2 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Direction – indicates in which part of an embryo (A(anterior) or P(posterior)) the level of gene expression should be higher.

8.7 ar_minus - Subtract the image dimensions

This operator subtracts the image dimensions that are given as a string of four numbers which are distances measured in pixels from the image center to its borders. These dimensions can be calculated by the `halfsizes` (see 8.61) operator.

INPUT

1 text:.txt

2 text:.txt

OUTPUT

1 text:.txt

8.8 ar_plus - Sum up the image dimensions

This operator sums up the image dimensions that are given by a string of four numbers which are the distances measured in pixels from the image center to its borders. The dimensions can be calculated by the `halfsizes` (see 8.61) operator.

INPUT

1 text:.txt

2 text:.txt

OUTPUT

1 text:.txt

8.9 ar_x - Multiply the image dimensions by a constant

This operator multiplies by a constant the image dimensions given by a string of four numbers which are the distances measured in pixels from the image center to its borders. The dimensions can be calculated by the `halfsizes` (see 8.61) operator. The constant is defined as a parameter.

INPUT

1 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Value – the multiplier.

8.10 arcplot - Average the intensities over bins

This operator averages the intensities over bins defined in polar coordinates. The input file must contain the list of pixels in polar coordinates. The output contains the table of bins with average angle (theta) and the value of average intensity in a bin.

INPUT

1 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Rho - lower bound for radius,

Rho max - upper bound for radius,

Rho bins - number of bins along the radius,

Theta min - lower bound for theta,

Theta max - upper bound for theta,

Theta bins - number of bins along the arc.

8.11 avg - Average images pixel by pixel

This operator averages three images pixel by pixel.

INPUT

1 image:.tif

2 image:.tif

3 image:.tif

OUTPUT

1 image:.tif

8.12 avg2 - Average images pixel by pixel

This operator averages two images pixel by pixel.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

8.13 blo2pol - Convert the coordinates to polar coordinates

This operator converts the coordinates of centroids of blobs to polar coordinates. The center of coordinate system is calculated from the input mask image. The list of blobs is input to the second port and can be calculated by the blob (see 8.14) operator.

INPUT

1 image:.tif - mask image

2 text:.txt

OUTPUT

1 text:.txt

8.14 blob - Extract the objects from the binary image

Contiguous regions of "on" pixels surrounded by "off" pixels (blobs) are stored in the output as the list of pixel indices. The limits are checked for the number of pixels in the region. The number of blobs is also limited. If the input images are different the first is treated as the image of domains each of which contains not more than one blob. The list of pixel indices for the domains containing blobs is also stored in the output. This makes it possible to attach the information of the surroundings to the blobs.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Minimal number of pixels

Maximal number of pixels

Maximal number of blobs

8.15 bolb - Draw an image of the blobs

This operator restores an image of blobs from the list stored in the input file.

INPUT

1 text:.txt

OUTPUT

1 image:.tif

8.16 bolin - Draw an image of the selected blob

This operator restores an image of the selected blob from the list stored in the input file. The blob is selected by the index.

INPUT

1 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Index – index of the blob.

8.17 chemar - Check the orientation of the Drosophila embryo

This operator calculates the average intensity in the left and right parts of the image.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

Format of the output file:

<chemar>

R - if the the average intensity on the left is greater and L otherwise,

"average intensity on the left" "average intensity on the right",

</chemar>

8.18 chole - Closing of holes

This operator fills the regions of "off" pixels surrounded by "on" pixels using a structural element. Unclosed contours are erased. The structural element can be calculated by the `strel` (see [8.139](#)) operator.

INPUT

1 image:.tif

2 text:.txt - structural element

OUTPUT

1 image:.tif

8.19 chole3d - Closing of holes

This operator fills the regions of "off" pixels surrounded by "on" pixels using a structural element. Unclosed contours are erased. The structural element can be calculated by the `stre13d` (see [8.140](#)) operator.

INPUT

1 image:.tif

2 text:.txt - structural element

OUTPUT

1 image:.tif

8.20 cmove - Move data using curl

This operator moves data between servers using curl utility.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Scheme – http or https.

Uri – address.

Login – username.

Password – password.

8.21 cmove_i - Move data using curl

This operator moves data between servers using curl utility.

OUTPUT

1 image:.tif

PARAMETERS

Scheme – http or https.

Uri – address.

Login – username.

Password – password.

Image – file name.

8.22 crop - Crop image

This operator reduces the dimensions of the input image to the minimal rectangular area required to cover all "on" pixels. The number of erased pixels is stored in the second output file that can be used to crop other images by the `geometry` (see 8.50) operator.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 text:.txt

The format of output file:

<crop>

upper offset, lower offset, left offset, right offset - numbers of pixels cropped at each side,

</crop>

8.23 curlup - Move data using curl

This operator moves data between servers using curl utility.

INPUT

1 image:.tif

PARAMETERS

Scheme – http or https.

Uri – address.

Name – file name.

Login – username.

Password – password.

8.24 cwtsd - Watershed transform

This operator makes a watershed transformation of an input image. The connectivity can be defined as a parameter.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Connectivity - defines the neighborhood: 4 or 8.

8.25 **cwtsd3d - Watershed transform**

This operator makes a watershed transformation of an input image. The connectivity can be defined as a parameter.

INPUT

1 **image:.tif**

OUTPUT

1 **image:.tif**

PARAMETERS

Connectivity - defines the neighborhood: 6 or 26.

8.26 **decbsup - Blind deconvolution**

This operator performs a blind deconvolution of the input image. It reads the initial guess for the PSF parameters from the second input. It also prints the PSF approximation in the second output.

INPUT

1 **image:.tif**

2 **text:.txt**

OUTPUT

1 **image:.tif**

2 **text:.txt**

PARAMETERS

Max.num.of iterations - maximal number of iterations.

Max.blind iter - maximal number of iterations for each PSF guess.

Criterion - stopping criterion.

Lambda - Laplace coefficient.

PSF type - **bessel,gauss,exp,ones** or **confocal_beessel**.

