OpenCV Laboratory

Release 1.0 alpha

OCVL team / Teredo team

May 02, 2020

Contents

Content:

CHAPTER 1

Installation

1.1 Installation from a binary package

If the program is installed from a binary package, all dependencies are already in this package. So if the system meets the minimum requirements, the program will be ready for use after installing the binary package.

1.2 Installation of the source package directly in Blender

In case you want to install the program from the source code, we need to meet several prerequisites.

- Blender (https://www.blender.org/download/) - suggested version 2.80
- opencv-python (https://pypi.python.org/pypi/opencv-python) - suggested version 3.1.0.1

1.2.1 Belnder installation

Blender can be downloaded from the program website http://blender.org/. OCVL works steadily on version 2.80 and it is recommended installation of this version of Blender.

1.2.2 Installation of OpenCV

To install *python* packages, it is best to use *pip*. By default, 'Pip' is not included in Blenders Python and you must install it first. *Pip* can be downloaded from:' https: // bootstrap.pypa.io / get-pip.py'. Then just run the Python script.

Depending on the operating system and version of Blender, Python may have a different name and location. Here's an example of Python's location in Blender: *~/Downloads/blender-2.79-macOS-10.6/ blender.app/Contents/Resources/2.79/python/bin/python3.5m* The Pip installation will look like: *~/Downloads/blender-2.79-macOS-10.6/blender.app/Contents/Resources/2.79/python/bin/python3.5m ./get-pip.py*

Now with Pip you can install packages directly through it. Pip depending on the system can install itself as a script or as a module.

Now you can install packages: ../python/bin/python3.5m ../python/bin/pip install opencv-python==3.1.0.5

cd ~/Downloads/blender-2.79-macOS-10.6/OpenCVLaboratory.app/Contents/Resources/2.79/scripts/

ln -s ~/workspace/tales/ocvl-addon ocvl

1.2.3 Installation of OCVL addon to Blender

After downloading OCVL (https://github.com/feler404/ocvl-addon), just unpack it and copy it to the addons directory

1.2.4 Running Blender with addon

In order for Blender to be able to run all the add-ons correctly, it must be run by the command: *./blender –addons ocvl*

CHAPTER 2

Dependencies

1) Base application: Blender (https://www.blender.org/download/)
2) **Python packages**
    - numpy
    - opnecv-python
3) **Blender extensions**
    - ocvl

CHAPTER 3

Introduction

3.1 Introduction to OpenCV

OpenCV (Open Source Computer Vision Library http://opencv.org) is an open library containing several hundred algorithms of computer vision based on BSD. OpenCV library is divided into modules and this division is also reflected in OpenCV Laboratory.

- *core* - a compact module defining basic data structures and containing basic functions used in the rest of the modules
- ***imgproc*** **- image processing module includes functions: linear and non-linear filtering, transformation, changes**
  size, color hue changes, operations on histograms, etc.
- *video* - module for video analysis includes, among others, functions: traffic estimation, background removal and object tracking
- ***calib3d*** **- basic algorithms for calculating the geometry of multiple images, single calibration and double camera,**
  image position estimation, stero correspondence algorithm and 3D reconstruction functions
- *features2d* - essential feature detectors, descriptors and matching descriptors
- *objdetect* - detection of objects and instances of predefined classes (eg face, eyes, mugs, people, cars, etc.)
- . . .

3.2 Introduction to Blender

Blender is a free and open source 3D creation package. It supports entire modeling of 3D pipelines, rigging, animation, simulation, rendering, composing, movement tracking, and even editing video and creating games. Advanced Users use the Blender API to support scripting in Python to customize applications and writing specialist tools; they are often included in future releases of Blender. Blender is fine suited to individuals and small studies that benefit from a unified system and a flexible development process. Examples of many projects based on Blender are available in the form of presentations.

For the needs of OpenCV Laboratory, the Blender node system is used, which is the basis of the application.

3.3 Introduction to OpenCV Laboratory

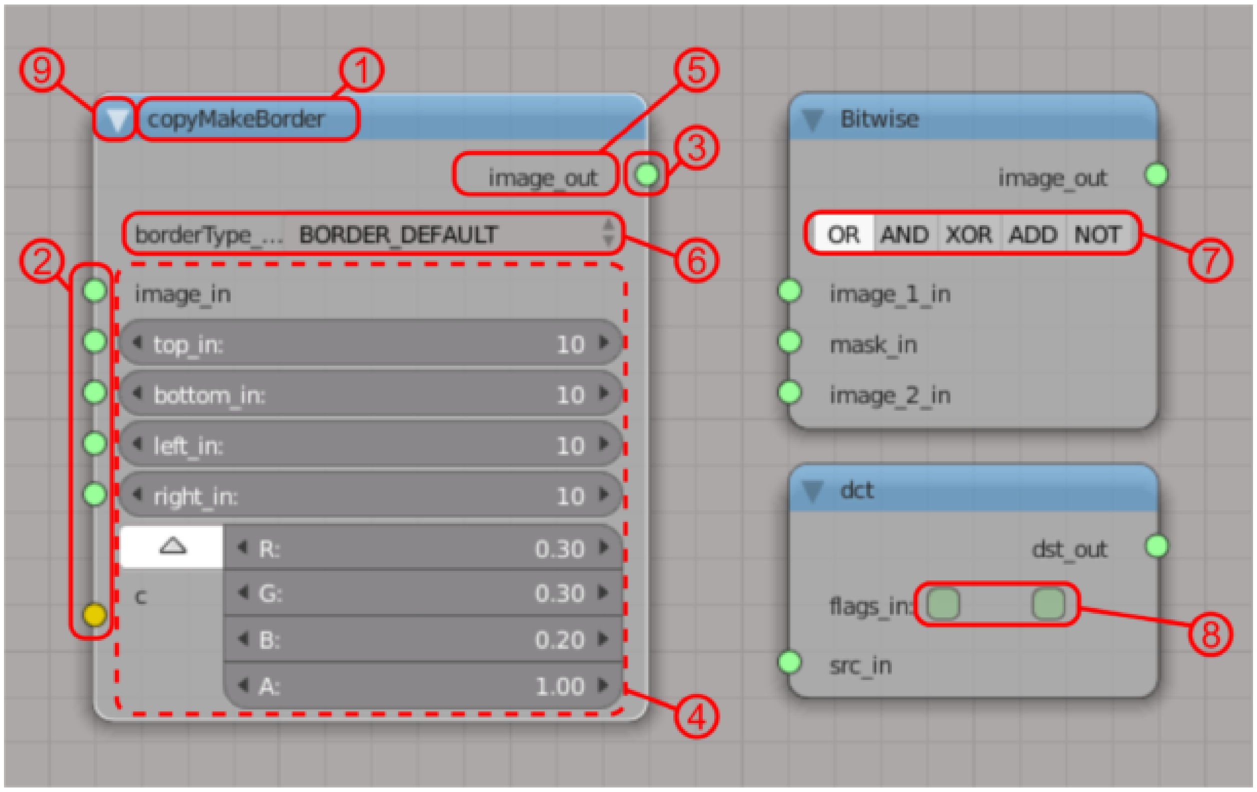
The laboratory is a series of preinstalled Python libraries and a set of Blender extensions. On this base A set of OpneCV functions has been implemented in the form of convenient to connect nodes, in which we have fast and convenient access to all parameters of the function and, in addition, an immediate preview of the result of these functions. Laboratory in addition to the primary node equivalents of the OpenCV library has also very important input / output nodes. To this pool of nodes include: Image Sampler, Image Viewer, ROI, Custom Input, Custom I / O, Stethoscope, TypeConvert.

3.3.1 Node

OpenCV Laboratory is a series of preinstalled Python libraries and the Blender extension kit. On this basis, a set of OpneCV functions has been implemented in the form of convenient to connect nodes in which we have fast and convenient access to all parameters of the function, and also an immediate preview of the result of these functions.

Visually, each node in the OpenCV Laboratory is represented by a rectangle with rounded corners. Each of them has round sockets, an input socket on the left and output sockets on the opposite side.

Below is an example with a short description of different nodes.



1. Node name.
2. Input sockets.
3. Output socket.
4. Input parameters to which the appropriate input sockets are assigned. It is often possible to freely adjust individual arguments using sliders or by entering a specific value.
5. The output parameter is the result of operations performed by a given node.

6. Internal parameter which is a list of choice of a specific function.
7. Internal parameter in the form of buttons defining the function selection.
8. Internal parameter in the form of an acceptance field that takes into account the operation of the function.
9. Button for minimizing the node view.

3.3.2 Connecting nodes

Connecting nodes is nothing else but a command to perform appropriate functions by a computer program with the final result of their actions. Mutual connection of individual nodes is a relatively simple operation consisting in connecting with a line, the output socket of one of the nodes with the corresponding input socket of another node.

The above operation is carried out as follows:

1. Click with the left mouse button the output socket of the given node.
2. Without releasing the button, route the lines to the input socket of another node.
3. Release the mouse button.

In the presented example, no complicated operations were performed, resulting in an output image identical to the entered input image.

It's simple, right?

3.3.3 Invalid node data entry

Due to the properties and functions performed, not all nodes can be directly connected with each other. Each node requires a specific parameter for the appropriate input, some of them need to enter all relevant data. In OpenCV Laboratory, irregularities in the above cases are illustrated by a change in node color.

In problematic cases, to achieve the right final result, it is often enough to fine-tune the node settings, supplement with the required data or, depending on the need, apply the compilation of other nodes.

3.3.4 Image Sampler

A node which task is to generate / load an image for further processing

3.3.5 Image Viewer

This node is used to view the image. It has two built-in modes. The default thumbnail mode where the image is displayed in the node itself, and preview mode, where we have access to all the details of the image in full screen, along with the possibility zooming or previewing a pixel by pixel.

3.3.6 ROI

With this node, we can conveniently cut the image and select the fragment we are interested in.

3.3.7 Custom Input

This node can generate / download data from any source based on the Python code.

3.3.8 Custom I/O

This node can accept any data from other nodes and process them from the Python code level.

3.3.9 Stethoscope

Used to view data in numerical form.

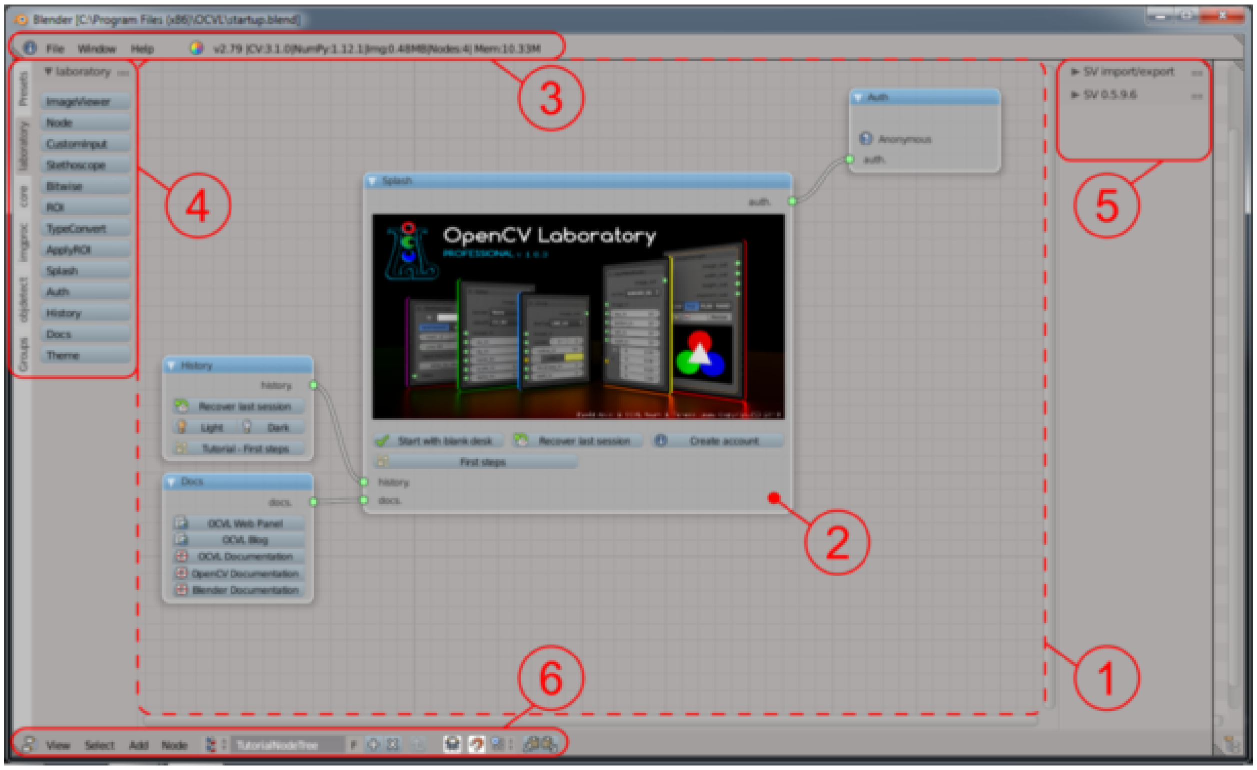
3.3.10 TypeConvert

With this node, we can quickly change the data type (uint8, float32, float64, etc.) from which the image is composed.

CHAPTER 4

Welcome to the OpenCV Laboratory program!

The OpenCV Laboratory program is launched with the main view of interface, which includes: work area (1) with a welcome window (2), information panel bar (3), nodes panel (4), property panel (5) and bar of nodes editor panel (6).



4.1 Work area and welcome window

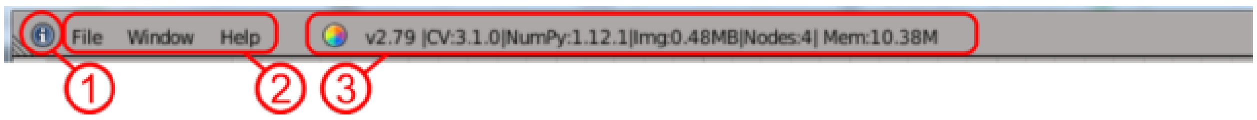
The work area occupies the central part of the view, and after starting the OpenCV Laboratory program, a welcome window is additionally displayed.

The welcome window consists of four elements: Splash, History, Docs i Auth.

- **Splash - the main section contains the logo, the name of the program with information about its version, and the follo**
  - *Start with blank desk* - start work from an empty workspace.
  - *Recover last session* - as it is written.
  - *Create account* - as it is written.
  - *First steps* - tutorial.
- **History - section with the following options:**
  - *Recover last session* - as it is written.
  - *Light* - “day” view display mode.
  - *Dark* - “night” view display mode.
  - *Tutorial - First steps* - linkt to tutorials about program.
- **Docs - a section with links to documentation related to the program:**
  - *OCVL Web Panel*: <https://ocvl-cms.herokuapp.com/admin/login/>
  - *OCVL Blog*: <http://kube.pl/>
  - *OCVL Dacumentation*: <http://opencv-laboratory.readthedocs.io/en/latest/?badge=latest>
  - *OpenCV Documentation*: <https://docs.opencv.org/3.0-beta/index.html>
  - *Blender Documentation*: <https://docs.blender.org/manual/en/dev/editors/node_editor/introduction.html>
- Auth - authorization section.

4.2 Information panel bar

The information panel bar contains the panel icon (1), a menu with drop-down lists (2) and information about the program version, the number of nodes present in the work area and the memory used by the current project (3).



v2.79 |CV:3.1.0|NumPy:1.12.1|Img:0.48MB|Nodes:4| Mem:10.38M

The menus are lists concerning: file operations, view options and help.

- **File - traditional document-related operations used in most application programs:**
  - *New* - Create new project.
  - *Open* - Open project.
  - *Open Recent. . .* - open one of the most recently used projects.
  - *Revent* - re-load the last saved version of the file.

- *Recover Last Session* - as written.
- *Recover Auto Save…* - restore the last automatic recording.
- *Save* - save project.
- *Save As…* - as written.
- *Save Copy* - Save as copy of the procejt.
- *Quit* - as written.

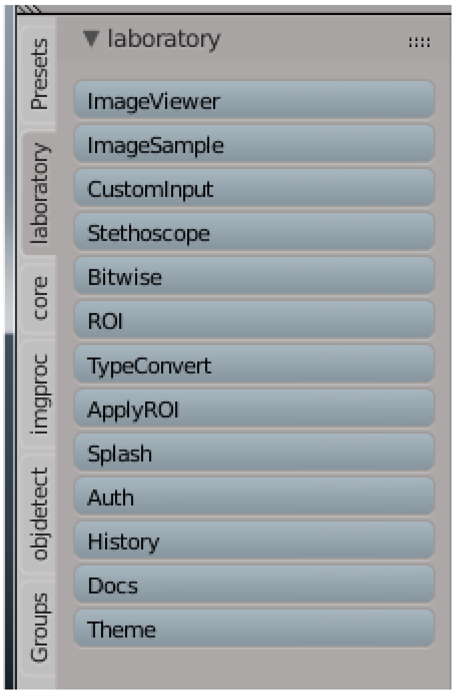
• **Window - options for the view:**

- *Duplicate Window* - create a duplicate of the current view in a new window.
- *Toggle Window Fullscreen* - expand the window to the entire screen.
- *Save Screenshot* - create a view image.
- *Toggle System Console* - console with logs.

• **Help - mainly relevant documentation:**

- *OCVL Documentation*: <http://opencv-laboratory.readthedocs.io/en/latest/?badge=latest>
- *OpenCV Documentation*: <http://kube.pl/>
- *Blender Dokumentation*: <https://docs.blender.org/manual/en/dev/editors/node_editor/introduction.html>
- *OCVL Web Panel*: <https://ocvl-cms.herokuapp.com/admin/login/>
- *OCVL Blog*: <http://kube.pl/>
- *Show Node Splash* - show a welcome window.

4.3 Tools panel - nodes

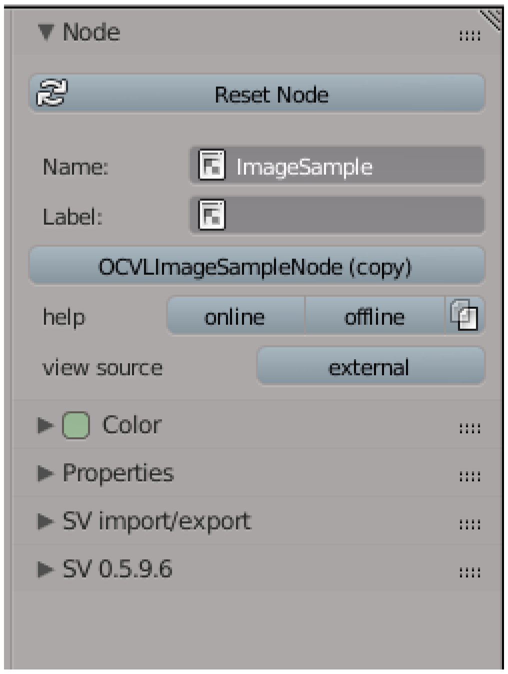


The tool panel - nodes located on the left side of the view screen are tabs in which individual nodes are grouped and placed in th

- Laboratory - basic nodes most often used.
- Core - nodes related to the kernel.
- Imgproc - nodes directly related to the visual side of the image.
- Objdetect - nodes associated with objects in the image.
- Groups - tab associated with grouping nodes.

Note: Description of individual nodes can be found in the OpenCV Laboratory documentation by clicking the link: <http://opencv-laboratory.readthedocs.io/en/latest/nodes.html>

4.4 Properties panel



Name: ImageSample

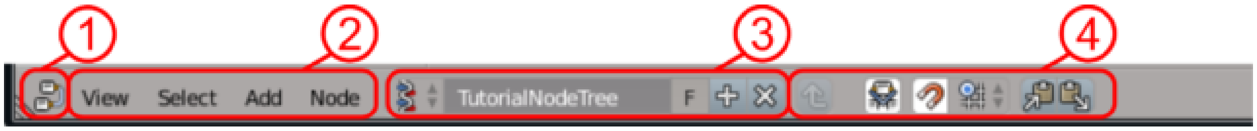
Label:

The property panel located on the right side of the view screen contains a number of options related to displaying information a

- Node - in the tab there is, among other things, the possibility of resetting the settings made at a given node (Reset Node), entering your own name (Level), and help related to a specific node is available.
- Color - the ability to set any color of the node.
- Properties - properties that a given node has and access to additional information contained in specific documentation as well as calculation times.