8.27 **decbsup_i - Blind deconvolution**

This operator performs a blind deconvolution of the input image. The initial guess for the PSF parameters is provided as the parameters. It also prints the PSF approximation in the second output.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 text:.txt

PARAMETERS

Max.num.of iterations - maximal number of iterations.

Max.blind iter - maximal number of iterations for each PSF guess.

Criterion - stopping criterion.

Lambda - Laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

PSF parameter - standard deviation.

Window - -1 for auto.

8.28 decinv - Inverse filter

This operator performs deconvolution of the input image using inverse filter. It reads the initial guess for the PSF parameters from the second input.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Lambda - Laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

Gamma - regularization coefficient.

8.29 decinv_i - Inverse filter

This operator performs deconvolution of the input image using inverse filter. The initial guess for the PSF parameters is provided as the parameters.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Lambda - Laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.
Gamma - regularization coefficient.
PSF parameter - standard deviation.
Window - `-1` for auto.

8.30 `decsup` - Richardson-Lucy deconvolution

This operator performs deconvolution of the input image using Richardson-Lucy algorithm. It reads the initial guess for the PSF parameters from the second input.

INPUT

1 `image:.tif`

2 `text:.txt`

OUTPUT

1 `image:.tif`

PARAMETERS

`Max.num.of iterations` - maximal number of iterations.

`Criterion` - stopping criterion.

`Lambda` - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

8.31 `decsup_i` - Richardson-Lucy deconvolution

This operator performs deconvolution of the input image using Richardson-Lucy algorithm. The initial guess for the PSF parameters is provided as the parameters.

INPUT

1 `image:.tif`

OUTPUT

1 `image:.tif`

PARAMETERS

`Max.num.of iterations` - maximal number of iterations.

`Criterion` - stopping criterion.

`Lambda` - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

PSF parameter - standard deviation.

Window - `-1` for auto.

8.32 dectm - Tikhonov-Miller deconvolution

This operator performs deconvolution of the input image using Tikhonov-Miller algorithm. It reads the initial guess for the PSF parameters from the second input.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Lambda - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

Gamma - regularization coefficient.

8.33 dectm_i - Tikhonov-Miller deconvolution

This operator performs deconvolution of the input image using Tikhonov-Miller algorithm. The initial guess for the PSF parameters is provided as the parameters.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Lambda - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

Gamma - regularization coefficient.

PSF parameter - standard deviation.

Window - -1 for auto.

8.34 decwiener - Wiener filter

This operator performs deconvolution of the input image using Wiener filter. It reads the initial guess for the PSF parameters from the second input.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Lambda - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

Gamma - regularization coefficient.

Alpha - Value 1 corresponds to the classic Wiener filter.

8.35 decwiener_i - Wiener filter

This operator performs deconvolution of the input image using Wiener filter. The initial guess for the PSF parameters is provided as the parameters.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Lambda - laplace coefficient.

PSF type - `bessel,gauss,exp,ones` or `confocal_beessel`.

Gamma - regularization coefficient.

Alpha - Value 1 corresponds to the classic Wiener filter.

PSF parameter - standard deviation.

Window - -1 for auto.

8.36 despekle - Reduce speckle noise in the input image

This operator reduces the intensity of salt and pepper noise in an image using the Crimmins complementary hulling algorithm [1]. This algorithm smoothes the image by reducing the magnitude of locally inconsistent pixels, as well as by increasing the magnitude of pixels in the neighbourhood surrounding a spike. The spike is defined here as a pixel whose value is different from its surroundings by more than 2 intensity levels. Increasing number of iterations of the algorithm can introduce an effect of blurring of the image. In the ultimate case all sharp gradients will be smoothed down to a magnitude of 2 intensity levels. case

INPUT

1 image:.tif

OUTPUT

```
1 image:.tif
PARAMETERS
Iterations - number of sweeps.
```

8.37 despekle3d - Reduce speckle noise in the input image

This operator reduces the intensity of salt and pepper noise in an image using the Crimmins complementary hulling algorithm [1]. This algorithm smoothes the image by reducing the magnitude of locally inconsistent pixels, as well as by increasing the magnitude of pixels in the neighbourhood surrounding a spike. The spike is defined here as a pixel whose value is different from its surroundings by more than 2 intensity levels. Increasing number of iterations of the algorithm can introduce an effect of blurring of the image. In the ultimate case all sharp gradients will be smoothed down to a magnitude of 2 intensity levels. case

```
INPUT
1 image:.tif
OUTPUT
1 image:.tif
PARAMETERS
Iterations - number of sweeps.
```

8.38 diglsm - Distance transform

This operator extracts one channel from the multichannel LSM image.

```
INPUT
1 image:.tif
OUTPUT
1 image:.tif
PARAMETERS
Channel - the channel to extract.
```

8.39 distance - Distance transform

This operator replaces the value of each pixel in the input image by the Euclidean distance from this pixel to the nearest "off" pixel. The Window

parameter defines the size (in pixels) of the neighbourhood of the pixel under consideration used in calculations.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Window- defines the approximation of the distance measure.

8.40 distance3d - Distance transform

This operator replaces the value of each pixel in the input image by the Euclidean distance from this pixel to the nearest "off" pixel. The Window parameter defines the size (in pixels) of the neighbourhood of the pixel under consideration used in calculations.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Window- defines the approximation of the distance measure.

8.41 edge - Edge detection

This operator detects the edges of objects in the input image using the Shen-Castan Detector [5, 6].

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

a1, a2 - parameters of the Infinite Symmetric Exponential Filter (ISEF), $0 < a1 < 1, 0 < a2 < 1$.

Low threshold - the minimal value of gradient magnitude,

High threshold - the maximal minimal value of gradient magnitude,

Window - size of window to calculate the magnitude of gradient,

Segment - minimal number of pixels in the edge segment. The width or

height of a window cannot be an even number,
Connectivity - 4 or 8-neighborhood

8.42 edge3d - Edge detection

This operator detects the edges of objects in the input image using the Shen-Castan Detector [5, 6].

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

a1, a2 - parameters of the Infinite Symmetric Exponential Filter (ISEF), $0 < a1 < 1, 0 < a2 < 1$.

Low threshold - the minimal value of gradient magnitude,

High threshold - the maximal minimal value of gradient magnitude,

Window - size of window to calculate the magnitude of gradient,

Segment - minimal number of pixels in the edge segment. The width or height of a window cannot be an even number,

Connectivity - 6 or 26-neighborhood

8.43 expand - Expand image via pixel replication

This operator increases the size of the input image using the pixel replication.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Scale - the scale factor.

8.44 expand3d - Expand image via pixel replication

This operator increases the size of the input image using the pixel replication.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Scale - the scale factor.

8.45 fill - Fill the holes

This operator fills the regions of "off" pixels surrounded by "on" pixels.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Connectivity - 4 or 8 neighbourhood.

8.46 gclose - Morphological closing

This operator performs morphological closing which is the dilation followed by erosion using the structural element supplied as the second input. The structural element can be calculated by the `strel` (see [8.139](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.47 gclose3d - Morphological closing

This operator performs morphological closing which is the dilation followed by erosion using the structural element supplied as the second input. The structural element can be calculated by the `strel3d` (see [8.140](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.48 gdilation - Dilation

This operator does successive dilations of the input image by the structural element supplied as the second input. The structural element can be calculated by the `strel` (see [8.139](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.49 gdilation3d - Dilation

This operator does successive dilations of the input image by the structural element supplied as the second input. The structural element can be calculated by the `strel3d` (see [8.140](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.50 geometry - Crop the preset number of pixels from the image

This operator reduces dimensions of the input image. It crops pixels from each side of the image. The number of pixels is defined in the second input that can be generated by the `crop` (see [8.22](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.51 geometry3d - Crop the preset number of pixels from the image

This operator reduces dimensions of the input image. It crops pixels from each side of the image. The number of pixels is defined in the second input that can be generated by the `crop` (see [8.22](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.52 gerosion - Erosion

This operator does successive erosions of the input image by the structural element supplied as the second input. The structural element can be calculated by the `strel` (see [8.139](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.53 gerosion3d - Erosion

This operator does successive erosions of the input image by the structural element supplied as the second input. The structural element can be calculated by the `strel3d` (see [8.140](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions - number of sweeps.

8.54 `getp` - Extract one plane from the stack

This operator extracts one plane from the stack.

INPUT

1 `image:.tif`

OUTPUT

1 `image:.tif`

PARAMETERS

Plane - the plane to extract.

8.55 `gmag` - Calculate the magnitude of the gradient

This operator calculates the magnitude of the gradient.

INPUT

1 `image:.tif`

OUTPUT

1 `image:.tif`

8.56 `gopen` - Morphological opening

This operator performs morphological opening which is the erosion followed by dilation using the structural element supplied as the second input. The structural element can be calculated by the `strel` (see 8.139) operator.

INPUT

1 `image:.tif`

2 `text:.txt`

OUTPUT

1 `image:.tif`

PARAMETERS

Repetitions - number of sweeps.

8.57 `gopen3d` - Morphological opening

This operator performs morphological opening which is the erosion followed by dilation using the structural element supplied as the second input. The structural element can be calculated by the `strel3d` (see 8.140) operator.

INPUT

1 `image:.tif`

2 text:.txt
OUTPUT
1 image:.tif
PARAMETERS
Repetitions - number of sweeps.

8.58 grid - Produce an image with the cartesian grid

This operator produces the image of the cartesian grid.

INPUT
1 image:.tif
OUTPUT
1 image:.tif
PARAMETERS
Number of bins - number of grid cells.

8.59 grid3d - Produce an image with the grid

This operator produces the image of the grid. The description of the shape is supplied as the second input. It can be produced by the `qu3dinit` (see [8.106](#)) operator.

INPUT
1 image:.tif
2 text:.txt
OUTPUT
1 image:.tif
PARAMETERS
Type - type of the grid - cartesian or elliptical.
Cell height - the height of the cell measured in per cent.
Cell width - the width of the cell measured in per cent.
Cell depth - the depth of the cell measured in per cent.
Offset - the offset of the center of the coordinate system measured in per cent. If set to 0 the center of the image will be the center of the coordinate system, if set to 0.5 the center of the image will be the center of the central cell.
Criterion - the allowed error.
Shape section - the name of the section in the second input that describes the shape of the object.

8.60 grid3d_i - Produce an image with the grid

This operator produces the image of the grid. The center of the object in the image is entered in the parameter dialog.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Type - type of the grid - cartesian or elliptical.

Cell height - the height of the cell measured in per cent.

Cell width - the width of the cell measured in per cent.

Cell depth - the depth of the cell measured in per cent.

Offset - the offset of the center of the coordinate system measured in per cent. If set to 0 the center of the image will be the center of the coordinate system, if set to 0.5 the center of the image will be the center of the central cell.

Criterion - the allowed error.

X Center - x coordinate of centroid.

Y Center - y coordinate of centroid.

Z Center - z coordinate of centroid.

8.61 halvesizes - Print the dimensions of the image

This operator prints dimensions of the image as a string of four numbers representing the distance in pixels from the image center to the borders.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

The format of output file:

<halfsizes>

upper offset, lower offset, left offset, right offset - the distances in pixels from the image center to the borders

</halfsizes>

8.62 heqm - Equalize histogram using mask

This operator performs the histogram equalization taking into account only those pixels that are "on" in the mask that comes from the second input.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

8.63 heq - Histogram equalization

This operator equalizes the histogram of the image. The idea is to spread out the histogram values to fill the entire range of the data type to enhance details that would be otherwise lost.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

8.64 hues - Combine color channels into one image

This operator combines grayscale images of three color channels (Red, Green and Blue) to the one grayscale image in which each pixel equals the product of Hue and Saturation values in this pixel. The color channels can be extracted by the `splitrgb` (see [8.136](#)) operator.