4.5 The node editor panel bar

The node editor panel bar located at the bottom of the view screen, contains the bar icon (1), menu with drop-down lists (2), node tree viewer (3), and additional function keys (4).

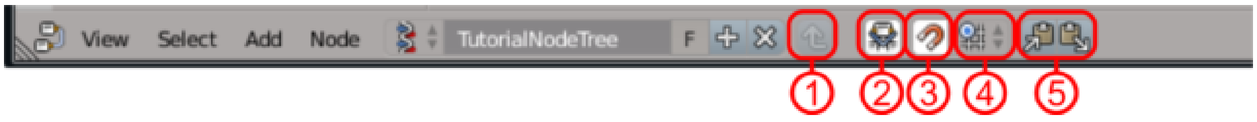


The bottom panel menu consists of lists for: view, selection, node placement and node operations.

- **View - view options:**
  - *Properties* - show or hide the node properties panel.
  - *Tool Shelf* - show or hide the tools panel - nodes.
  - *Zoom In* - enlarge the view in the work area.
  - *Zoom Out* - reduce the view in the work area.
  - *View Selected* - change the size of the view so that you can see the selected nodes located in the work area.
  - *View All* - change the view size so that you can see all nodes located in the work area.
  - *Duplicate Area into New Window* - .
- **Select - selection options:**
  - *Border Select* - select specific nodes within a rectangular frame.
  - *Circle Select* - select specific nodes within a circular area.
  - *(De)select All* - deselect or select all nodes.
  - *Iverse* - the inverse of the selection.
  - *Select Linked From* - select the nodes with connections that reach the selected node.
  - *Select Linked To* - select the nodes with connections coming from the selected node.
  - *Select Grouped* - select a group of nodes by: type, color, prefix, suffix.
  - *Active Same Type Previous* - activate the previous node of the same type.
  - *Active Same Type Next* - activate another node of the same type.
  - *Find Node* - Find a specific node.
- Add - nodes grouped and deployed in the appropriate categories as in the tools - nodes panel.
- **Node - nodes options:**
  - *Translate* - move the node to the desired location.
  - *Duplicate* - duplicate the node.
  - *Delete* - remove node.
  - *Delete with Reconnect* - delete with reconnection.
  - *Make Links* - create a connection between selected nodes.
  - *Make and Replace Links* - create a connection between selected nodes.
  - *Cut Link* - cut off the connection between nodes.

- *Detach Links* - delete all connections of selected nodes and make new connections with neighboring nodes.
- *Edit Group* - edit group of nodes.
- *Ungroup* - ungroup the nodes.
- *Make Group* - make group of nodes.
- *Group Insert* - place the selected nodes in the selected group.

The following functions are located under the additional buttons on the bar of the node editor panel:



1. Go to the "parent" of the node on the tree.
2. Automatically extend nodes after adding a new node to an existing chain.
3. Snap to the grid.
4. Pull the node to: Grid, Node X, Node Y, Node X/Y.
5. 'Copy' and ' paste' the node.

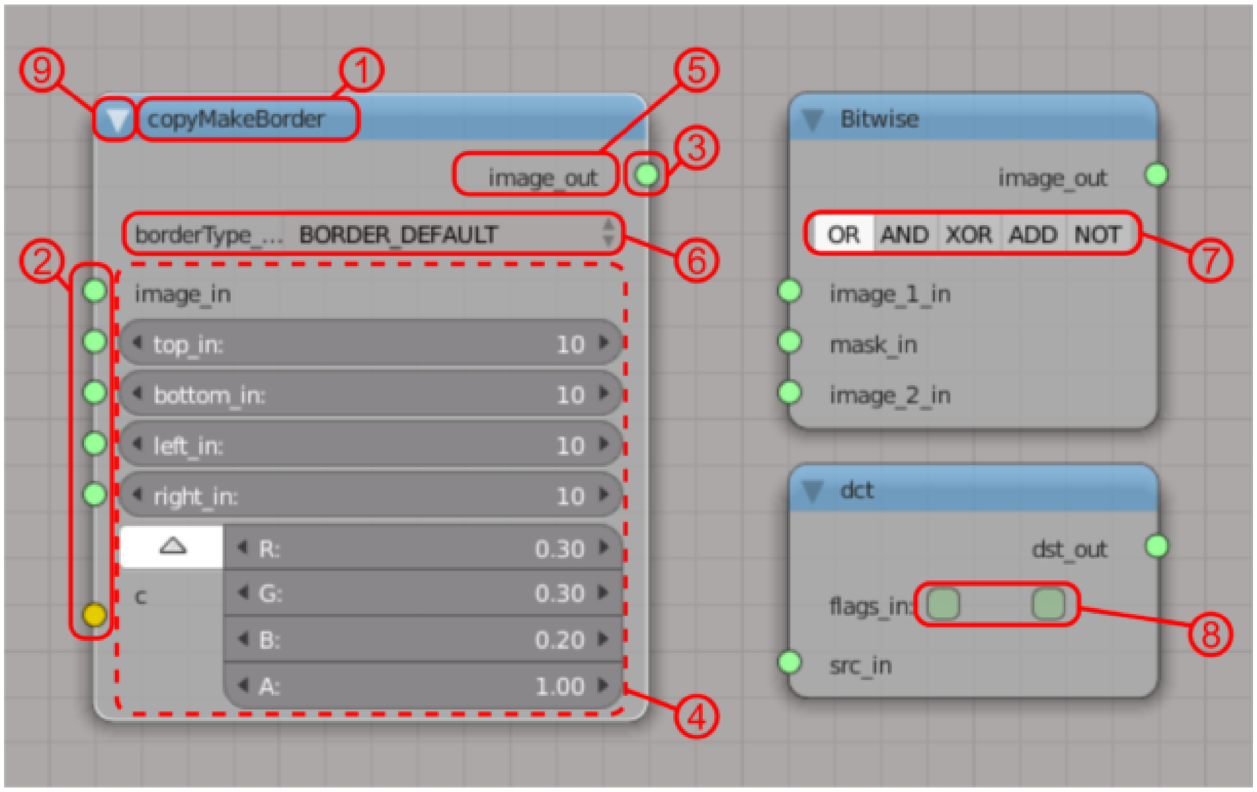
CHAPTER 5

Node

OpenCV Laboratory is a series of pre-installed Python libraries and a set of Blender extensions. On this basis, a set of OpenCV functions has been implemented in the form of convenient to connect nodes where we have quick and convenient access to all function parameters and an immediate preview of the effects of these functions.

The node is represented by a rectangle with rounded corners in OpenCV Laboratory. Each of them has a circular input socket(s) on the left and an output socket(s) on the opposite side.

The example with a brief description of different nodes is below:

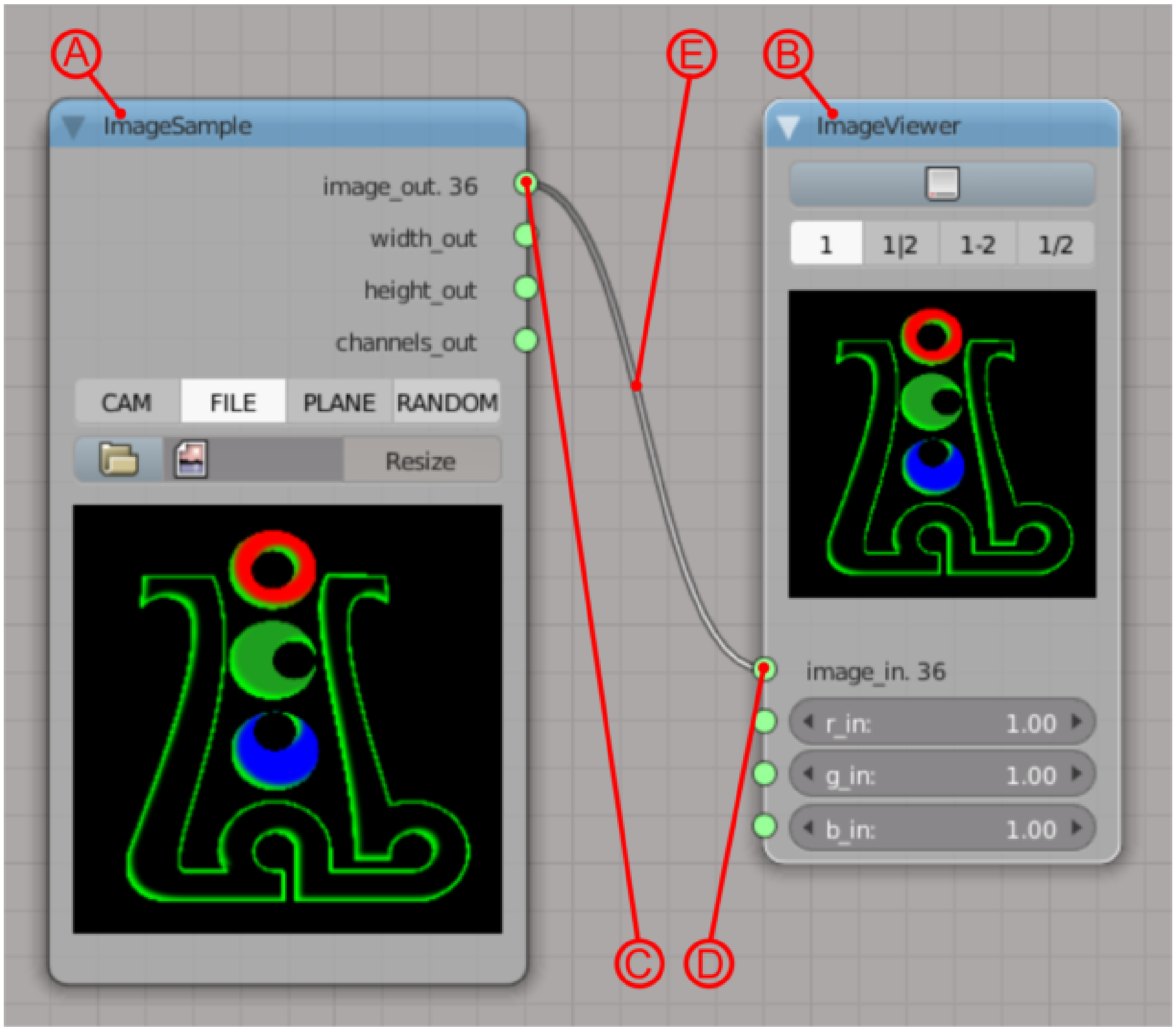


1. Nodes name.
2. Input sockets.
3. Output socket.
4. Input parameters to which the corresponding input sockets are assigned. It is often possible to freely adjust individual arguments using the sliders or by entering a specific value.
5. The output parameter is the result of operations performed by the node.
6. An internal parameter that is a list for selecting a specific function.
7. Internal parameter in the form of buttons defining the selection of functions.
8. An internal parameter in the form of an acceptance field taking into account the function.
9. A button to minimize the view of the node.

CHAPTER 6

Interconnecting of nodes

Interconnecting nodes is nothing more than a command for a computer program to perform the relevant functions, specifying the final result of their actions. Interconnecting individual nodes is a relatively simple procedure consisting in connecting, by means of a line, the output socket of one of the nodes with the appropriate input socket of another one.



A) First node

B) Second node

C) Output socket

D) Input socket

E) Interconnecting

The above operation is performed as follows:

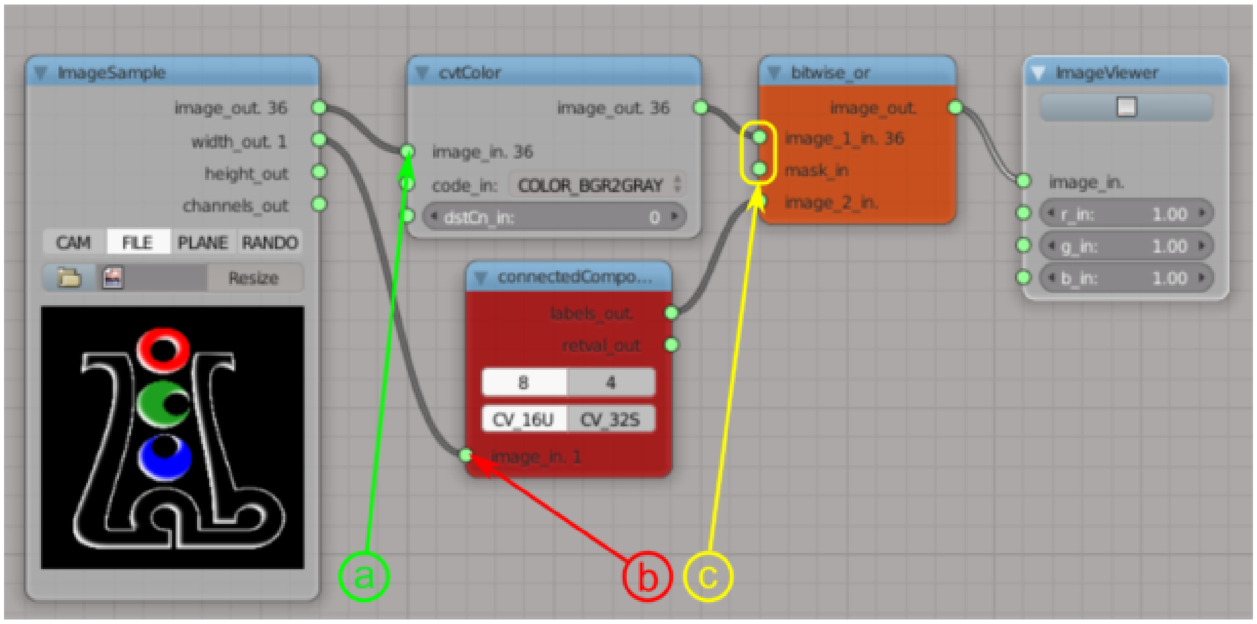
1. Left-click on the output socket of the node.
2. Without releasing the button, route the lines to the input of another node.
3. Release the mouse button.

No complex operations are applied in this example so resulting in an output image identical to the input image.

CHAPTER 7

Incorrect input data of a node

Not all nodes can be directly connected to each other due to their properties and functions. Each node requires a specific parameter to be entered at the appropriate input, some of which require all relevant data to be entered. Any irregularities in the above cases are illustrated by a change in the color of the node.



a) correct input data of the node

b) incorrect input data of the node

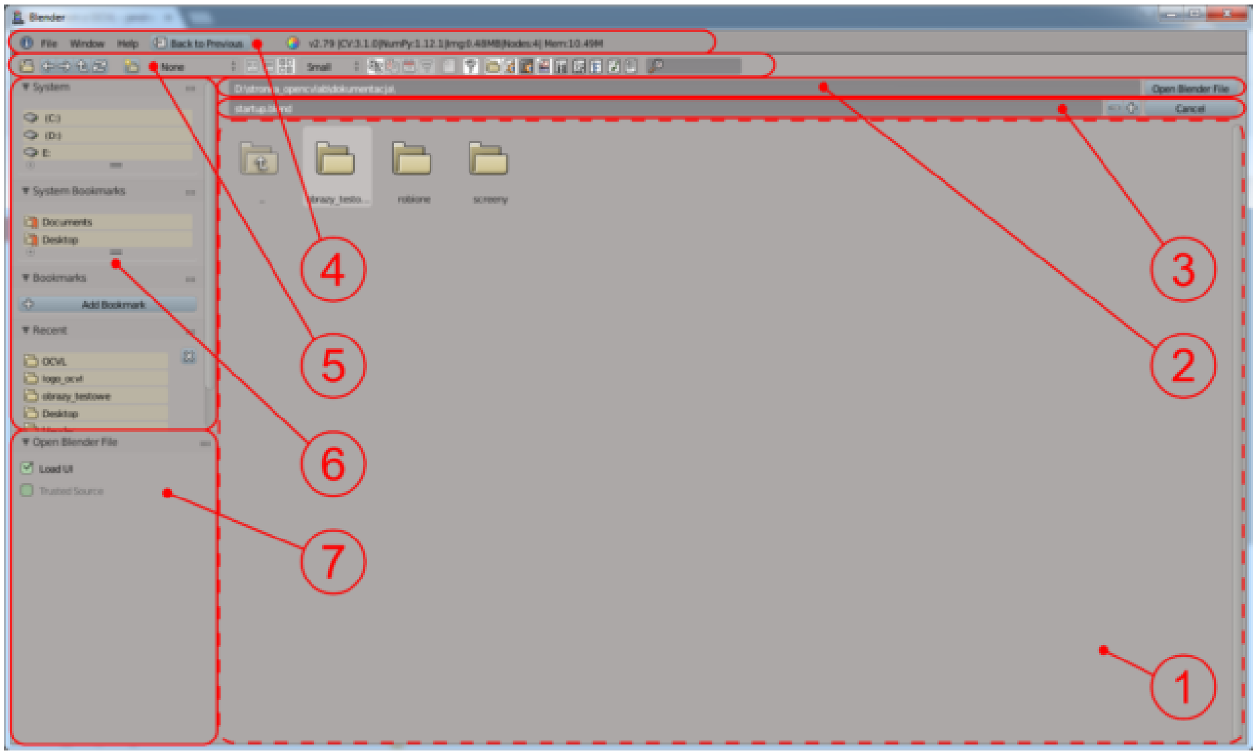
c) lack of necessary data at the input of the node

It is often enough to make a small correction to the node settings or to add the required data or to use a compilation of other nodes if necessary.

CHAPTER 8

File review mode

You can easily locate and read or save a specific file in file review mode. The file review mode contains: the file display area (1), the file path bar (2), the file name bar (3), the information panel bar (4), the bar of the file review panel (5), the tool panel (6) and the properties panel (7).



8.1 File display area

The file display area displays all available files.

8.2 File path bar

The file path bar consists:

- *File path*,
- *Open Image* - execute selected file.

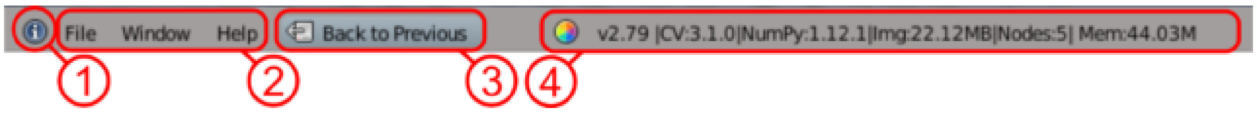
8.3 File name bar

The file name bar is included:

- *File name*,
- -/+ - buttons: decrement or increment the filename number,
- *Cancel* - cencel loading of selected file.

8.4 Information panel bar

Information panel bar includes a panel icon (1), a menu with drop-down lists (2), a return button to work mode (3), and information about the program version, number of nodes present in the work area and memory used by the current project (4).



v2.79 |CV:3.1.0|NumPy:1.12.1|Img:22.12MB|Nodes:5| Mem:44.03M

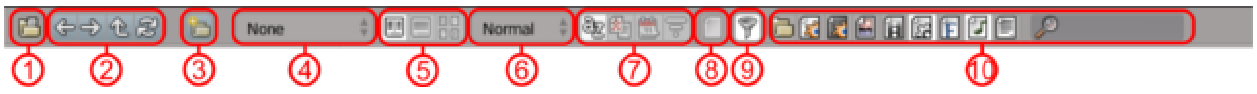
The menus are lists concerning: file operations, view options and help.