INPUT

1 image:.tif

2 image:.tif

3 image:.tif

OUTPUT

1 image:.tif

8.65 iapee - Visually correct the embryo orientation

This operator displays the image oriented according to the rules supplied in the second input and allows to flip and/or flop the image to correct the orientation. The output text file contains the list of needed operations.

```
INPUT
1 image:.tif
2 text:.txt
OUTPUT
1 text:.txt
    The format of an output file:
<iapee>
-
reversecolumn or - if not needed
reverserows or - if not needed
</iapee>
```

8.66 invert - Invert the image

This operator inverts the grayscale values of all pixels in input image.

```
INPUT
1 image:.tif
OUTPUT
1 image:.tif
```

8.67 invert3d - Invert the image

This operator inverts the grayscale values of all pixels in input image.

```
INPUT
1 image:.tif
OUTPUT
1 image:.tif
```

8.68 lev - Extract the pixels with certain intensity

Those pixels that have the given intensity in the input image have the maximal intensity in the output image. The other pixels are "off".

```
INPUT
1 image:.tif
OUTPUT
1 image:.tif
PARAMETERS
Index - intensity.
```

8.69 lheq - Local histogram equalization

This operator performs histogram equalization separately for each pixel using the neighborhood defined by the structural element from the second input. The structural element can be calculated by the `strel` (see [8.139](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions- number of sweeps

8.70 lheq3d - Local histogram equalization

This operator performs histogram equalization separately for each pixel using the neighborhood defined by the structural element from the second input. The structural element can be calculated by the `strel3d` (see [8.140](#)) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions- number of sweeps

8.71 lv3d - Rotate the stack around the vertical axis

This operator rotates the stack around the vertical axis.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

8.72 `m_ar_plus` - Sum up the image dimensions

This operator sums up the image dimensions that are given by a string of four numbers which are the distances measured in pixels from the image center to its borders. The dimensions can be calculated by the `halfsizes` (see 8.61) operator.

INPUT

1 `text:.txt`

OUTPUT

1 `text:.txt`

PARAMETERS

`Upper offset` - number of pixels to add at the top,

`Lower offset` - number of pixels to add at the bottom,

`Left offset` - number of pixels to add at the left,

`Right offset` - number of pixels to add at the right.

8.73 `m_ar_x` - Multiply the image dimensions by a constant

This operator multiplies the image dimensions given by a string of four numbers which are the distances measured in pixels from the image center to its borders. The dimensions can be calculated by the `halfsizes` (see 8.61) operator. The constant is defined as a parameter.

INPUT

1 `text:.txt`

OUTPUT

1 `text:.txt`

PARAMETERS

`Upper offset` - multiplier for the number of pixels to add at the top,

`Lower offset` - multiplier for the number of pixels to add at the bottom,

`Left offset` - multiplier for the number of pixels to add at the left,

`Right offset` - multiplier for the number of pixels to add at the right.

8.74 `match` - Find objects matching the template

This operator finds in the binary image objects matching the template provided in the second input as the structural element. The structural element can be calculated by the `strel` (see 8.139) operator. Each matching object

in the output binary image is marked with one bright pixel. This output can be used as the marker image for the `reconstruct` (see 8.113) operator.

```
INPUT
1 image:.tif
2 text:.txt
OUTPUT
1 image:.tif
```

8.75 `mask` - Mask the image

This operator masks the first input image by the second one. The pixels that are "off" in the mask are "off" in the output image. Other pixels are kept unchanged from the input. The mask comes from the second input.

```
INPUT
1 image:.tif
2 image:.tif
OUTPUT
1 image:.tif
```

8.76 `max` - Pixel by pixel maximum

This operator computes pixel by pixel maximum of three input images.

```
INPUT
1 image:.tif
2 image:.tif
3 image:.tif
OUTPUT
1 image:.tif
```

8.77 `max2` - Pixel by pixel maximum of two images

This operator computes pixel by pixel maximum of two input images.

```
INPUT
1 image:.tif
2 image:.tif
OUTPUT
1 image:.tif
```

8.78 max3d - Pixel by pixel maximum of the images in the stack

This operator computes pixel by pixel maximum of the images in the stack.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

8.79 maxlsm - Pixel by pixel maximum of the channels in the stack

This operator computes pixel by pixel maximum of the channels in the stack.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Comma separated list - the channels.

8.80 mcrop - Crop an image manually

This operator reduces the dimensions of the input image. It crops the defined number of pixels from the each side of the image.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Upper offset - number of pixels to crop from the top,

Lower offset - number of pixels to crop from the bottom,

Left offset - number of pixels to crop from the left,

Right offset - number of pixels to crop from the right.

8.81 mcrop3d - Crop an image manually

This operator reduces the dimensions of the input image. It crops the defined number of pixels from the each side of the image.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Upper offset - number of pixels to crop from the top,

Lower offset - number of pixels to crop from the bottom,

Left offset - number of pixels to crop from the left,

Right offset - number of pixels to crop from the right.

8.82 median - Median filter

This operator computes a two dimensional median filter of a structural element over the given image. The structural element comes from the second input.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions- number of sweeps

8.83 median3d - Median filter

This operator computes a two dimensional median filter of a structural element over the given image. The structural element comes from the second input.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Repetitions- number of sweeps

8.84 minusabs - Subtract images

This operator produces the image of the absolute value of differences between pixel values of input images.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

8.85 mlsreg - Moving least squares registration of images

This operator registers two images using control points and moving least squares method.

INPUT

1 image:.tif

2 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Alpha - alpha coefficient,

Type - affine, similar or rigid.

8.86 movl2 - Combine two grayscale images into one color image

This operator combines two grayscale images into one color image. The colors for each input can be selected from the list.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Color 1 - color for the first input,

Color 2 - color for the second input.

8.87 movl3 - Combine three grayscale images into one color image

This operator combines three grayscale images into one color image. The colors for each input can be selected from the list.

INPUT

1 image:.tif

2 image:.tif

3 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Color 1 - color for the first input,

Color 2 - color for the second input,

Color 3 - color for the third input.

8.88 movl6 - Combine six grayscale images into one color image

This operator combines six grayscale images into one color image. The colors for each input can be selected from the list.

INPUT

1 image:.tif

2 image:.tif

3 image:.tif

4 image:.tif

5 image:.tif

6 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Color 1 - color for the first input,

Color 2 - color for the second input,

Color 3 - color for the third input,

Color 4 - color for the fourth input,

Color 5 - color for the fifth input,

Color 6 - color for the sixth input.

8.89 `mpad` - Pad the image with a preset number of pixels

This operator expands the dimensions of the input image. It adds the given number of pixels from the each side of the image.

INPUT

1 `image:.tif`

OUTPUT

1 `image:.tif`

PARAMETERS

`Upper offset` - number of pixels to add from the top,

`Lower offset` - number of pixels to add from the bottom,

`Left offset` - number of pixels to add from the left,

`Right offset` - number of pixels to add from the right.

8.90 `mpad3d` - Pad the image with a preset number of pixels

This operator expands the dimensions of the input image. It adds the given number of pixels from the each side of the image.

INPUT

1 `image:.tif`

OUTPUT

1 `image:.tif`

PARAMETERS

`Upper offset` - number of pixels to add from the top,

`Lower offset` - number of pixels to add from the bottom,

`Left offset` - number of pixels to add from the left,

`Right offset` - number of pixels to add from the right.

8.91 `mslice3d` - Visualize distance map

This operator visualizes the distance image supplied as the first input. The shape description is supplied as the second input. The resulting image contains equidistance surfaces.

INPUT

1 `image:.tif`

2 `text:.txt`

OUTPUT

1 image:.tif

PARAMETERS

Cell step - the distance between surfaces measured in per cent,

Direction - the direction in which steps are made to mark control points,

Shape section - the name of the section in the second input that describes the shape of the object.

8.92 mslicegcp3d - Find landmarks or control points for registration

This operator finds control points for registration using the distance image. The shape description is supplied as the second input. It works like `mslice3d` (see 8.91) but steps are made in one direction to produce equidistance surfaces and in the other direction to mark landmarks.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Cell step - the distance between surfaces measured in per cent,

Direction - the direction in which steps are made,

Slice direction - the direction in which steps are made,

Shape section - the name of the section in the second input that describes the shape of the object.

Cell height - the height of the cell measured in per cent.

Cell width - the width of the cell measured in per cent.

Cell depth - the depth of the cell measured in per cent.

8.93 mul - Multiply two images

This operator multiplies each data point in multiplier image by the corresponding data point in input image and returns the product in the corresponding data point of the output image.

INPUT

1 image:.tif

```
2 image:.tif
OUTPUT
1 image:.tif
```

8.94 mul3d - Multiply two images

This operator multiplies each data point in multiplier image by the corresponding data point in input image and returns the product in the corresponding data point of the output image.

```
INPUT
1 image:.tif
2 image:.tif
OUTPUT
1 image:.tif
```

8.95 nplot - Plot 2D graph

This operator produces the 2D graph in PNG or XFIG format using `gnuplot`. The input file should contain the table of two or more columns one of which represents the arguments and the others represent the function values in corresponding points. The input file can be generated by `arcplot` (see 8.10) or `apro` (see 8.4) or `apron` (see 8.5) operator. Several columns can be plotted at once. Data should be sorted with respect to the argument. This operator can not produce the legend but `nplot2` (see 8.96) can.

```
INPUT
1 text:.txt
```

```
OUTPUT
```

```
1 image:.png
```

```
PARAMETERS
```

X column - number of the column with argument,

Y columns - the list of columns with function values,

Terminal - XFIG or PNG,

Line width - gnuplot option for the line width,

Line type - gnuplot option for the line type: lines, points, etc.

8.96 nplot2 - Plot 2D graph

This operator produces the 2D graph in PNG or XFIG format using `gnuplot`. The input file should contain the table of two or more columns one of which represents the arguments and others represent the function values at corresponding points. The input file can be generated by `arcplot` (see 8.10) or `apro` (see 8.4) or `apron` (see 8.5) operator. Several columns can be plotted at once. Data should be sorted in respect to the argument. This operator can produce the legend.

INPUT

1 `text:.txt`

OUTPUT

1 `image:.png`

PARAMETERS

`X column` - number of the column with argument,

`Y columns` - the list of columns with function values,

`Titles` - the list of titles, one title for each column,

`Terminal` - XFIG or PNG,

`Line width` - gnuplot option for the line width,

`Line type` - gnuplot option for the line type: lines, points, etc.,

8.97 pad - Pad the image

This operator expands the dimensions of the input image. It adds the pixels to the each side of the image. The number of pixels is defined in the second input that can be generated by the `crop` (see 8.22) operator.

INPUT

1 `image:.tif`

2 `text:.txt`

OUTPUT

1 `image:.tif`

8.98 pad3d - Pad the image

This operator expands the dimensions of the input image. It adds the pixels to the each side of the image. The number of pixels is defined in the second input that can be generated by the `crop` (see 8.22) operator.

```
INPUT
1 image:.tif
2 text:.txt
OUTPUT
1 image:.tif
```

8.99 plot_sp - Print the pixel values to the text file

This operator prints the values of pixels to the text file. Increment parameter n greater than 1 allows to print out only each n th pixel. Function parameter allows to print the actual values (eqn) or the natural logarithm (log) of the intensity.

```
INPUT
1 image:.tif
OUTPUT
1 text:.txt
PARAMETERS
Increment - counter increment,
Function - actual value or logarithm.
```

8.100 ppix - Print the pixel values along the line

This operator prints the values of pixels along the line to the text file. The line is defined as two parametric functions: $r = a_r * t + b_r$ for row and $c = a_c * t + b_c$ for column, where parameter t spans the interval $[t_0, t_N]$. The second output shows the image with the line superimposed on it.

```
INPUT
1 image:.tif
OUTPUT
1 text:.txt
2 image:.tif
PARAMETERS
ac - slope of parametric function for column,
bc - offset of parametric function for column,
ar - slope of parametric function for row,
br - offset of parametric function for row,
t0 - lower limit for parameter,
tN - upper limit for parameter.
```

8.101 prutik - Measure two distances visually

This operator shows the dialog window with the input image in which the user is allowed to mark two intervals that are then measured in pixels. The measurements are print in the output file.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

8.102 pump3d - Convert distance image into volume

This operator converts an image into a stack using the distance map to determine the z coordinate.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

8.103 pump3d_data - Convert distance image into volume with data

This operator converts an image into a stack using the distance map to determine the z coordinate. It also set the intensities in the stack according the second input.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

8.104 qu3d - Add quantitative data from another channel

This operator adds the quantitative data from another channel to the initialized storage. The image to extract data from is passed as the first input. The second input should contain the binary object mask. The initialized storage

is supplied in the third input. The storage is initialized by the `qu3dinit` (see 8.106) operator.