- **File - traditional document-related operations used in most application programs:**
  - *New* - open the default file (doesn't save the current file).
  - *Open* - open project.
  - *Open Recent…* - open one of the most recently used projects.
  - *Revent* - re-load the last saved version of the file.
  - *Recover Last Session* - open the last closed file.
  - *Recover Auto Save…* - open an automatically saved file to recover it.
  - *Save* - save the current file.
  - *Save As…* - save the current file in the desired location.
  - *Save Copy* - save as copy of the project.
  - *Quit* - as written.
- **Window - options for the view:**

- *Duplicate Window* - duplicate the current window.
- *Toggle Window Fullscreen* - toggle the current window fullscreen.
- *Save Screenshot* - capture a picture of the active area or whole window.
- *Toggle System Console* - console with logs.

- **Help - mainly relevant documentation:**
  - *OCVL Documentation*: <http://opencv-laboratory.readthedocs.io/en/latest/?badge=latest>
  - *OpenCV Documentation*: <http://kube.pl/>
  - *Blender Dokumentation*: <https://docs.blender.org/manual/en/dev/editors/node_editor/introduction.html>
  - *OCVL Web Panel*: <https://ocvl-cms.herokuapp.com/admin/login/>
  - *OCVL Blog*: <http://kube.pl/>
  - *Show Node Splash* - show a welcome window.

8.5 File review panel bar

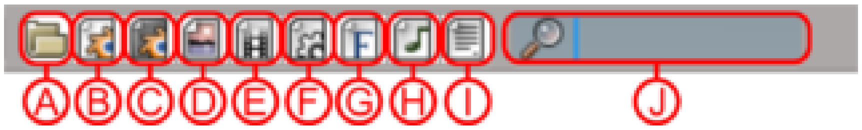


File review panel bar includes:

1. A panel icon.
2. **Passage buttons marked with arrow icons:**
   - *Move to previous folder*,
   - *Move to next folder*,
   - *Move to parent directory*,
   - *Refresh the file list*.
3. Button: *Create a new dirctory*.
4. ***Recursion* - numbers of dirtree levels to show simultaneously:**
   - *None* - only list current directory's content, with no recursion
   - *One Level* - list all sub-directories' content, one level of recursion
   - *Two Levels* - list all sub-directories' content, two levels of recursion
   - *Three Level* - list all sub-directories' content, three levels of recursion.
5. **File list display mode buttons:**
   - *Short List* - display files as short list,
   - *Long list* - display files as a detailed list,
   - *Thumbnails* - display files as thumbnails.
6. ***Display size* - change the size of the display (width or columns or thumbnails size):**
   - *Tiny*,

- *Small*,
- *Normal*,
- *Large*.

7. Type of sorting of the list of files in the form of buttons. Sorting options available in order: alphabetical, extension or type, time of modification, size.
8. Button: *Show hidden dot files*.
9. Button: *Enable filtering of files*.
10. File filtering options:

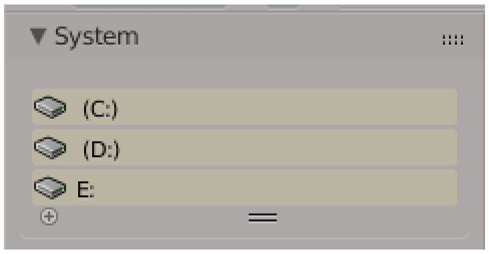


- *A* - show folders,
- *B* - show *.blend* files,
- *C* - show *.blend1*, *.blend2*, etc. files;
- *D* - show image files,
- *E* - show movie files,
- *F* - show script files,
- *G* - show font files,
- *H* - show sound files,
- *I* - show text files,
- *J* - filter by name, supports '*' wildcard.

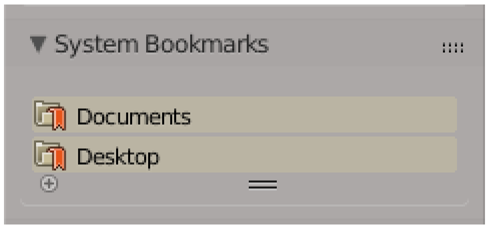
8.6 Tool panel

There are tabs with the selected file regionalization in the tool panel bar:

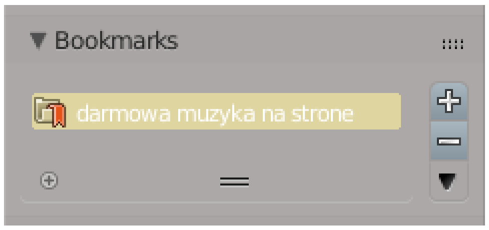
- *System* -



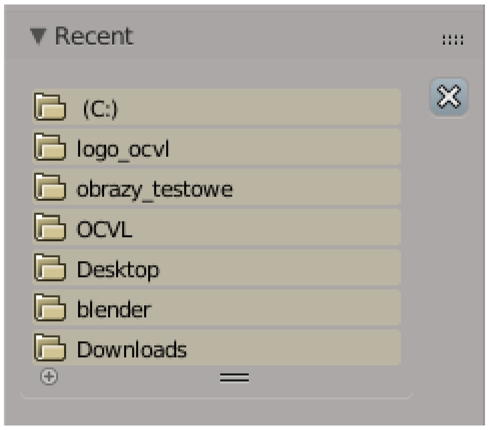
- *System Bookmarks* -



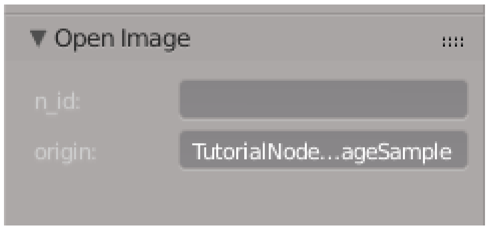
- *Bookmarks -*



- *Recent -*



8.7 Properties panel



n_id:

origin: TutorialNode...ageSample

CHAPTER 9

OpenCV Laboratory

9.1 core

9.1.1 KeyPoint

Functionality

The keypoint constructors

Inputs

- angle_in – Keypoint orientation.
- class_id_in – Object id.
- octave_in – Pyramid octave in which the keypoint has been detected.
- pt_in – The x & y coordinates of the keypoint.
- response_in – Keypoint detector response on the keypoint (that is, strength of the keypoint).
- size_in – Keypoint diameter.

Outputs

Locals

Examples

9.1.2 LUT

Functionality

Performs a look-up table transform of an array.

Inputs

- image_in – Input array of 8-bit elements.
- lut_in – Look-up table of 256 elements; in case of multi-channel input array, the table should either have a single channel (in this case the same table is used for all channels) or the same number of channels as in the input array.

Outputs

- image_out – Output array of the same size and number of channels as src, and the same depth as lut.

Locals

Examples

9.1.3 Mahalanobis

Functionality

Calculates the Mahalanobis distance between two vectors.

Inputs

- icovar_in – Inverse covariance matrix.
- v1_in – First 1D input vector.
- v2_in – Second 1D input vector.

Outputs

- retval_out – Return value.

Locals

Examples

9.1.4 Point

Functionality

Point.

Inputs

- x_in – X
- y_in – Y

Outputs

Locals

Examples

9.1.5 Point3

Functionality

Point 3

Inputs

- x_in – X
- y_in – Y
- z_in – Z

Outputs

Locals

Examples

9.1.6 Range

Functionality

Range.

Inputs

- end_in – End input.
- start_in – Start input.

Outputs

Locals

Examples

9.1.7 Rect

Functionality

Rect.

Inputs

- height_in – Height input.
- width_in – Width input.
- x_in – X input.
- y_in – Y input.

Outputs

Locals

Examples

9.1.8 RotatedRect

Functionality

Rotated Rect node.

Inputs

- angle_in – Angle input.
- center_in – Center input.
- size_in – Size input.

Outputs

Locals

Examples

9.1.9 Size

Functionality

Size node.

Inputs

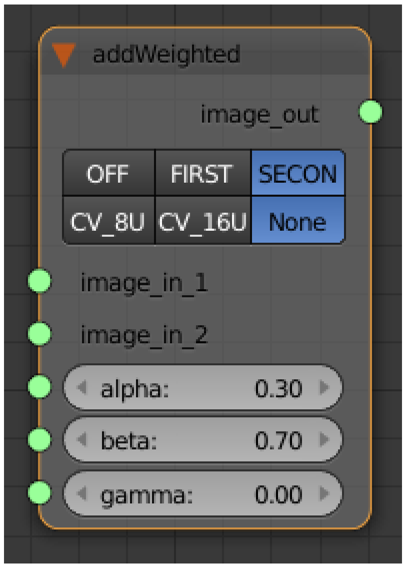
- height_in – Height input.
- width_in – Width input.

Outputs

Locals

Examples

9.1.10 addWeighted



Functionality

Calculates the weighted sum of two arrays.

Inputs

- alpha_in – Weight of the first array elements.
- beta_in – Weight of the second array elements.
- dtype_in – Desired depth of the destination image, see @ref filter_depths 'combinations'.
- gamma_in – Scalar added to each sum.
- image_1_in – First input array.
- image_2_in – Second input array.

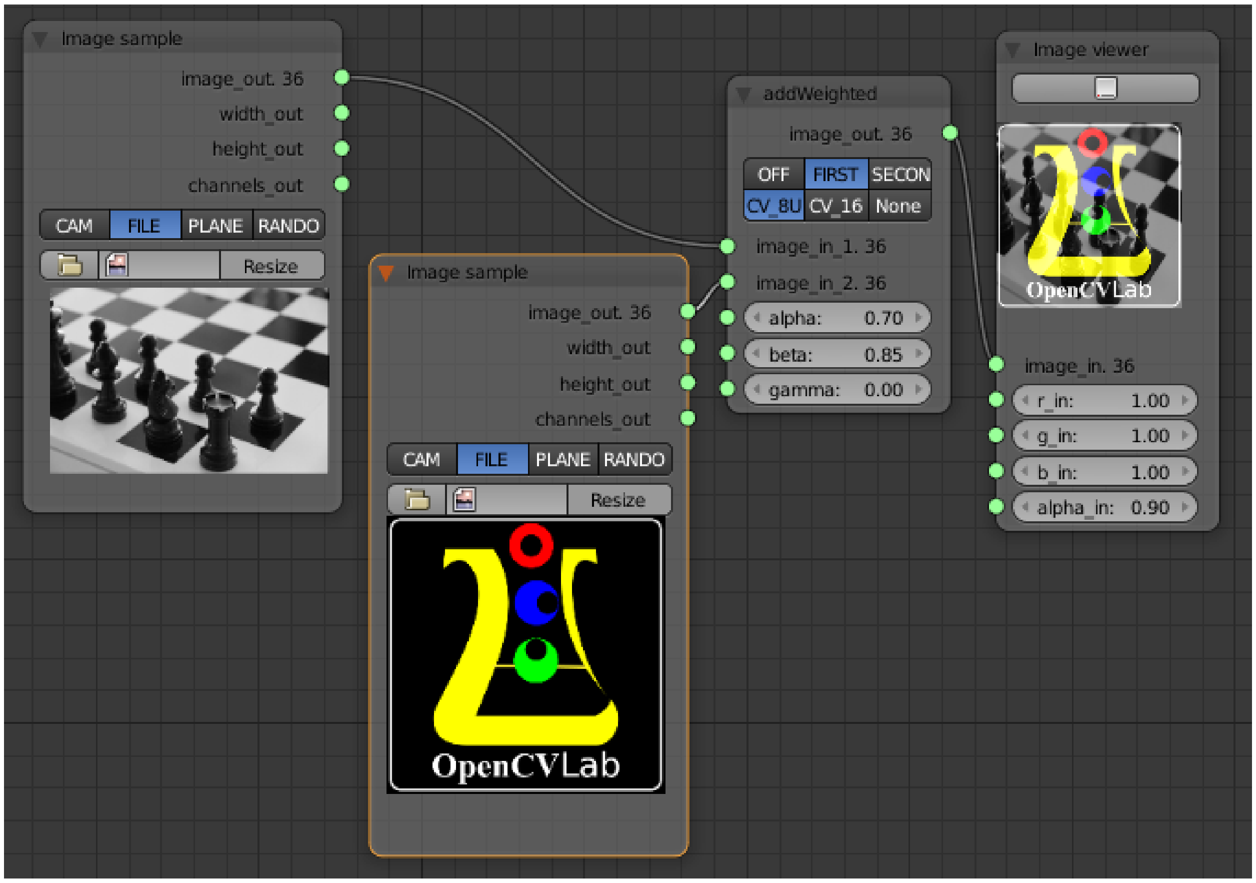
Outputs

- image_out – Output image.

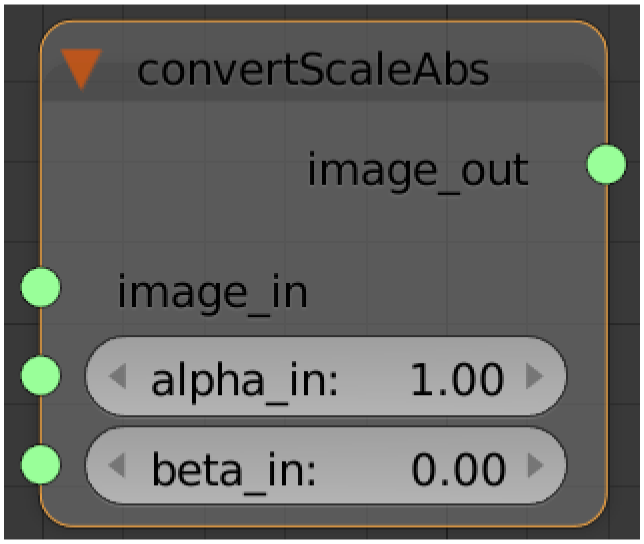
Locals

- loc_auto_resize – Automatic adjust size image.

Examples



9.1.11 convertScaleAbs



Functionality

Scales, calculates absolute values, and converts the result to 8-bit.

Inputs

- alpha_in – Optional scale factor.
- beta_in – Optional delta added to the scaled values.
- image_in – Input image.

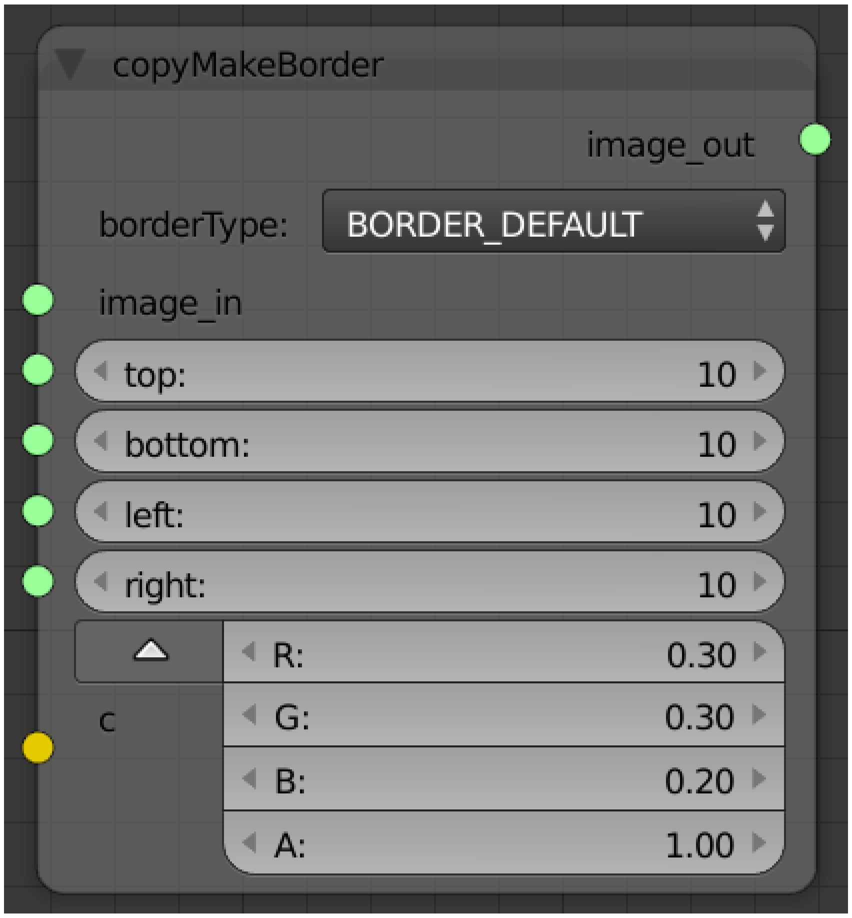
Outputs

- image_out – Output image.

Locals

Examples

9.1.12 copyMakeBorder



Functionality

Forms a border around an image.

Inputs

- borderType_in – Border type. See borderInterpolate for details.
- bottom_in – Border width in number of pixels in corresponding directions.
- color_in – Border value if borderType==BORDER_CONSTANT.
- image_in – Input image.
- left_in – Border width in number of pixels in corresponding directions.
- right_in – Border width in number of pixels in corresponding directions.
- top_in – Border width in number of pixels in corresponding directions.

Outputs

- image_out – Output image.