INPUT

1 image:.tif

2 image:.tif

3 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Connectivity - 6 or 26,

Tag - String identifier for this channel.

8.105 qu3d2csv - Convert quantitative information to CSV format

This operator converts the quantitative data to CSV format.

INPUT

1 text:.txt

OUTPUT

1 text:.txt

8.106 qu3dinit - Initialize storage for quantitative data

This operator initializes the storage and adds the quantitative data from one channel. The image to extract data from is passed as the first input. The second input should contain the binary object mask.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 text:.txt

PARAMETERS

Connectivity - 6 or 26,

Tag - String identifier for this channel.

8.107 qu3dtrans - Initialize storage for quantitative data

This operator filters the list of the objects and transforms their coordinates. The second input should contain the description of the shape of the object mask.

INPUT

1 text:.txt

2 text:.txt

OUTPUT

1 text:.txt

PARAMETERS

Shape section - the name of the section that describes the shape of the mask,

Percent - if set coordinates are converted to the per cent,

Center - if set coordinates are shifted to the center,

Min Volume - minimal number of pixels in the object,

Max Volume - maximal number of pixels in the object.

8.108 qumap3d - Paint the mask with quantitative data

This operator paints the binary mask with the intensities from quantitative data. The second input should contain the quantitative data produced by the `qu3dinit` (see 8.106) or the `qu3d` (see 8.104) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

PARAMETERS

Connectivity - 6 or 26,

Channel - String identifier of the channel,

Index - The number starting from zero that determines the quantity in the list of measurements. Usually 0 means mean value, 1 - variation, 2 - standard deviation, 3 - maximum, 4 - minimum and 5 - median.

8.109 **qumark3d - Paint the mask with quantitative data**

This operator collapses the islands of bright pixels in the binary image to exactly one pixel. It is used to refine the landmarks for registration with `mlsreg` (see 8.85) operator.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

8.110 **qurelabel - Relabel objects after the registration of the mask**

This operator labels each object in the registered mask with its index in the unregistered mask. Thus it is possible to compare quantitative data extracted from different images using the same mask registered to these images. The unregistered mask is supplied in the first input as binary image. The registered mask is supplied in the second input as the output of the `qu3dinit` (see 8.106) operator. The third input should contain the landmark of the registered image, and the fourth input should contain the landmarks of the unregistered mask.

INPUT

1 image:.tif

2 text:.txt

3 image:.tif

4 image:.tif

OUTPUT

2 text:.txt

PARAMETERS

Alpha - alpha coefficient,

Type - affine, similar or rigid,

Connectivity - 6 or 26.

8.111 **raw - Make tiff from raw**

This operator reads raw image and writes it in tiff format.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Columns - number of columns in the image,

Rows - number of rows in the image,

Bps - number of bits per pixel.

8.112 rec3dbp - Morphological reconstruction

This operator performs morphological reconstruction of the stack from the marker image. The specified plane of the stack is compared to the marker image.

INPUT

1 image:.tif

2 marker:.tif

OUTPUT

1 image:.tif

PARAMETERS

Connectivity - 6 or 26,

Plane - the plane of the stack to compare.

8.113 reconstruct - Morphological reconstruction

This operator performs morphological reconstruction of the image from the marker image.

INPUT

1 image:.tif

2 marker:.tif

OUTPUT

1 image:.tif

8.114 reconstruct3d - Morphological reconstruction

This operator performs morphological reconstruction of the image from the marker image.

INPUT

1 image:.tif

2 marker:.tif

OUTPUT

1 image:.tif

8.115 regmax - Regional maxima

This operator finds regional maxima.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Minimal number of pixels - Minimal number of pixels in the object,

Maximal number of pixels - Maximal number of pixels in the object,

Minimal number of blobs - Minimal number of objects,

Connectivity - 4 or 8.

8.116 regmin - Regional minima

This operator finds regional minima.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Minimal number of pixels - Minimal number of pixels in the object,

Maximal number of pixels - Maximal number of pixels in the object,

Minimal number of blobs - Minimal number of objects,

Connectivity - 4 or 8.

8.117 restack3d - Remove planes from the stack

This operator removes planes from the stack with the specified step.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Step - planes are removed with this step.

8.118 revcol - Reverse columns

This operator reverses columns in the image. Thus the leftmost pixel in each row becomes the rightmost one and so on.

INPUT

```
1 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.119 revcol3d - Reverse columns

This operator reverses columns in the image. Thus the leftmost pixel in each row becomes the rightmost one and so on.

INPUT

```
1 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.120 revrow - Reverse rows

This operator reverses rows in the image. Thus the uppermost pixel in each column becomes the lowest and so on.

INPUT

```
1 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.121 revrow3d - Reverse rows

This operator reverses rows in the image. Thus the uppermost pixel in each column becomes the lowest and so on.

INPUT

```
1 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.122 robel - Produce the image of round strip

This operator produces the image of round strip that includes all blobs from the second input. The first input is used to determine the center of polar coordinates. The third input contains the list of points in polar coordinates that determine the inner and outer borders of the round strip. This file can be generated by the `ropri` (see 8.125) operator. The pixel in the output image is "on" if it is "on" in the second input and its coordinates belong to the round strip.

INPUT

1 image:.tif

2 image:.tif

3 text:.txt

OUTPUT

1 image:.tif

8.123 rogri - Divide the image into sectors

This operator divides the image from the second input into sectors. The pixels are switched off in the output if their polar angle is closer than the predefined value (Accuracy) to the sector delimiter.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Number of bins - number of sectors to produce,

Accuracy

8.124 ropol - Print the pixel values

This operator prints the pixel values from the third input to the text file in polar coordinates. The center of the coordinate system is calculated from the first input. The second input is used as the mask. Function parameter allows to print the actual values (eqn) or the natural logarithm (log) of the intensity.

INPUT

1 image:.tif

2 image:.tif

3 image:.tif

OUTPUT

1 text:.txt

PARAMETERS

Function - the actual value (eqn) or the natural logarithm (log).

8.125 ropri - Print the landmarks of the borders of the round strip

This operator calculates the list of points in polar coordinates that determines the inner and outer borders of the round strip. The center of the polar coordinates is calculated from the first input. The second input is the list of blobs with polar coordinates of centroids that can be calculated by the `blo2pol` (see 8.13) operator. The output is sorted with respect to the polar angle.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 text:.txt

8.126 rotate - Calculate the rotation angle

This operator computes the rotation angle using invariant moments and rotates the image. The text file with calculated angle is the second output.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 text:.txt

8.127 rv3d - Rotate the stack around the vertical axis

This operator rotates the stack around the vertical axis in the direction opposite to that used in the `lv3d` (see 8.71) operator.

INPUT
1 image:.tif
OUTPUT
1 image:.tif

8.128 save - Save file

This operator saves an input file in the server "public" folder with assigned name.

INPUT
1 image:.tif

8.129 setp - Set one plane in the stack

This operator sets one plane in the stack from the second input.

INPUT
1 image:.tif
OUTPUT
1 image:.tif
PARAMETERS
Plane - the plane to set.

8.130 shape - Characterize the shape

This operator outputs the description of the shape.

INPUT
1 image:.tif
OUTPUT
1 image:.tif

8.131 shape3d - Characterize the shape

This operator outputs the description of the shape.

INPUT
1 image:.tif
OUTPUT
1 image:.tif

PARAMETERS

Connectivity - 6 or 26.

8.132 shrink - Shrink image via pixel subsampling

This operator reduces the size of the input image using pixel subsampling.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Scale - the scaling factor from 0 to 1.

8.133 shrink3d - Shrink image via pixel subsampling

This operator reduces the size of the input image using pixel subsampling.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Scale - the scaling factor from 0 to 1.

8.134 solo - Extract quantitative information

This operator calculates the given statistical estimator of pixel intensities for each blob, listed in the first input that can be generated by the blob operator. The statistical estimators are calculated for each of five input images – channels (inputs 2 – 6). They are placed in the corresponding column of the output file. The mean value, median value, maximum, minimum, number of pixels, variation or standard deviation can be calculated for blobs (nuclear), their surroundings (outnuc) and the union of them (energid). The ratio of variances in pixel values between and within these two classes (varbc) can be calculated if outnuc is selected as the Mask parameter for this channel.

INPUT

1 text:.txt

2 image:.tif

3 image:.tif

4 image:.tif

5 image:.tif

6 image:.tif

OUTPUT

1 text:.txt

PARAMETERS

The following parameters are defined for each of five input channels:

Mask - nuclear, energid or outnuc,

Stat - mean, median, max, min, stdev, var, area or varbc,

P - one letter label.

8.135 splitlsm - Write each color in the separate file

This operator splits the input color image into three grayscale images, one for each color channel.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 image:.tif

3 image:.tif

8.136 splitrgb - Write each color in the separate file

This operator splits the input color image into three grayscale images, one for each color channel.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 image:.tif

3 image:.tif

8.137 sselect - Select objects according to the shape

This operator filters objects in the binary image according to the shape. The template is provided in the second input as the binary image. The comparison is insensitive to the rotation, translation and scaling.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Max segment - maximal number of pixels in the object,

Min segment - minimal number of pixels in the object,

Rule - accept or reject,

Criterion - maximal allowed error,

Connectivity - 4 or 8.

8.138 sselect3d - Select objects according to the shape

This operator filters objects in the binary image according to the shape.

The template is provided in the second input as the binary image. The

comparison is insensitive to the rotation, translation and scaling.

INPUT

1 image:.tif

2 image:.tif

OUTPUT

1 image:.tif

PARAMETERS

Max segment - maximal number of pixels in the object,

Min segment - minimal number of pixels in the object,

Rule - accept or reject,

Criterion - maximal allowed error,

Connectivity - 6 or 26.

8.139 strel - Structural element

This operator generates a structural element file.

INPUT

0 OUTPUT

1 text:.txt

PARAMETERS

Window width - in pixels,

Window height - in pixels,
Shape - disk or square.

8.140 strel3d - Structural element

This operator generates a structural element file.

INPUT

0 OUTPUT

1 text:.txt

PARAMETERS

Window width - in pixels,
Window height - in pixels,
Window depth - in pixels,
Shape - disk or square,
Shape - disk or square.

8.141 strel3dw - Structural element

This operator generates a structural element file.

INPUT

0 OUTPUT

1 text:.txt

PARAMETERS

Window width - in pixels,
Window height - in pixels,
Window depth - in pixels,
Window wall - in pixels,
Shape - disk or square,
Shape - disk or square.

8.142 surf2vol - Draw triangulated surface

This operator draws triangulated surface.

INPUT

1 text:.txt

OUTPUT

1 image:.tif

8.143 surf3d - Produce triangulated surface

This operator produces triangulated surface.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

PARAMETERS

Scale column - horizontal scale,

Scale row - vertical scale,

Scale plane - scale in z direction,

Maximal penalty of edge removal - for surface refinement,

Minimal angle - for surface refinement,

Format - vtk, oogl, ooglb or gts.

8.144 surf3dfull - Produce triangulated surface with full control

This operator produces triangulated surface.

INPUT

1 image:.tif

OUTPUT

1 text:.txt

PARAMETERS

Scale column - horizontal scale,

Scale row - vertical scale,

Scale plane - scale in z direction,

Maximal penalty of edge removal - for surface refinement,

Minimal angle - for surface refinement,

Format - vtk, oogl, ooglb or gts,

Step x - in pixels,

Step y - in pixels,

Step z - in pixels,

Function - cartesian, tetra, tetra_bounded or tetra_bcl.