Locals

Examples

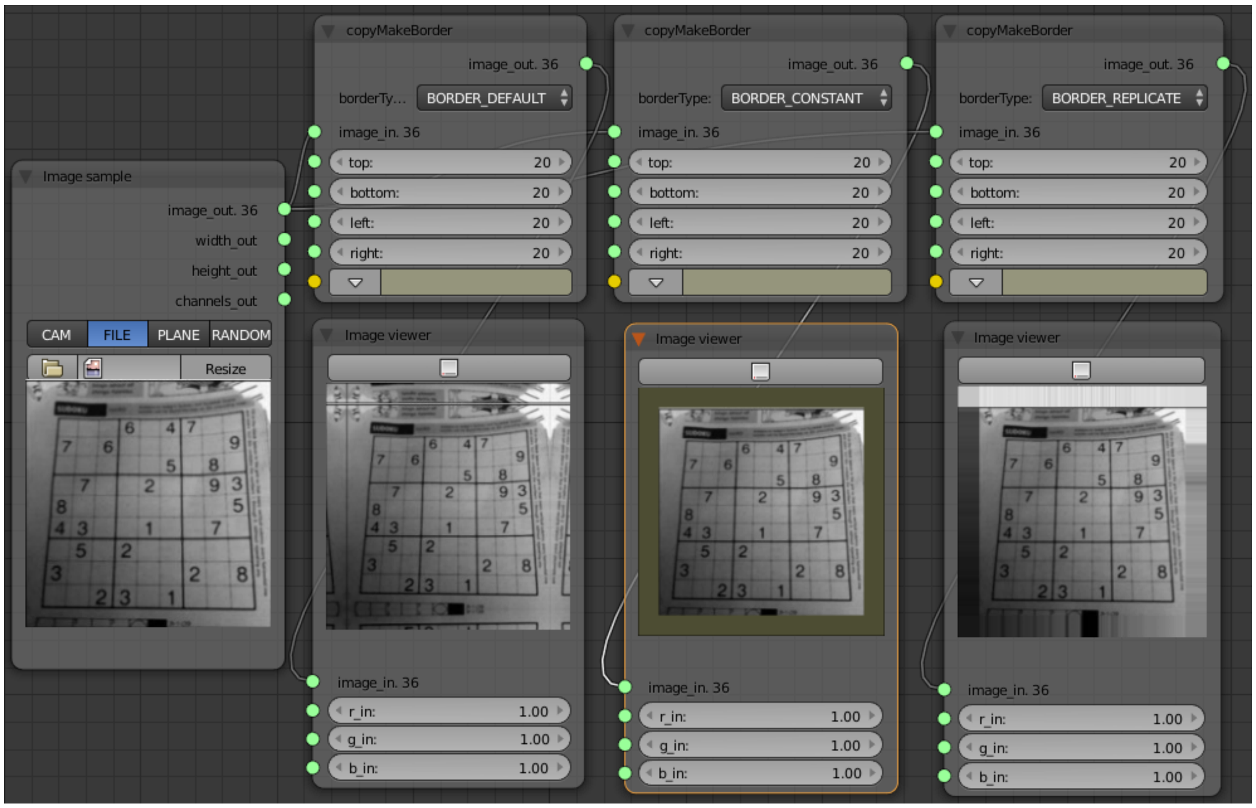


Image sample

image_out. 36

width_out

height_out

channels_out

CAM FILE PLANE RANDOM

Resize

copyMakeBorder

image_out. 36

borderTy… BORDER_DEFAULT

image_in. 36

top: 20

bottom: 20

left: 20

right: 20

copyMakeBorder

image_out. 36

borderType: BORDER_CONSTANT

image_in. 36

top: 20

bottom: 20

left: 20

right: 20

copyMakeBorder

image_out. 36

borderType: BORDER_REPLICATE

image_in. 36

top: 20

bottom: 20

left: 20

right: 20

Image viewer

image_in. 36

r_in: 1.00

g_in: 1.00

b_in: 1.00

Image viewer

image_in. 36

r_in: 1.00

g_in: 1.00

b_in: 1.00

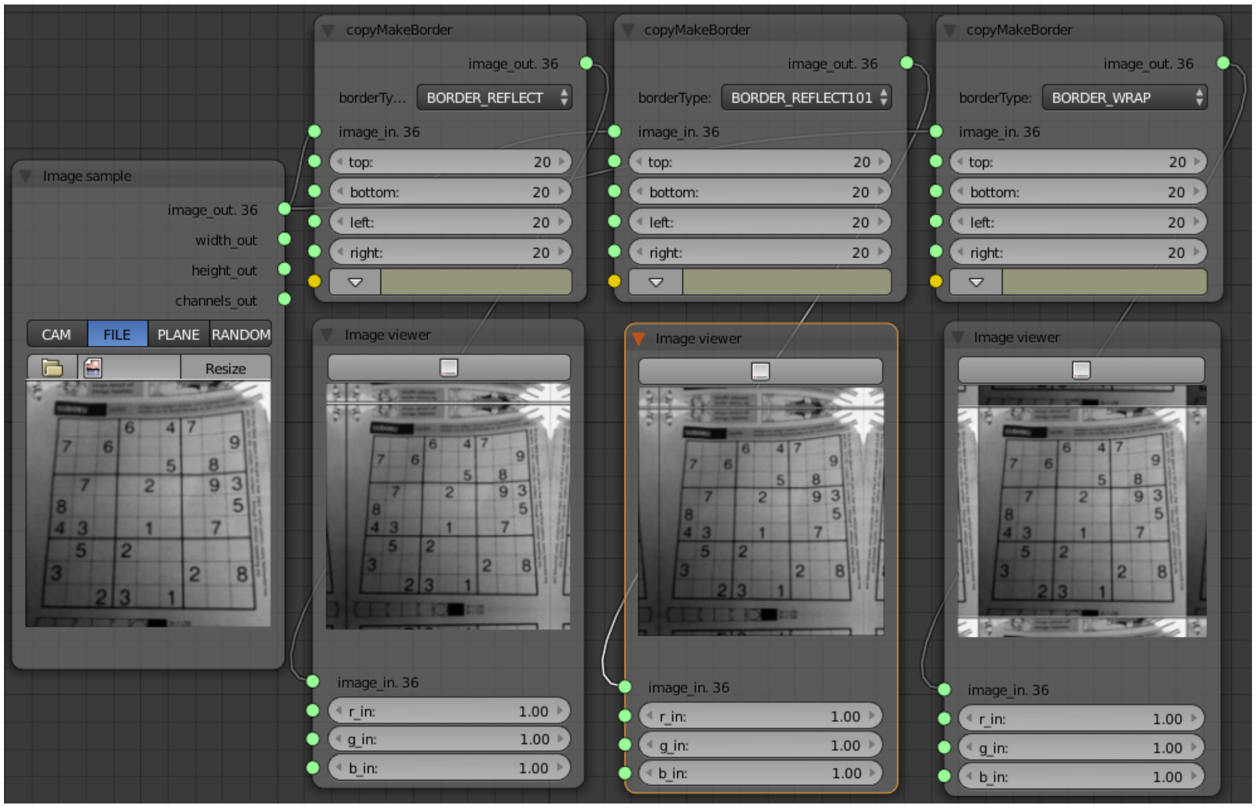
Image viewer

image_in. 36

r_in: 1.00

g_in: 1.00

b_in: 1.00



borderTy... BORDER_REFLECT
top: 20
bottom: 20
left: 20
right: 20

borderType: BORDER_REFLECT101
top: 20
bottom: 20
left: 20
right: 20

borderType: BORDER_WRAP
top: 20
bottom: 20
left: 20
right: 20

r_in: 1.00
g_in: 1.00
b_in: 1.00

9.1.13 dct

Functionality

Performs a forward or inverse discrete Cosine transform of 1D or 2D array.

Inputs

- flags_in – DCT_INVERSE, DCT_ROWS
- src_in – Input floating-point array.

Outputs

- dst_out – Output array of the same size and type as src .

Locals

Examples

9.1.14 dft

Functionality

Performs a forward or inverse Discrete Fourier transform of a 1D or 2D floating-point array.

Inputs

- flags_in – DFT_INVERSE, DFT_SCALE, DFT_ROWS, DFT_COMPLEX_OUTPUT, DFT_REAL_OUTPUT
- nonzeroRows_in – When the parameter is not zero, the function assumes that only the first nonzeroRows rows of the input array (DFT_INVERSE is not set) or only the first nonzeroRows of the output array (DFT_INVERSE is set) contain non-zeros.
- src_in – Input array that could be real or complex.

Outputs

- array_out – Output array whose size and type depends on the flags.

Locals

Examples

9.1.15 divide

Functionality

Performs per-element division of two arrays or a scalar by an array.

Inputs

- dtype_in – Desired depth of the destination image, see @ref filter_depths 'combinations'.
- scale_in – Scalar factor.
- src_1_in – First input array.
- src_2_in – Second input array of the same size and type as src1.

Outputs

- array_out – Output array.

Locals

Examples

9.1.16 eigen

Functionality

Calculates eigenvalues and eigenvectors of a symmetric matrix.

Inputs

- src_in – Input matrix that must have CV_32FC1 or CV_64FC1 type, square size and be symmetrical.

Outputs

- eigenvalues_out – Output vector of eigenvalues of the same type as src; the eigenvalues are stored in the descending order.
- eigenvectors_out – Output matrix of eigenvectors; it has the same size and type as src.
- retval_out – Return value.

Locals

Examples

9.1.17 exp

Functionality

Calculates the exponent of every array element.

Inputs

- src_in – Input array.

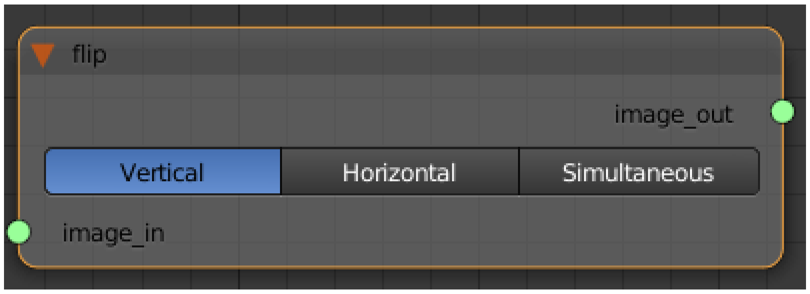
Outputs

- dst_out – Output array of the same size and type as input array.

Locals

Examples

9.1.18 flip



Functionality

Flips a 2D array around vertical, horizontal, or both axes.

Inputs

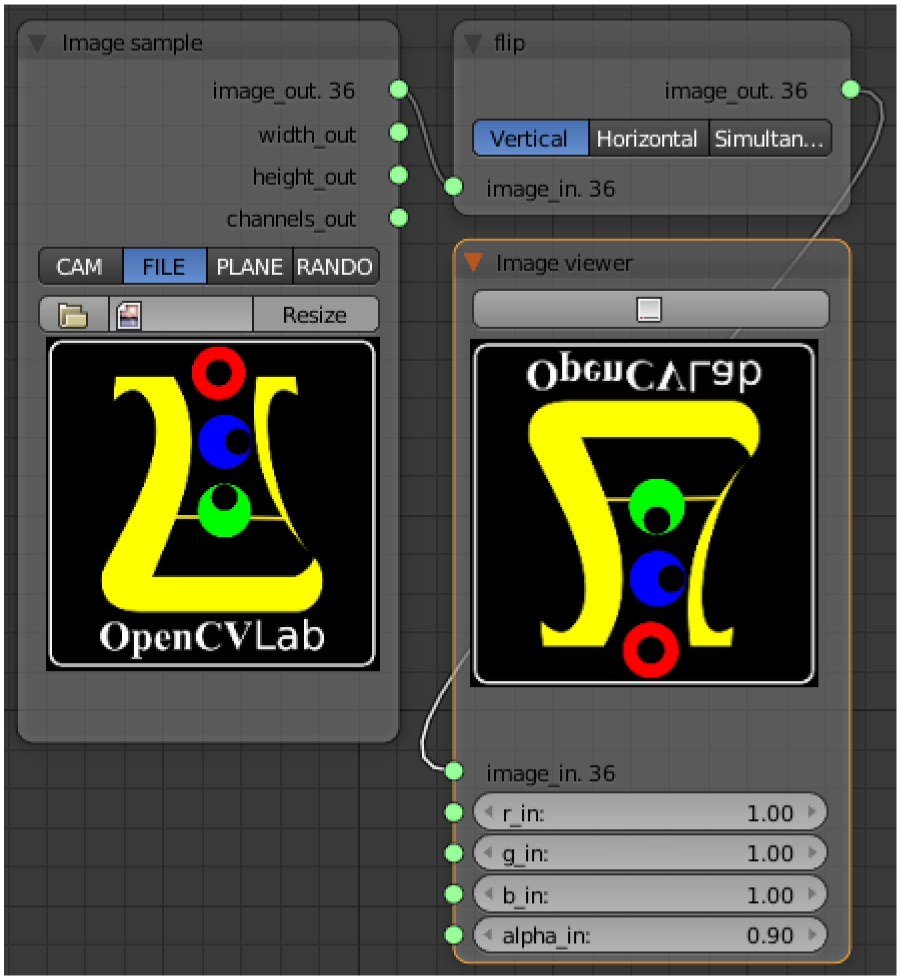
- flipCode_in – Flag to specify how to flip the array.
- image_in – Input array.

Outputs

- image_out – Output array of the same size and type as src.

Locals

Examples



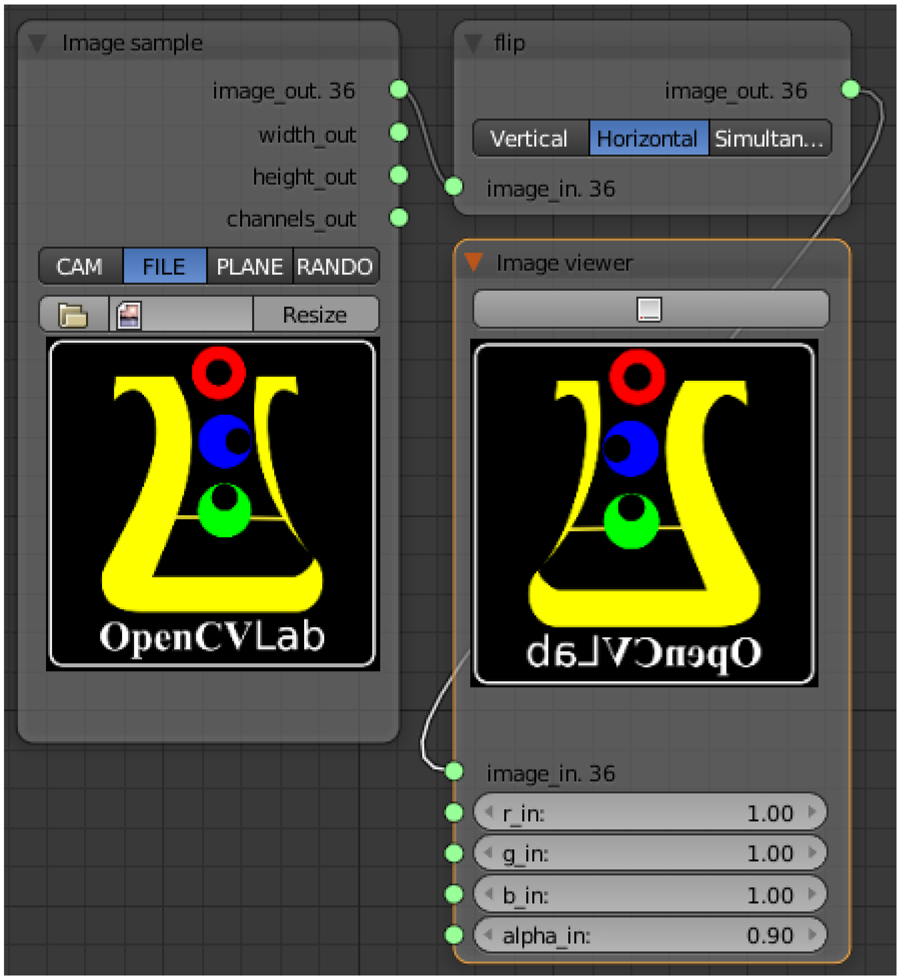
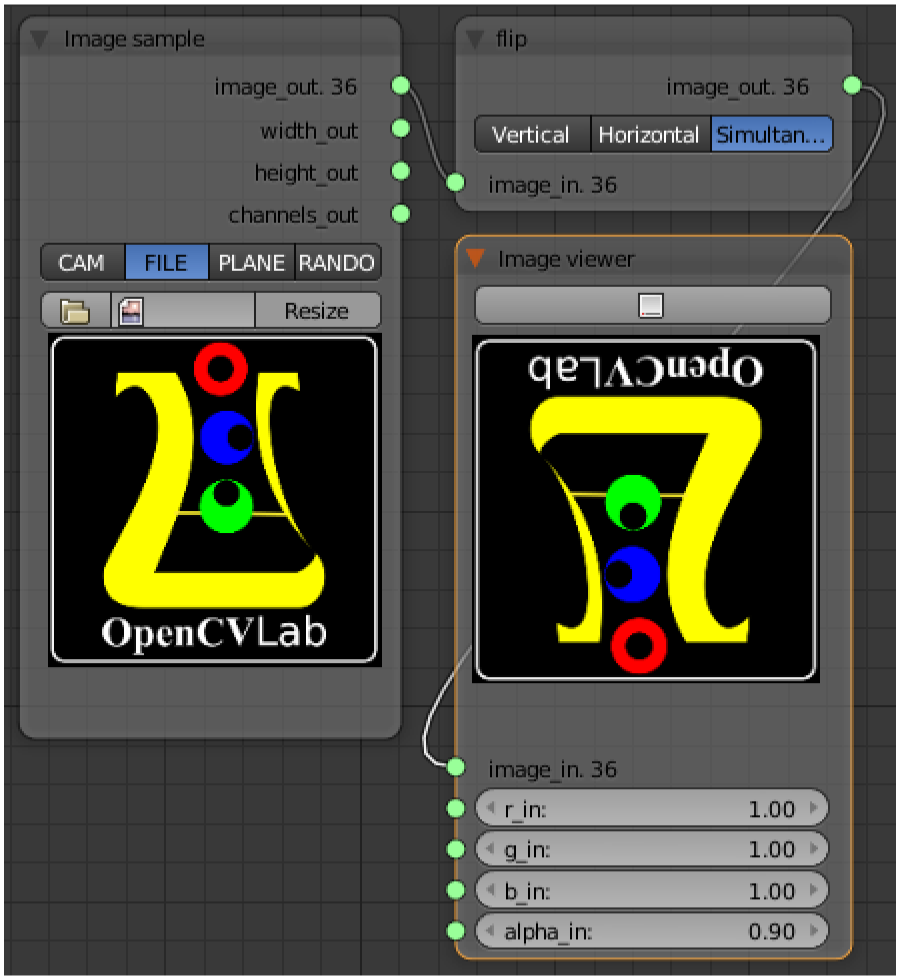


Image sample
image_out. 36
width_out
height_out
channels_out
CAM
FILE
PLANE
RANDO
Resize
OpenCVLab
flip
image_out. 36
Vertical
Horizontal
Simultan...
image_in. 36
Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90



9.1.19 gemm

Functionality

Performs generalized matrix multiplication.

Inputs

- alpha_in – Weight of the matrix product.
- beta_in – Weight of src3.
- flags_in – GEMM_1_T, GEMM_2_T, GEMM_3_T
- src_1_in – First multiplied input matrix that could be real(CV_32FC1, CV_64FC1) or complex(CV_32FC2, CV_64FC2).
- src_2_in – Second multiplied input matrix of the same type as src1.
- src_3_in – Third optional delta matrix added to the matrix product; it should have the same type as src1 and src2.

Outputs

- dst_out – Output matrix; it has the proper size and the same type as input matrices.

Locals

Examples

9.1.20 idct

Functionality

Calculates the inverse Discrete Cosine Transform of a 1D or 2D array.

Inputs

- flags_in – DCT_INVERSE, DFT_SCALE, DFT_ROWS, DFT_COMPLEX_OUTPUT, DFT_REAL_OUTPUT
- src_in – Input floating-point single-channel array.

Outputs

- dst_out – Output array of the same size and type as src.

Locals

Examples

9.1.21 idft

Functionality

Calculates the inverse Discrete Fourier Transform of a 1D or 2D array.

Inputs

- flags_in – DFT_INVERSE, DFT_SCALE, DFT_ROWS, DFT_COMPLEX_OUTPUT, DFT_REAL_OUTPUT
- nonzeroRows_in – Number of dst rows to process.
- src_in – Input floating-point real or complex array.

Outputs

- dst_out – Output array whose size and type depend on the flags.

Locals

Examples

9.1.22 inRange

Functionality

Checks if array elements lie between the elements of two other arrays.

Inputs

- lowerb_in – Inclusive lower boundary array or a scalar.
- src_in – First input array.
- upperb_in – Inclusive upper boundary array or a scalar.

Outputs

- dst_out – Output array of the same size as src and CV_8U type.

Locals

Examples

9.1.23 invert

Functionality

Finds the inverse or pseudo-inverse of a matrix.

Inputs

- flags_in – DECOMP_LU, DECOMP_SVD, DECOMP_CHOLESKY
- src_in – Input floating-point M x N matrix.

Outputs

- dst_out – Output matrix of N x M size and the same type as src.
- retval_out – Return value.

Locals

Examples

9.1.24 log

Functionality

Calculates the natural logarithm of every array element.

Inputs

- array_in – Input array.

Outputs

- array_out – Iutput array of the same size and type as input array .

Locals

Examples

9.1.25 magnitude

Functionality

Calculates the magnitude of 2D vectors.

Inputs

- x_in – Floating-point array of x-coordinates of the vectors.
- y_in – Floating-point array of y-coordinates of the vectors; it must have the same size as x.

Outputs

- array_out – Output array of the same size and type as x.

Locals

Examples

9.1.26 max

Functionality

Calculates per-element maximum of two arrays or an array and a scalar.

Inputs

- src1_in – First input array.
- src2_in – Second input array of the same size and type as src1.

Outputs

- array_out – Output array of the same size and type as src1.

Locals

Examples

9.1.27 mean

Functionality

Calculates an average (mean) of array elements.

Inputs

- mask_in – Optional operation mask.
- src_in – Input array that should have from 1 to 4 channels so that the result can be stored in **Scalar_**.

Outputs

- mean_out – Output parameter: calculated mean value.

Locals

Examples

9.1.28 meanStdDev

Functionality

Calculates a mean and standard deviation of array elements.

Inputs

- mask_in – Optional operation mask.
- src_in – Input array that should have from 1 to 4 channels so that the results can be stored in **Scalar_** ‘s.

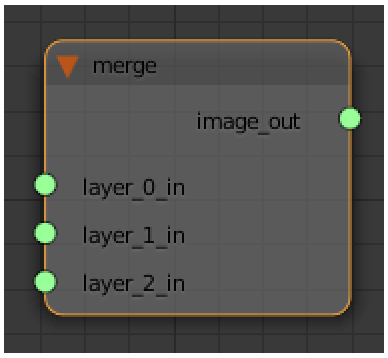
Outputs

- mean_out – Output parameter: calculated mean value.
- stddev_out – Output parameter: calculateded standard deviation.

Locals

Examples

9.1.29 merge



Functionality

Creates one multichannel array out of several single-channel ones.

Inputs

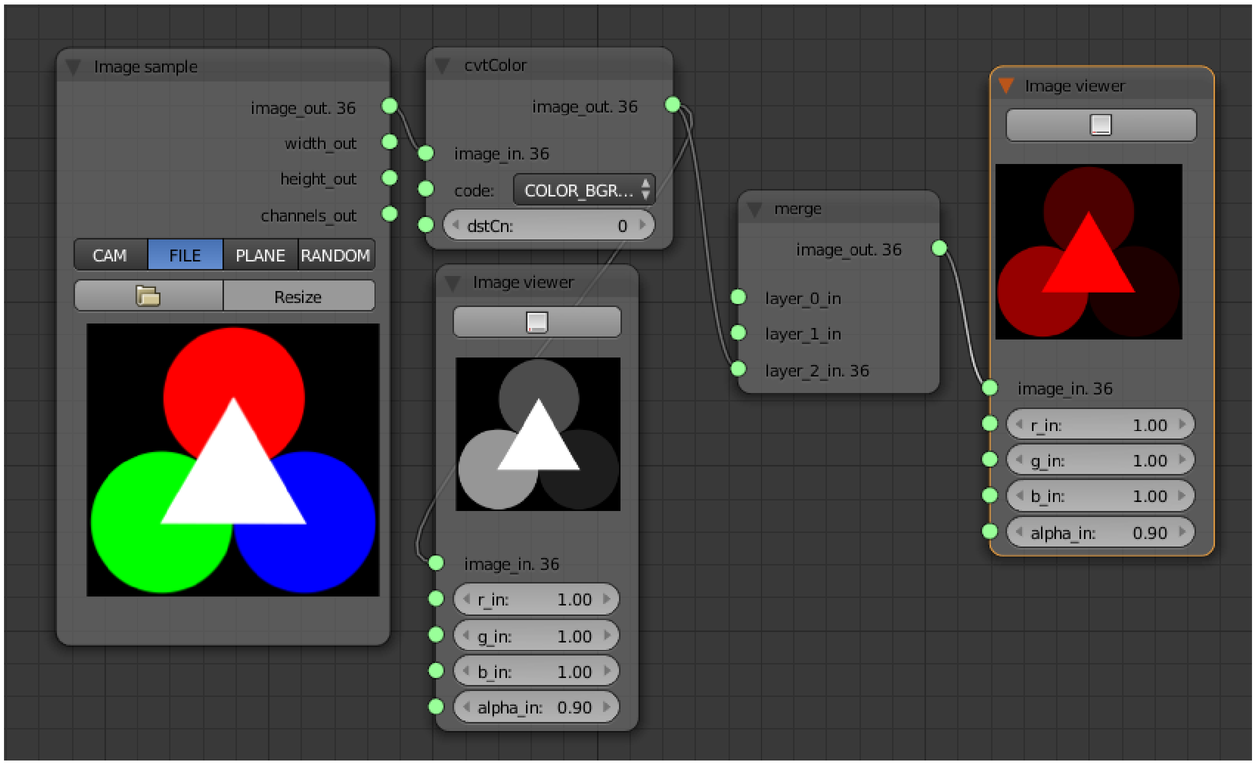
- layer_0_in – First channel Blue.
- layer_1_in – Second channel Green.
- layer_2_in – Third channel Red.

Outputs

- image_out – Image output.

Locals

Examples



9.1.30 min

Functionality

Calculates per-element minimum of two arrays or an array and a scalar.

Inputs

- src1_in – First input array.
- src2_in – Second input array of the same size and type as src1.

Outputs

- array_out – Output array of the same size and type as src1.

Locals

Examples

9.1.31 minMaxLoc

Functionality

Finds the global minimum and maximum in an array.

Inputs

- mask_in – Optional mask used to select a sub-array.
- src_in – Input single-channel array.

Outputs

- maxLoc_out – Pointer to the returned maximum location (in 2D case); NULL is used if not required.
- maxVal_out – Pointer to the returned maximum value; NULL is used if not required.
- minLoc_out – Pointer to the returned minimum location (in 2D case); NULL is used if not required.
- minVal_out – Pointer to the returned minimum value; NULL is used if not required.

Locals

Examples

9.1.32 mixChannels

Functionality

Copies specified channels from input arrays to the specified channels of output arrays.

Inputs

- fromTo_in – Array of index pairs specifying which channels are copied and where.
- src_in – Input array or vector of matrices; all of the matrices must have the same size and the same depth.

Outputs

- image_out – Output array or vector of matrices.

Locals

Examples

9.1.33 mulSpectrums

Functionality

Performs the per-element multiplication of two Fourier spectrums.

Inputs

- a_in – First input array.
- b_in – Second input array of the same size and type as src1.
- conjB_in – Optional flag that conjugates the second input array before the multiplication (true) or not (false).
- flags_in – DFT_ROWS

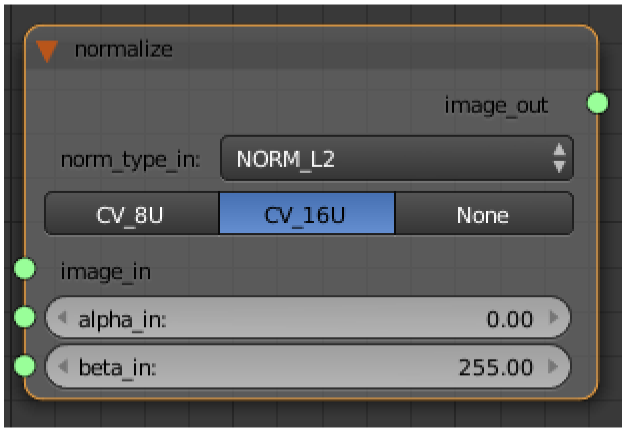
Outputs

- image_out – Output array.

Locals

Examples

9.1.34 normalize



norm_type_in: NORM_L2
CV_8U CV_16U None
alpha_in: 0.00
beta_in: 255.00

Functionality

Normalizes the norm or value range of an array.

Inputs

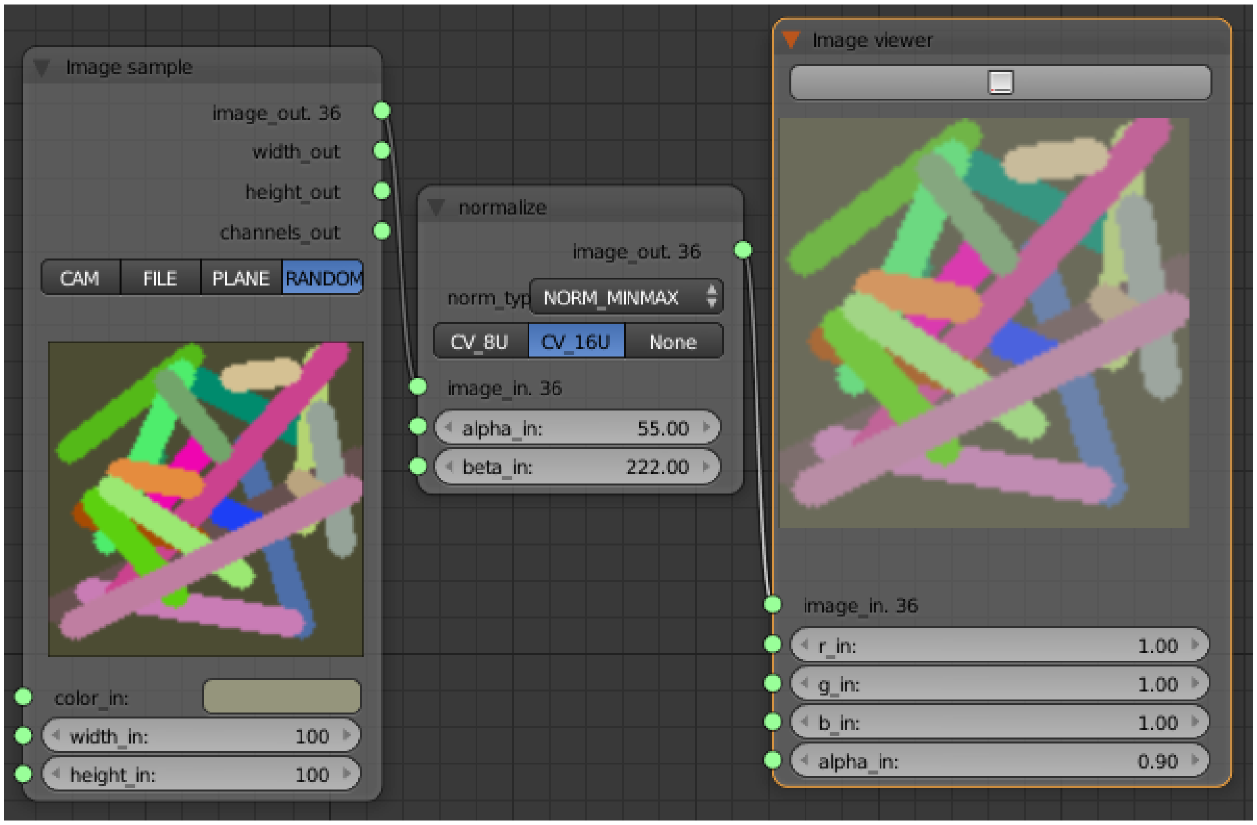
- alpha_in – Norm value to normalize to or the lower range boundary in case of the range normalization.
- beta_in – Upper range boundary in case of the range normalization; it is not used for the norm normalization.
- dtype_in – Channels as src and the depth =CV_MAT_DEPTH(dtype).
- image_in – Input array.
- norm_type_in – Normalization type (see cv::NormTypes).

Outputs

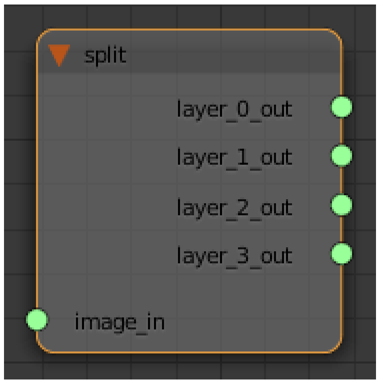
- image_out – Output array.

Locals

Examples



9.1.35 split



Functionality

Divides a multi-channel array into several single-channel arrays.

Inputs

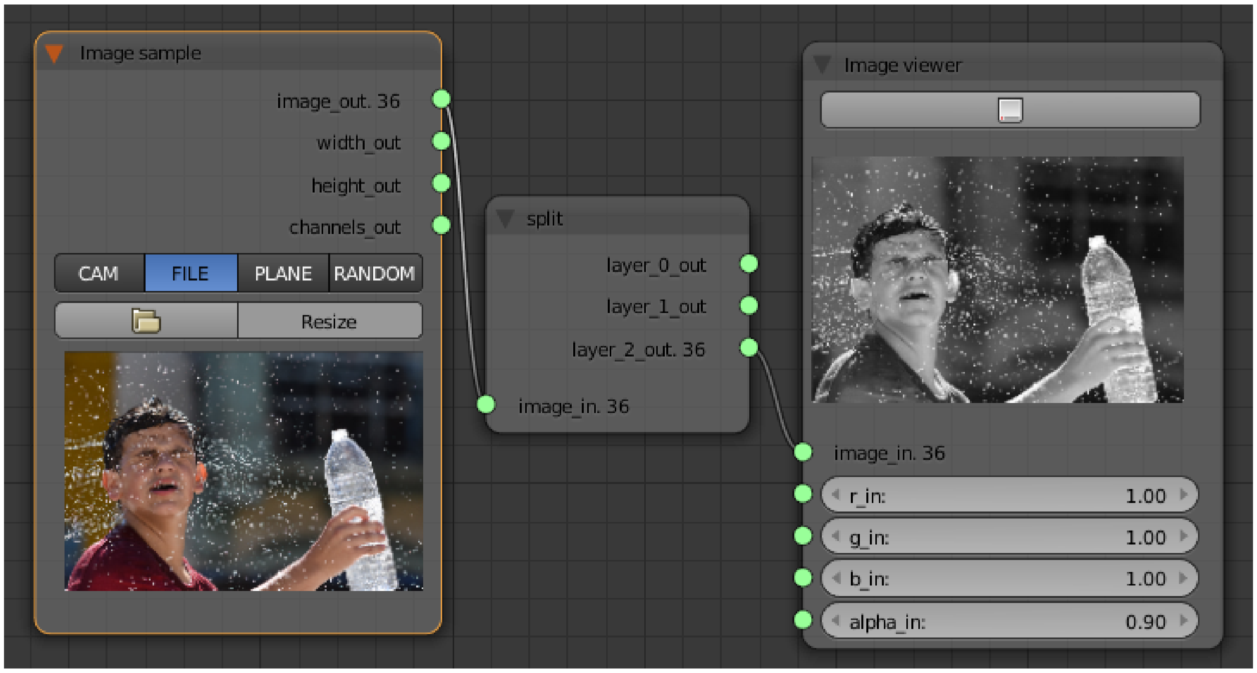
- image_in – Input multi-channel array.

Outputs

- layer_0_out – Channel 0.
- layer_1_out – Channel 1.
- layer_2_out – Channel 2.
- layer_3_out – Channel 3.

Locals

Examples



9.2 imgproc

9.2.1 GaussianBlur

Functionality

Blurs an image using a Gaussian filter.

Inputs

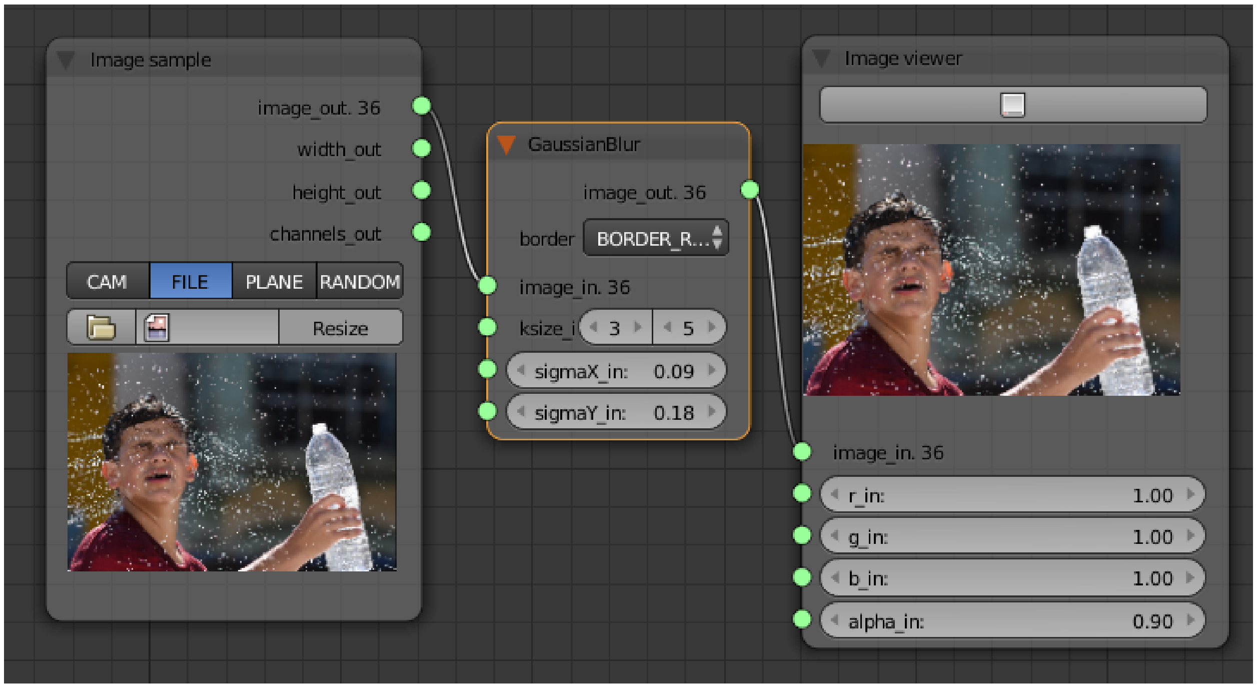
- borderType_in – Pixel extrapolation method, see cv::BorderTypes.
- image_in – Input image.
- ksize_in – Gaussian kernel size.
- sigmaX_in – Gaussian kernel standard deviation in X direction.
- sigmaY_in – Gaussian kernel standard deviation in Y direction.

Outputs

- image_out – Output image.

Locals

Examples



9.2.2 HoughLines

Functionality

Finds lines in a binary image using the standard Hough transform.

Inputs

- image_in – Input image.
- max_theta_in – For standard and multi-scale Hough transform, maximum angle to check for lines.

- min_theta_in – For standard and multi-scale Hough transform, minimum angle to check for lines.
- rho_in – Distance resolution of the accumulator in pixels.
- srn_in – For the multi-scale Hough transform, it is a divisor for the distance resolution rho.
- stn_in – For the multi-scale Hough transform, it is a divisor for the distance resolution theta.
- theta_in – Angle resolution of the accumulator in radians.
- threshold_in – Accumulator threshold parameter.

Outputs

- image_out – Output image.
- lines_out – Output vector of lines.

Locals

- loc_output_mode – Output mode.

Examples

9.2.3 HoughLinesP

Functionality

Finds line segments in a binary image using the probabilistic Hough transform.

Inputs

- image_in – Input image.
- maxLineGap_in – Maximum allowed gap between points on the same line to link them.
- minLineLength_in – Minimum line length. Line segments shorter than that are rejected.
- rho_in – Distance resolution of the accumulator in pixels.
- theta_in – Angle resolution of the accumulator in radians.
- threshold_in – Accumulator threshold parameter.

Outputs

- image_out – Output image.
- lines_out – Output vector of lines.

Locals

- loc_output_mode – Output mode.

Examples

9.2.4 Laplacian

Functionality

Calculates the Laplacian of an image.

Inputs

- borderType_in – Pixel extrapolation method, see cv::BorderTypes.
- ddepth_in – Desired depth of the destination image.
- delta_in – Optional delta value that is added to the results prior to storing them in dst.
- image_in – Input image.
- ksize_in – Aperture size used to compute the second-derivative filters.
- scale_in – Optional scale factor for the computed Laplacian values.

Outputs

- image_out – Output image.

Locals

Examples

9.2.5 Scharr

Functionality

Calculates the first x- or y- image derivative using Scharr operator.

Inputs

- borderType_in – Pixel extrapolation method, see cv::BorderTypes
- ddepth_in – Output image depth, see @ref filter_depths 'combinations'.
- delta_in – Optional delta value that is added to the results prior to storing them in dst.
- dx_in – Order of the derivative x.
- dy_in – Order of the derivative y.
- image_in – Input image.
- scale_in – Optional scale factor for the computed Laplacian values.

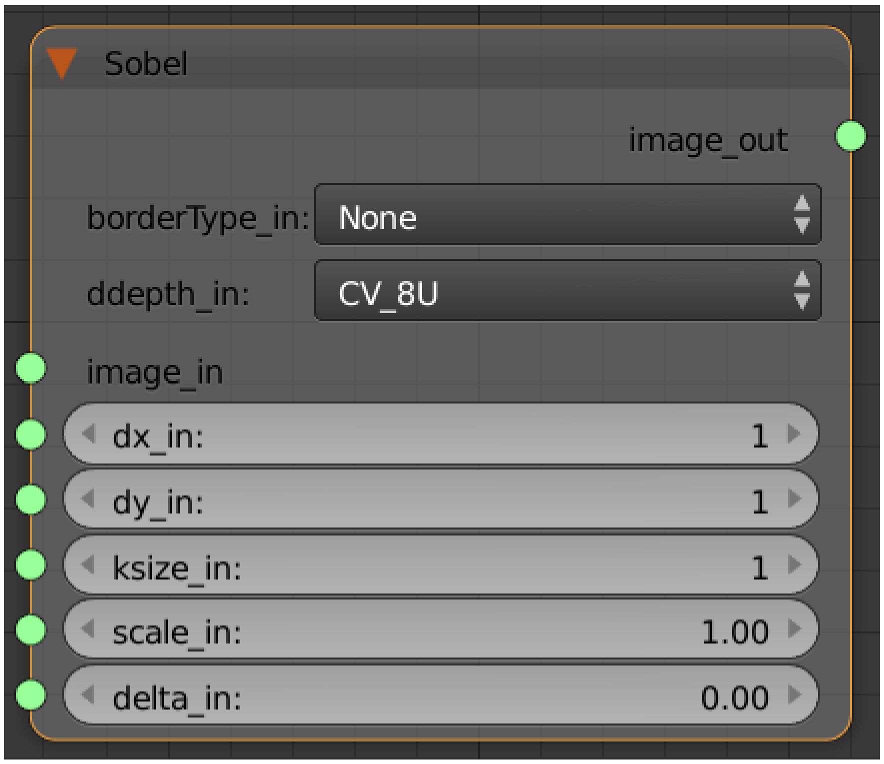
Outputs

- image_out – Output image.