8.145 **threshb** - Threshold pixel values in each blob

This operator generates a binary image by thresholding the input image. Pixel values greater than the cutoff value are set to 255 in the output image. Pixel values less or equal to the cutoff are set to 0. The cutoff value is calculated by Otsu's method in each blob contained in the second input. Otsu's method chooses the threshold to minimize the intraclass variance of black and white pixels [3].

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.146 **threshold** - Threshold pixel values

This operator generates a binary image by thresholding the input image. Pixel values greater than the cutoff value are set to 255 in the output image. Pixel values less or equal to the cutoff are set to 0. The cutoff value can be specified by a user or calculated by Otsu's method, which chooses the threshold to minimize the intraclass variance of black and white pixels [3]. Second output gives the actual threshold used.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 text:.txt

PARAMETERS

Threshold level - the threshold value,

Method - plain or otsu,

Process? - do actual transformation or not.

8.147 **threshold3d** - Threshold pixel values

This operator generates a binary image by thresholding the input image. Pixel values greater than the cutoff value are set to 255 in the output image. Pixel values less or equal to the cutoff are set to 0. The cutoff value can be specified by a user or calculated by Otsu's method, which chooses the

threshold to minimize the intraclass variance of black and white pixels [3].
Second output gives the actual threshold used.

INPUT

1 image:.tif

OUTPUT

1 image:.tif

2 text:.txt

PARAMETERS

Threshold level - the threshold value,

Method - plain or otsu,

Process? - do actual transformation or not.

8.148 turn - Rotate image on given angle

This operator rotates image to a given angle (second input) that can be calculated by the rotate (see 8.126) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.149 turn3d - Rotate image on given angle

This operator rotates image to a given angle (second input) that can be calculated by the rotate (see 8.126) operator.

INPUT

1 image:.tif

2 text:.txt

OUTPUT

1 image:.tif

8.150 tv3d - Rotate the stack around the horizontal axis

This operator rotates the stack around the horizontal axis.

INPUT

1 image:.tif

OUTPUT

```
1 image:.tif
```

8.151 vavg - Average images pixel by pixel

This operator averages two images pixel by pixel.

INPUT

```
1 image:.tif
```

```
2 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.152 vmabs - Subtract images

This operator produces the image of the absolute value of differences between pixel values of input images.

INPUT

```
1 image:.tif
```

```
2 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.153 vmax - Pixel by pixel maximum of two images

This operator computes pixel by pixel maximum of two input images.

INPUT

```
1 image:.tif
```

```
2 image:.tif
```

OUTPUT

```
1 image:.tif
```

8.154 vrgb - Combine three grayscale images into one color image

This operator combines three grayscale images into one color image.

INPUT

```
1 image:.tif
```

```
2 image:.tif
```

```
3 image:.tif
OUTPUT
1 image:.tif
```

8.155 vrmbg - Remove background signal

This operator removes background signal from the image. Background is to be estimated quantitatively and its mean and standard deviation is to be present in the appropriate section in the second input. The value $\langle \text{Bgr mean coeff} \rangle * \langle \text{bgr mean} \rangle + \langle \text{bgr stdev coeff} \rangle * \langle \text{bgr stdev} \rangle$ is subtracted from each pixel.

INPUT

```
1 image:.tif
2 text:.txt
```

OUTPUT

```
1 image:.tif
```

PARAMETERS

Bgr mean coeff - the coefficient for the mean value of background signal,
Bgr section - the section with measurements in the second input,
Bgr stdev coeff - the coefficient for the standard deviation,
Bgr data - the string identifier for the measurements.

8.156 vrmbg_1 - Remove background signal

This operator removes background signal from the image. The mean and standard deviation of it is to be provided by the user. The value $\langle \text{Bgr mean coeff} \rangle * \langle \text{bgr mean} \rangle + \langle \text{bgr stdev coeff} \rangle * \langle \text{bgr stdev} \rangle$ is subtracted from each pixel.

INPUT

```
1 image:.tif
2 text:.txt
```

OUTPUT

```
1 image:.tif
```

PARAMETERS

Bgr mean coeff - the coefficient for the mean value of background signal,
Bgr mean - the mean value of background signal,
Bgr stdev coeff - the coefficient for the standard deviation,
Bgr stdev - the standard deviation.

8.157 `vstrel` - Convert structural element to the image

This operator convert structural element to the image.

INPUT

1 `text:.txt`

OUTPUT

1 `image:.tif`

8.158 `vstrel3d` - Convert structural element to the image

This operator convert structural element to the image.

INPUT

1 `text:.txt`

OUTPUT

1 `image:.tif`

8.159 `vtxt` - Convert the stack to text file

This operator converts the binary image stack to the text file.

INPUT

1 `image:.tif`

OUTPUT

1 `text:.txt`

PARAMETERS

Format - XYZ or VRML,

Connectivity - 6 or 26.

8.160 `vvarbc3d` - Calculates the ratio of variances

This operator calculates the ratio of variancies of pixel values in the image between and within two classes. The pixels of the first class belong to blobs, the second class includes all the pixels that belong to surroundings. The input is the list of blobs and can be calculated by the `qu3dinit` (see 8.106) operator.

INPUT

1 `text:.txt`

OUTPUT

```
1 text:.txt
PARAMETERS
Output section - the name of the output section,
Class 1 section - the name of the section that describes first class,
Class 1 data - the string identifier of the data that describes first class,
Class 2 section - the name of the section that describes second class,
Class 2 data - the string identifier of the data that describes second class.
```

8.161 zscale3d - Scale the stack in z direction

This operator scales the stack in z direction.

INPUT

```
1 image:.tif
```

OUTPUT

```
1 image:.tif
```

PARAMETERS

Scale - the scale.

References

- [1] T. Crimmins. The geometric filter for speckle reduction. *Applied Optics*, 1985.
- [2] R.C. Gonzalez and R.E. Woods. *Digital image processing*. Prentice-Hall Inc., 2002.
- [3] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions Systems, Man, and Cybernetics*, 1979.
- [4] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Machine Intell.*, 1990.
- [5] J. Shen and S. Castan. Further results on drf method for edge detection. In *9th I.C.P.R.*
- [6] J. Shen and S. Castan. An optimal linear operator for step edge detection. *CVGIP: Graphical Models and Understanding*, 1992.