Locals

Examples

9.2.6 Sobel


Sobel
image_out
borderType_in: None
ddepth_in: CV_8U
image_in
dx_in: 1
dy_in: 1
ksize_in: 1
scale_in: 1.00
delta_in: 0.00

Functionality

Calculates the first, second, third, or mixed image derivatives using an extended Sobel operator.

Inputs

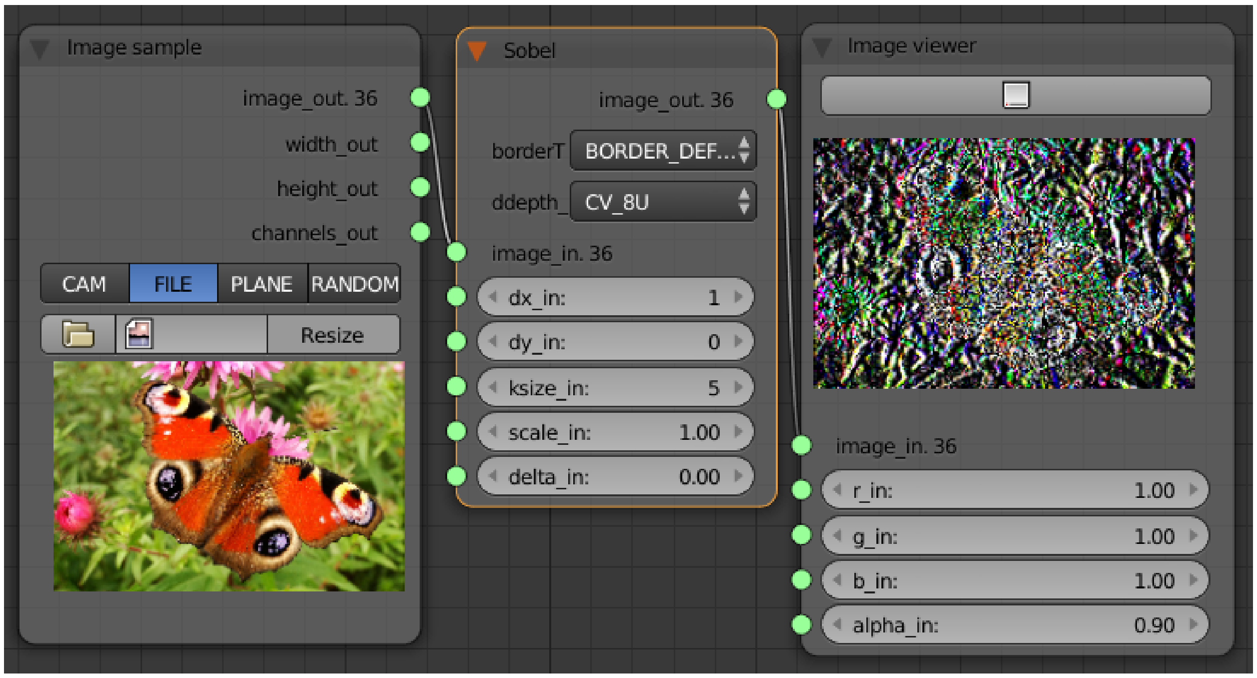
- borderType_in – Pixel extrapolation method, see cv::BorderTypes.
- ddepth_in – Desired depth of the destination image.
- delta_in – Optional delta value that is added to the results prior to storing them in dst.
- dx_in – Order of the derivative x.
- dy_in – Order of the derivative y.
- image_in – Input image.
- ksize_in – Aperture size used to compute the second-derivative filters.
- scale_in – Optional scale factor for the computed Laplacian values.

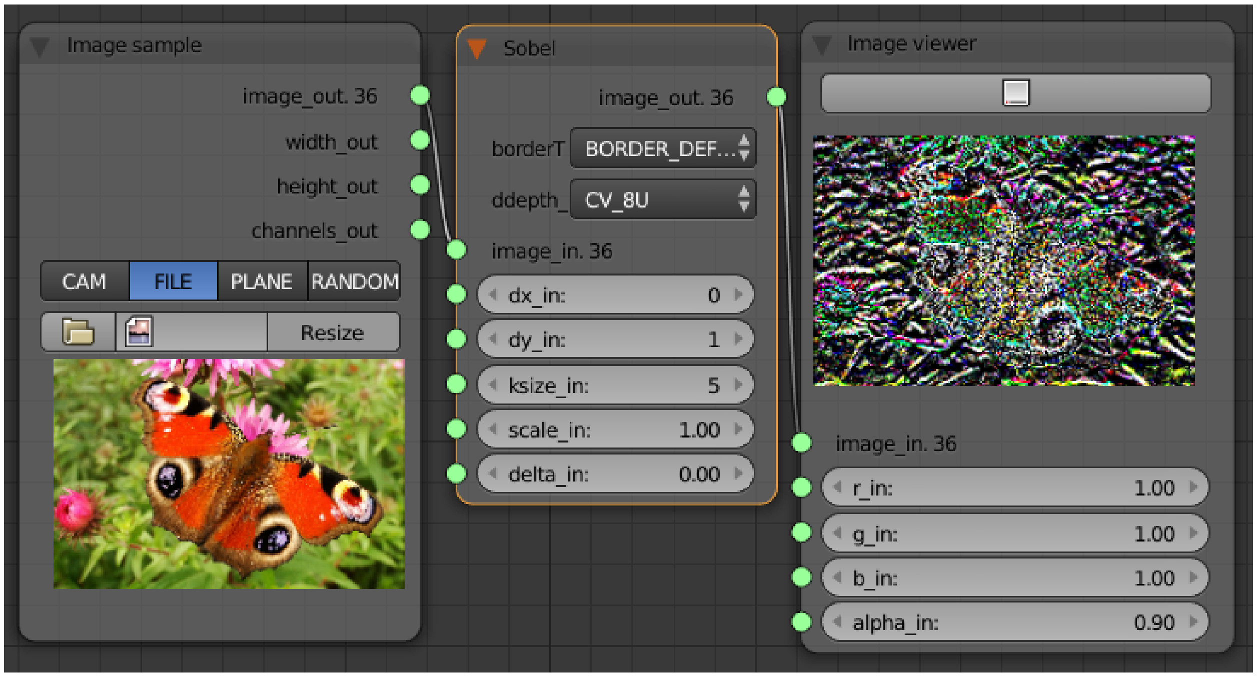
Outputs

- image_out – Output image.

Locals

Examples





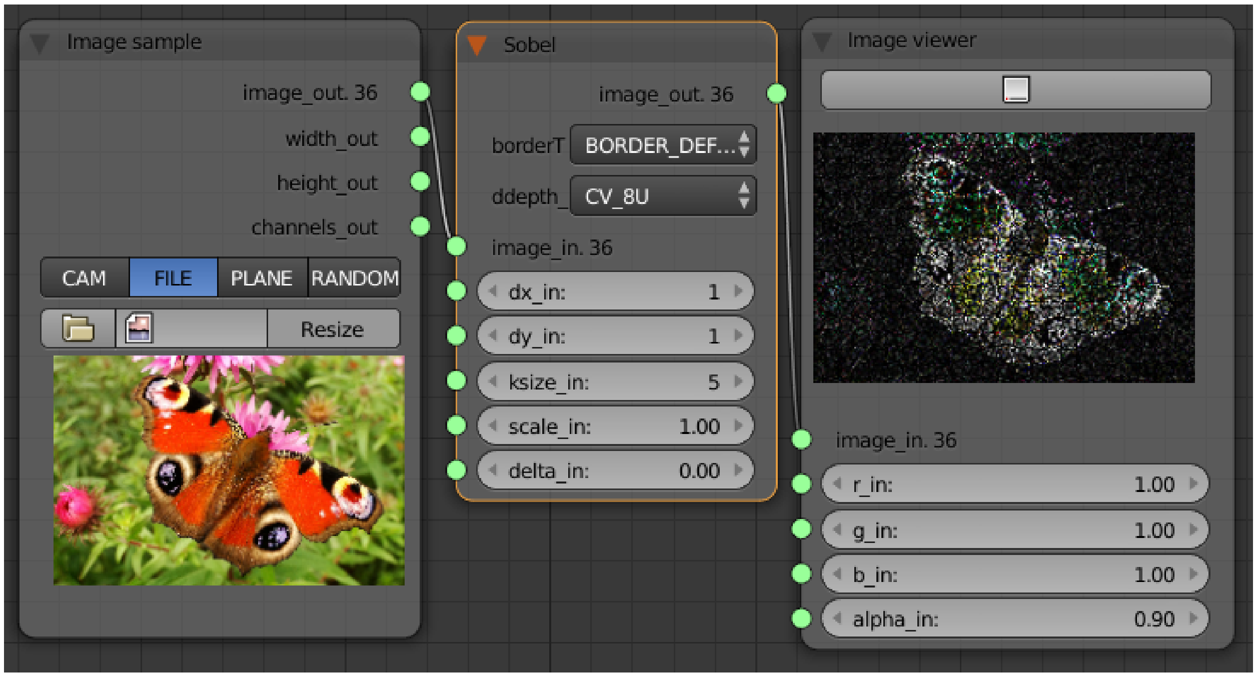


Image sample
image_out. 36
width_out
height_out
channels_out

Sobel
image_out. 36
borderT BORDER_DEF...
ddepth CV_8U
image_in. 36
dx_in: 1
dy_in: 1
ksize_in: 5
scale_in: 1.00
delta_in: 0.00

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

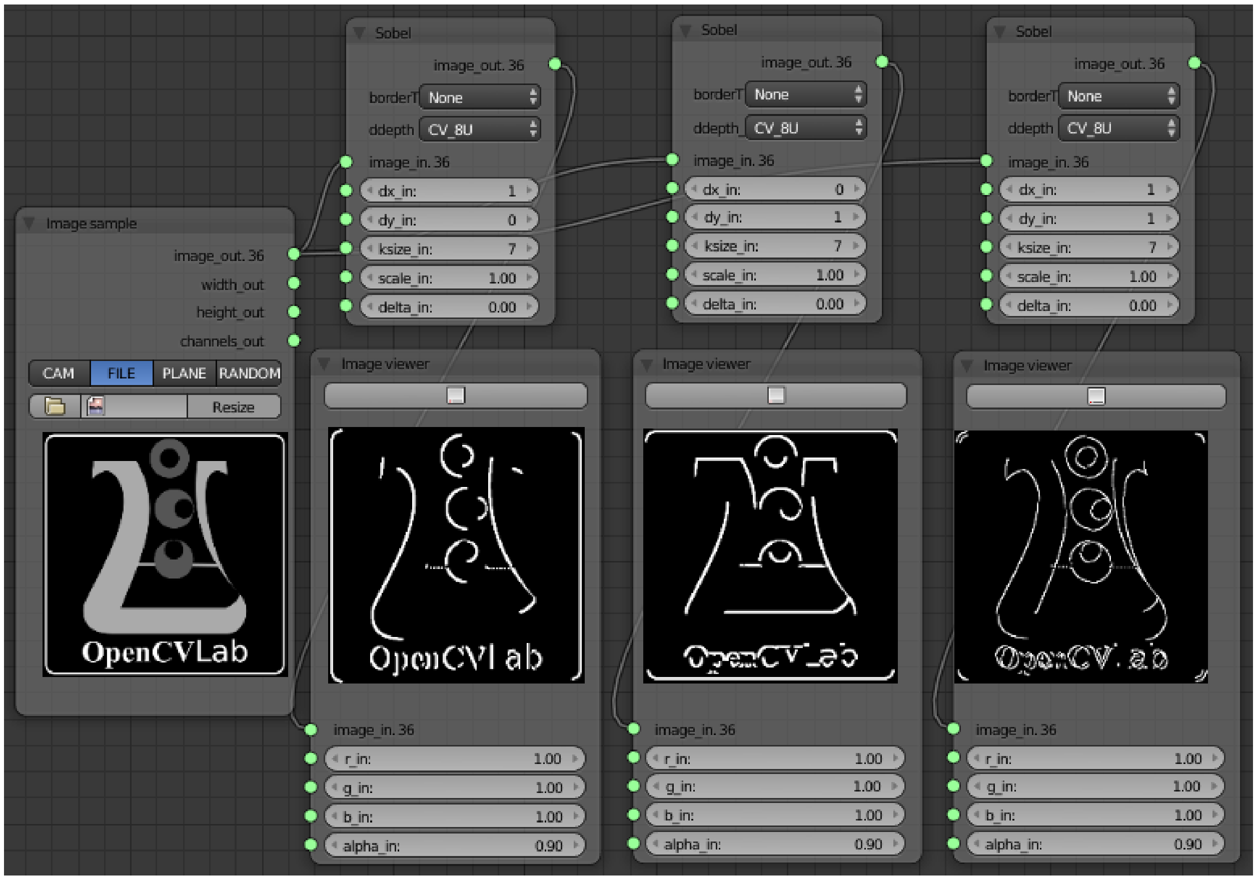


Image sample
image_out. 36
width_out
height_out
channels_out

Sobel
image_out. 36
borderT None
ddepth CV_8U
image_in. 36
dx_in: 1
dy_in: 0
ksize_in: 7
scale_in: 1.00
delta_in: 0.00

Sobel
image_out. 36
borderT None
ddepth CV_8U
image_in. 36
dx_in: 0
dy_in: 1
ksize_in: 7
scale_in: 1.00
delta_in: 0.00

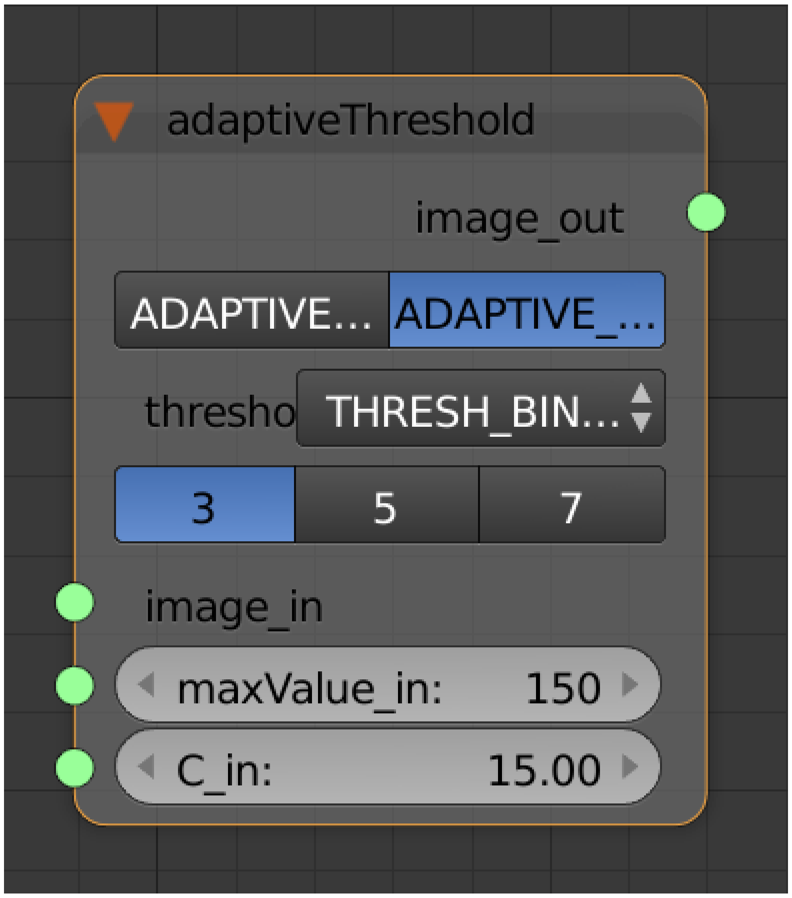
Sobel
image_out. 36
borderT None
ddepth CV_8U
image_in. 36
dx_in: 1
dy_in: 1
ksize_in: 7
scale_in: 1.00
delta_in: 0.00

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.7 adaptiveThreshold



Functionality

Applies an adaptive threshold to an array.

Inputs

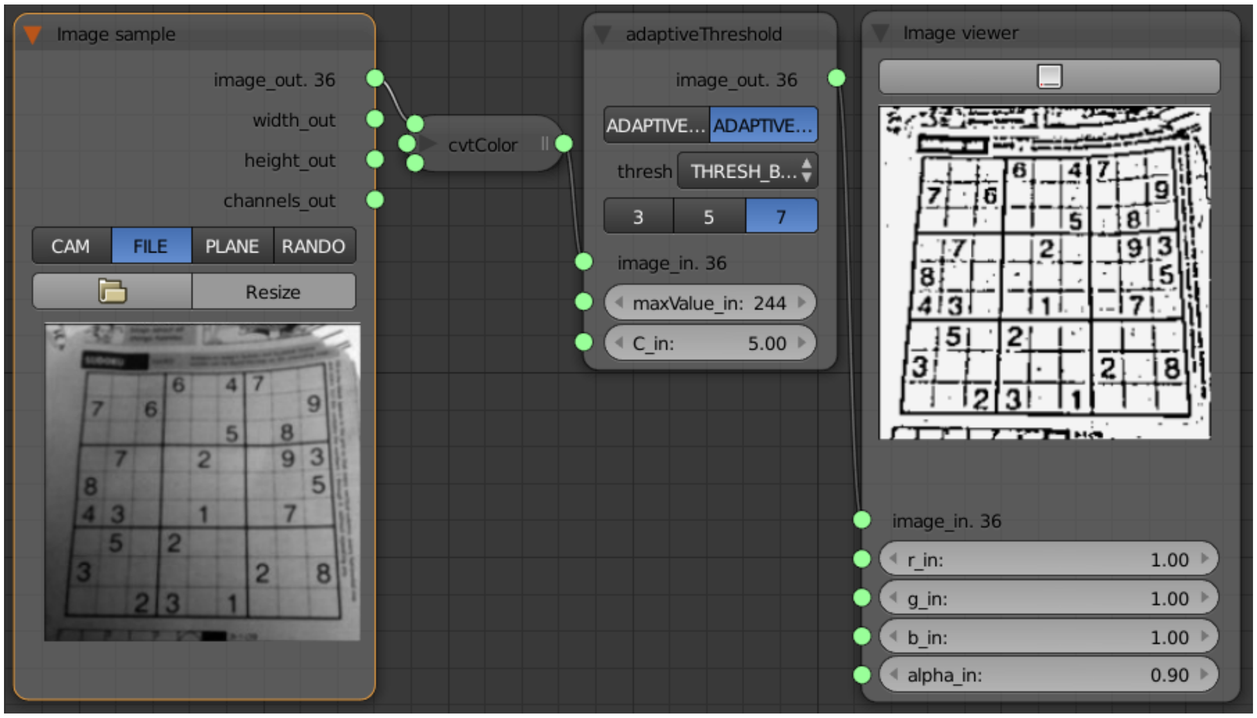
- C_in – Constant subtracted from the mean or weighted mean.
- adaptiveMethod_in – Adaptive thresholding algorithm to use, see cv::AdaptiveThresholdTypes .
- blockSize_in – Size of a pixel neighborhood that is used to calculate a threshold value for the pixel.
- image_in – Source 8-bit single-channel image.
- maxValue_in – Non-zero value assigned to the pixels for which the condition is satisfied.
- thresholdType_in – Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV, etc.

Outputs

- image_out – Destination image of the same size and the same type as src.

Locals

Examples



thresh THRESH_B...

maxValue_in: 244

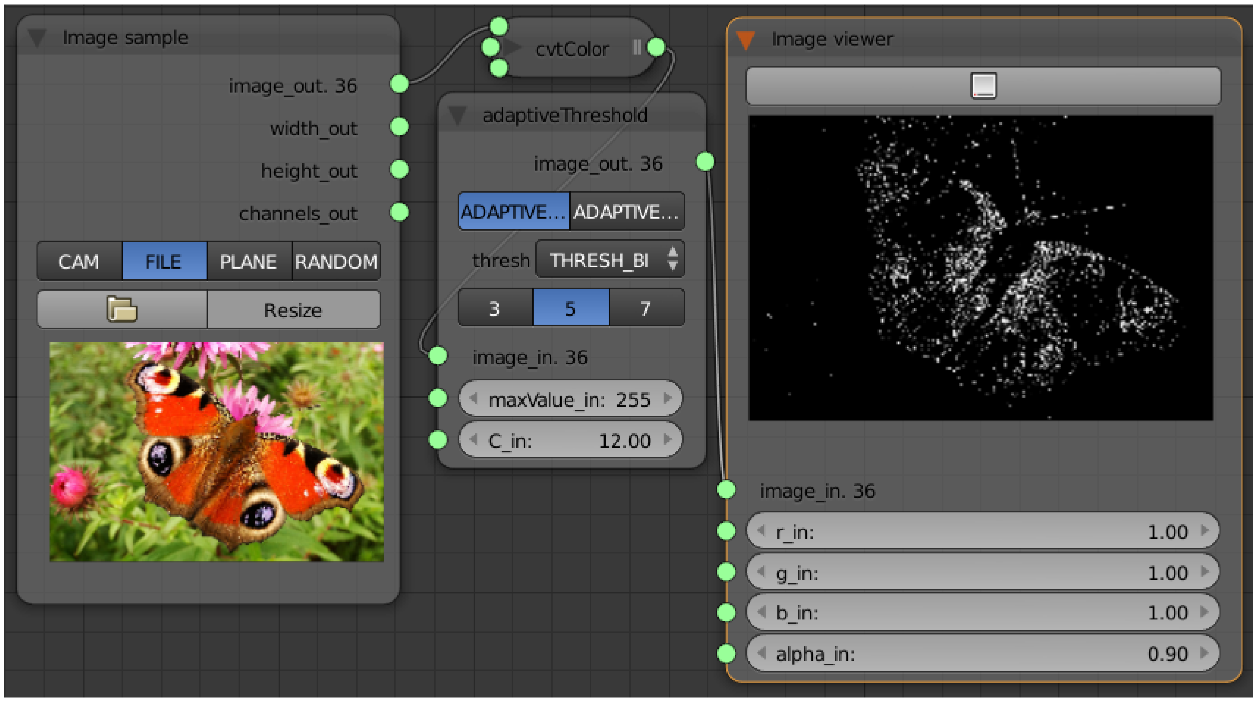
C_in: 5.00

r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90



thresh THRESH_BI

maxValue_in: 255

C_in: 12.00

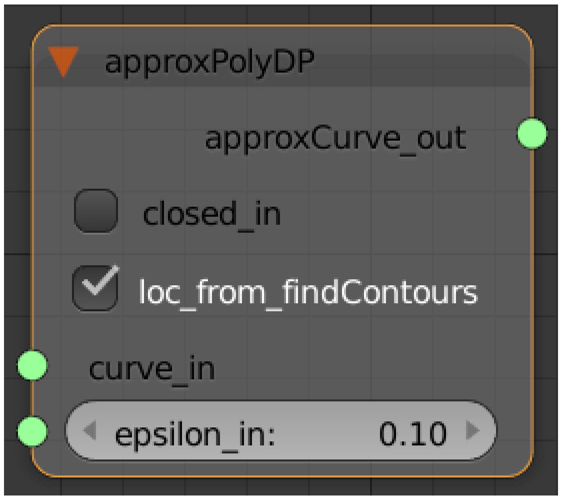
r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90

9.2.8 approxPolyDP



Functionality

Approximates a polygonal curve(s) with the specified precision.

Inputs

- closed_in – If true, the approximated curve is closed (its first and last vertices are connected). Otherwise, it is not closed.
- curve_in – Input vector of a 2D point stored in std::vector or Mat.
- epsilon_in – Parameter specifying the approximation accuracy. This is the maximum distance

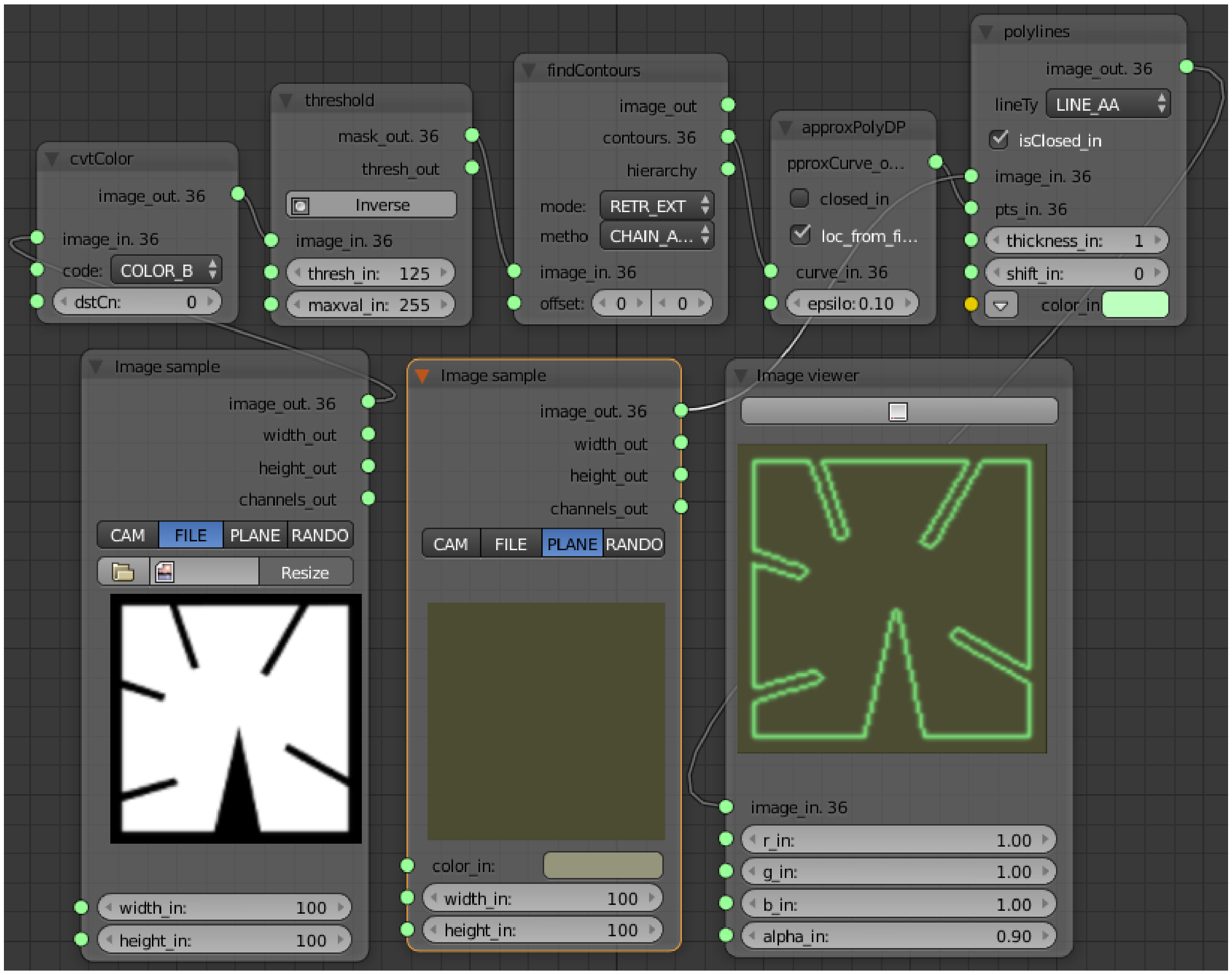
Outputs

- approxCurve_out – Result of the approximation. The type should match the type of the input curve.

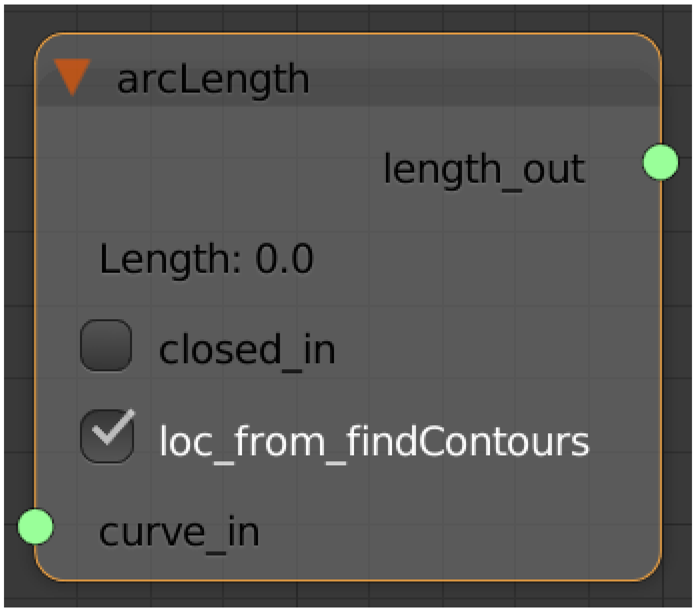
Locals

- loc_from_findContours – If linked with findContour node switch to True

Examples


cvtColor
image_out. 36
image_in. 36
code: COLOR_B
dstCn: 0
threshold
mask_out. 36
thresh_out
Inverse
image_in. 36
thresh_in: 125
maxval_in: 255
findContours
image_out
contours. 36
hierarchy
mode: RETR_EXT
metho CHAIN_A...
image_in. 36
offset: 0 0
approxPolyDP
pproxCurve_o...
closed_in
loc_from_fi...
curve_in. 36
epsilo: 0.10
polylines
image_out. 36
lineTy LINE_AA
isClosed_in
image_in. 36
pts_in. 36
thickness_in: 1
shift_in: 0
color_in
Image sample
image_out. 36
width_out
height_out
channels_out
CAM FILE PLANE RANDO
Resize
width_in: 100
height_in: 100
Image sample
image_out. 36
width_out
height_out
channels_out
CAM FILE PLANE RANDO
color_in:
width_in: 100
height_in: 100
Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.9 arcLength



Functionality

Calculates a contour perimeter or a curve length.

Inputs

- closed_in – Flag indicating whether the curve is closed or not.
- curve_in – Input vector of 2D points, stored in std::vector or Mat.

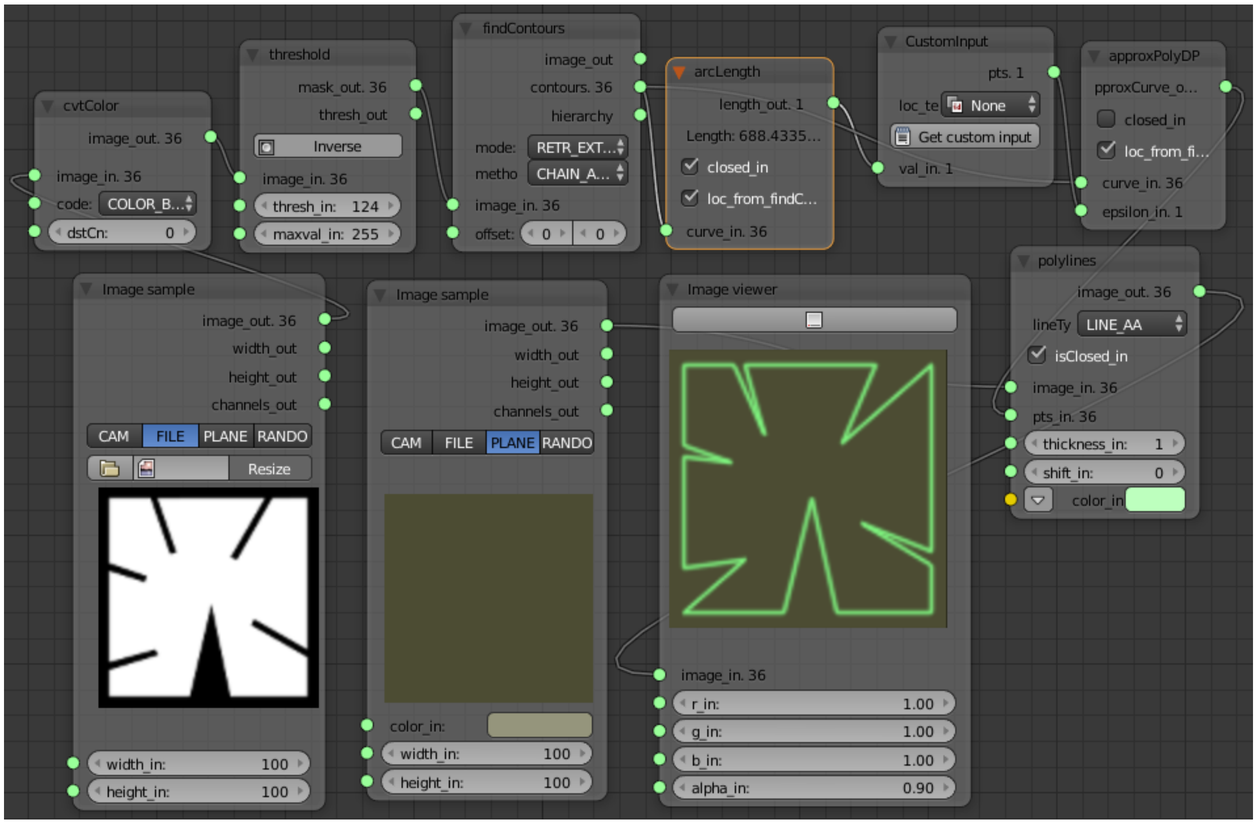
Outputs

- length_out – Length of contour.

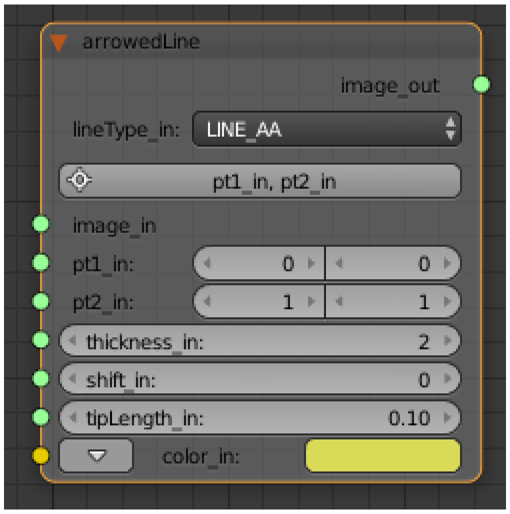
Locals

- loc_from_findContours – If linked with findContour node switch to True.

Examples



9.2.10 arrowedLine



Functionality

Draws a arrow segment pointing from the first point to the second one.

Inputs

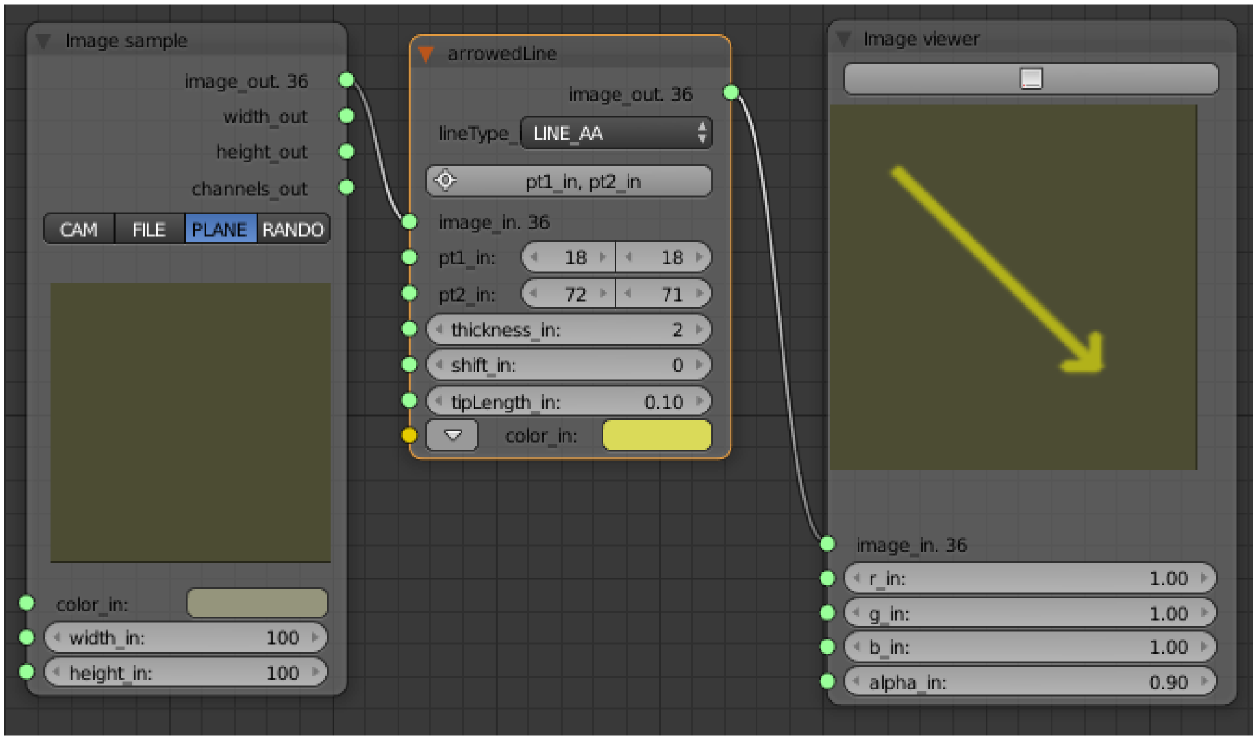
- color_in – Line color.
- image_in – Input image.
- lineType_in – Line type.
- pt1_in – First point of the line segment.
- pt2_in – Second point of the line segment.
- shift_in – Number of fractional bits in the point coordinates.
- thickness_in – Line thickness.
- tipLength_in – The length of the arrow tip in relation to the arrow length.

Outputs

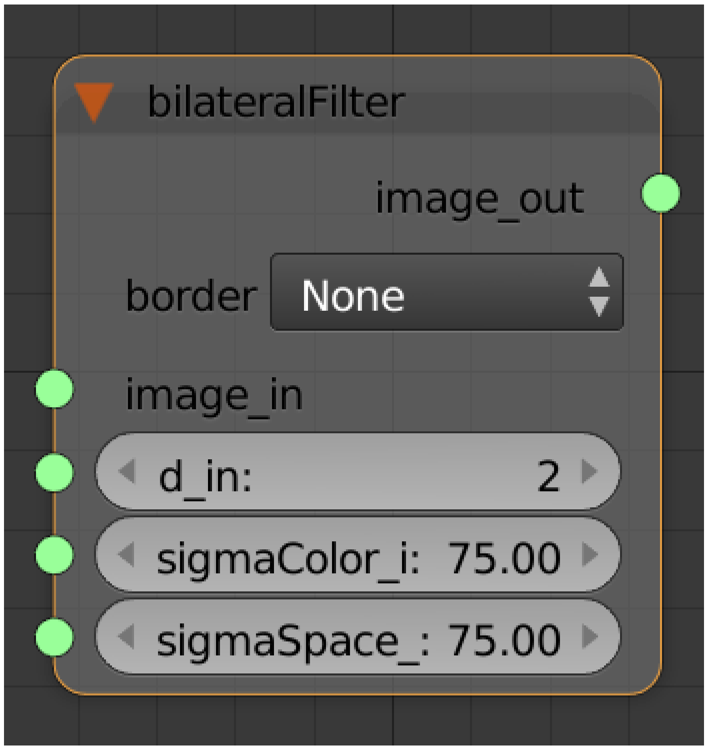
- image_out – Output image.

Locals

Examples



9.2.11 bilateralFilter



Functionality

Applies the bilateral filter to an image.

Inputs

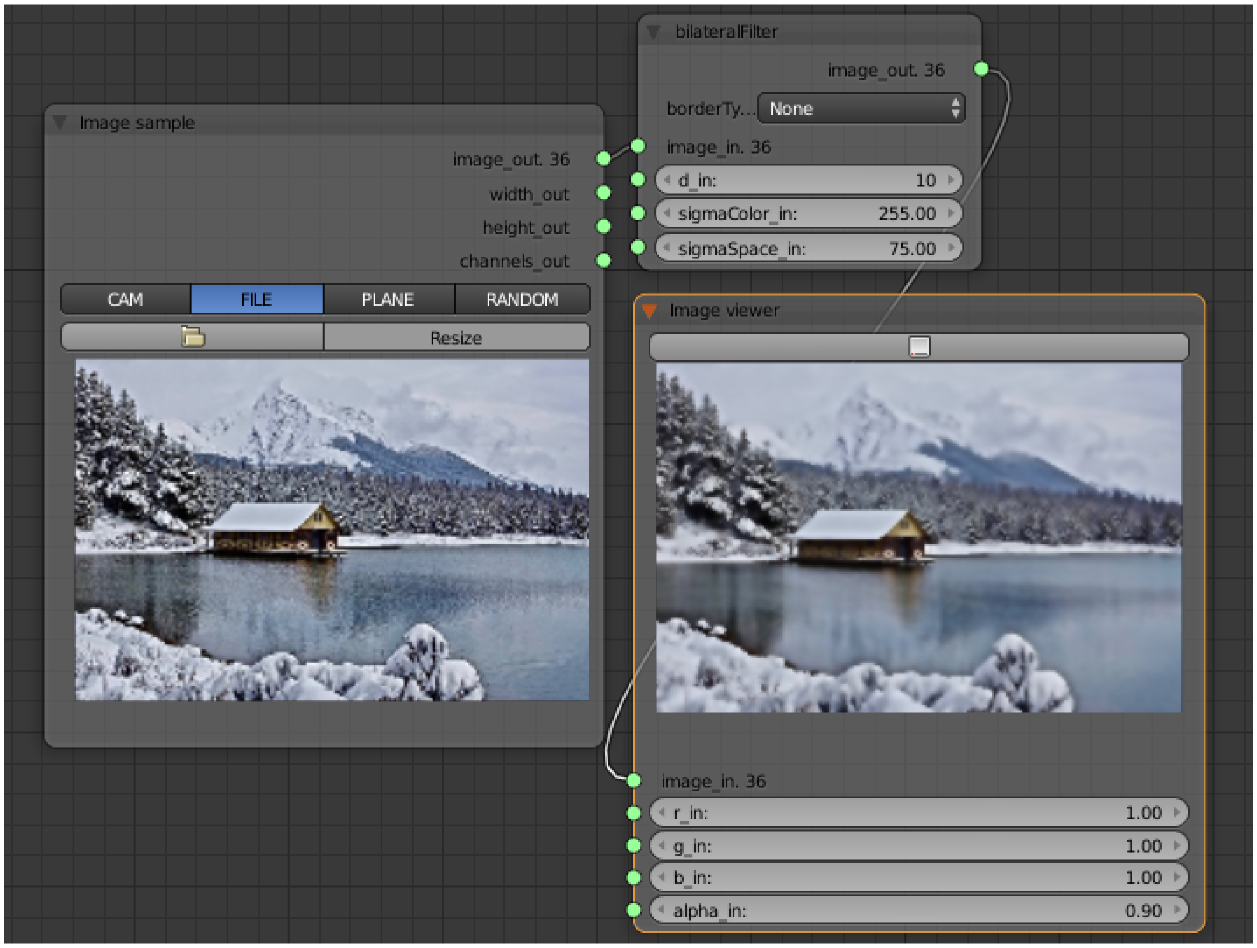
- borderType_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes.
- d_in – Diameter of each pixel neighborhood that is used during filtering. If it is non-positive, it is computed from sigmaSpace.
- sigmaColor_in – Filter sigma in the color space.
- sigmaSpace_in – Filter sigma in the coordinate space.

Outputs

- image_out – Output image.

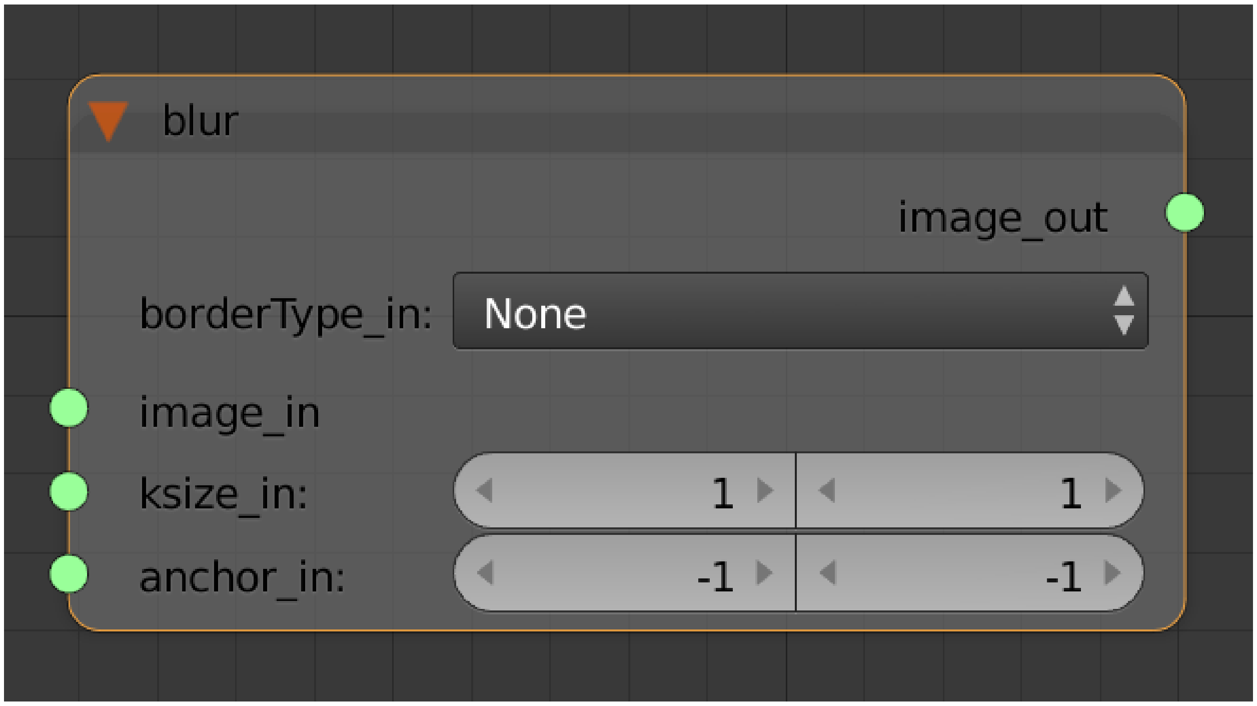
Locals

Examples



borderTy... None
d_in: 10
sigmaColor_in: 255.00
sigmaSpace_in: 75.00
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.12 blur



Functionality

Blurs an image using the normalized box filter.

Inputs

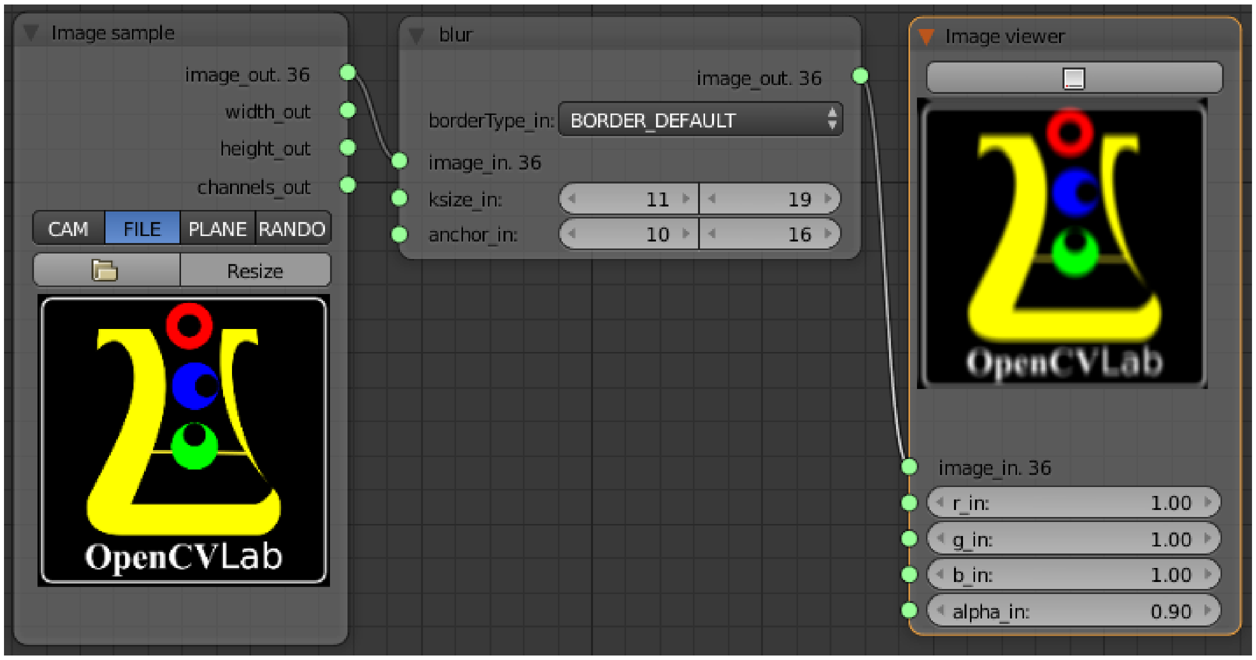
- anchor_in – Bnchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
- borderType_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes.
- image_in – Input image.
- ksize_in – Blurring kernel size.

Outputs

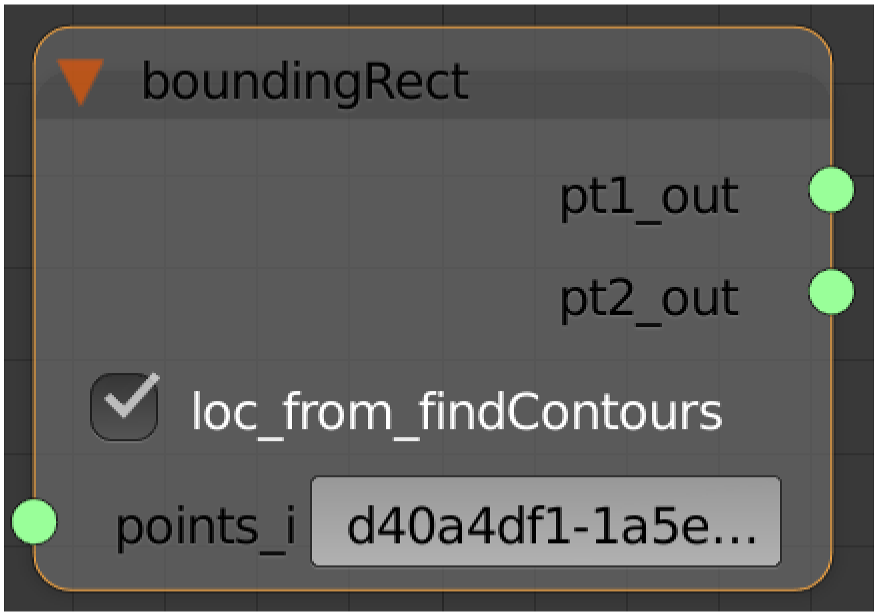
- image_out – Output image.

Locals

Examples



9.2.13 boundingRect



Functionality

Calculates the up-right bounding rectangle of a point set.

Inputs

- points_in – Input 2D point set, stored in std::vector or Mat.

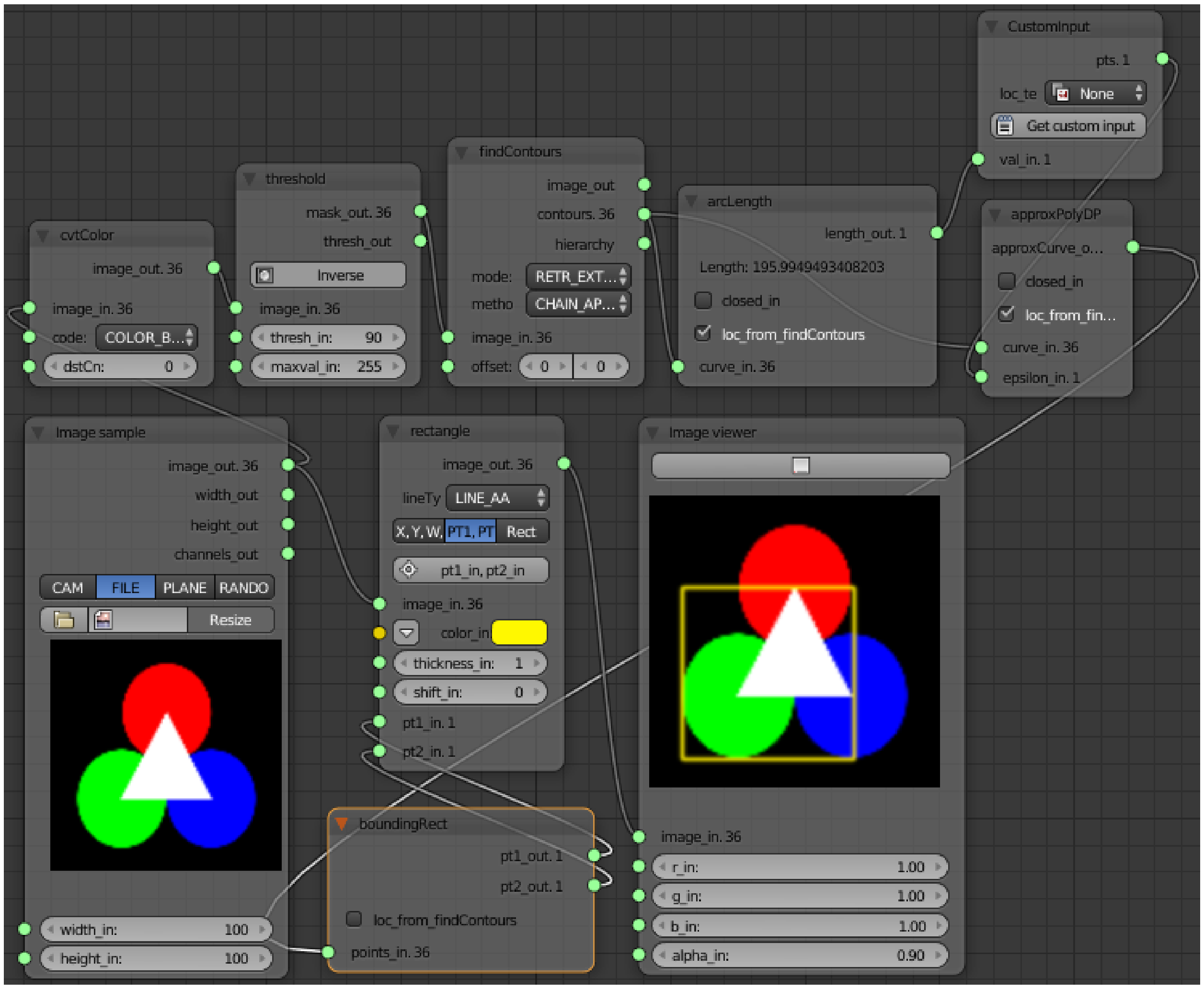
Outputs

- pt1_out – Pt1 output.
- pt2_out – Pt2 output.

Locals

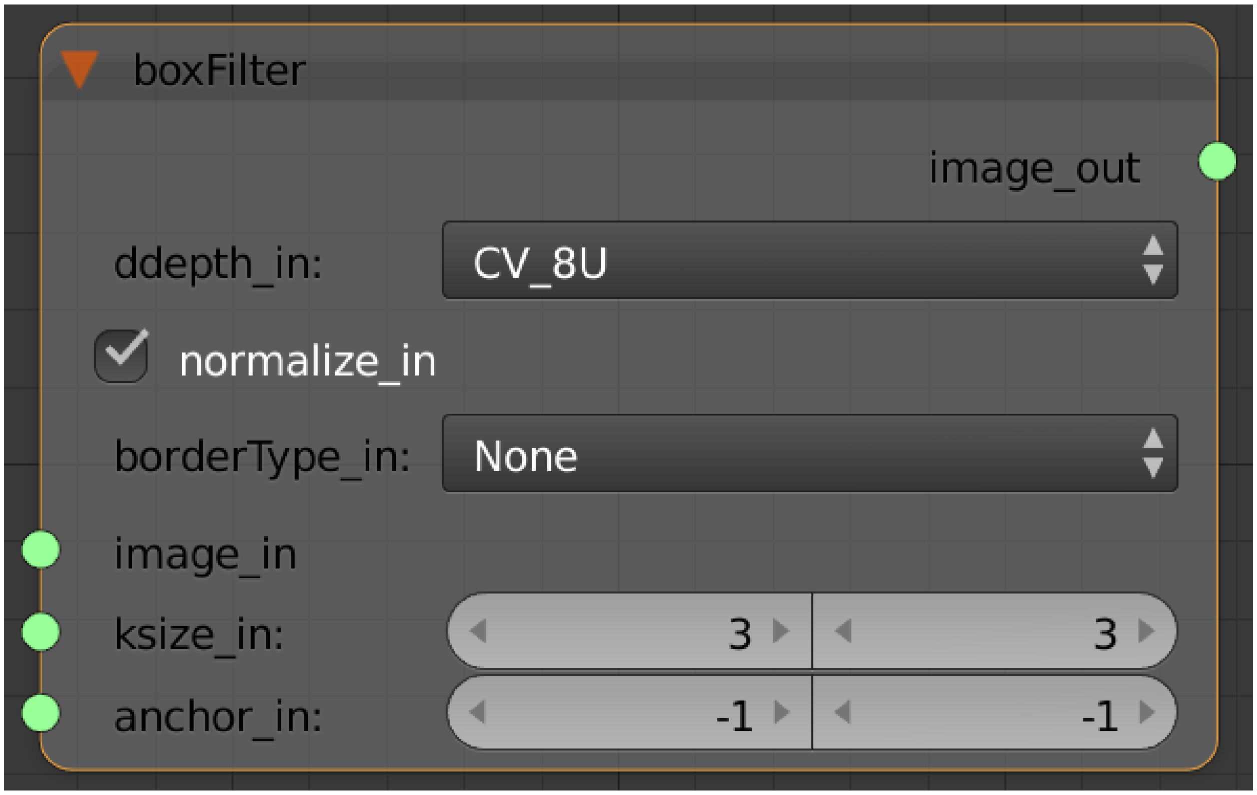
- loc_from_findContours – If linked with findContour node switch to True

Examples



code: COLOR_B...
dstCn: 0
thresh_in: 90
maxval_in: 255
mode: RETR_EXT...
metho CHAIN_AP...
offset: 0 0
Length: 195.9949493408203
lineTy LINE_AA
thickness_in: 1
shift_in: 0
width_in: 100
height_in: 100
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.14 boxFilter



Functionality

Blurs an image using the box filter.

Inputs

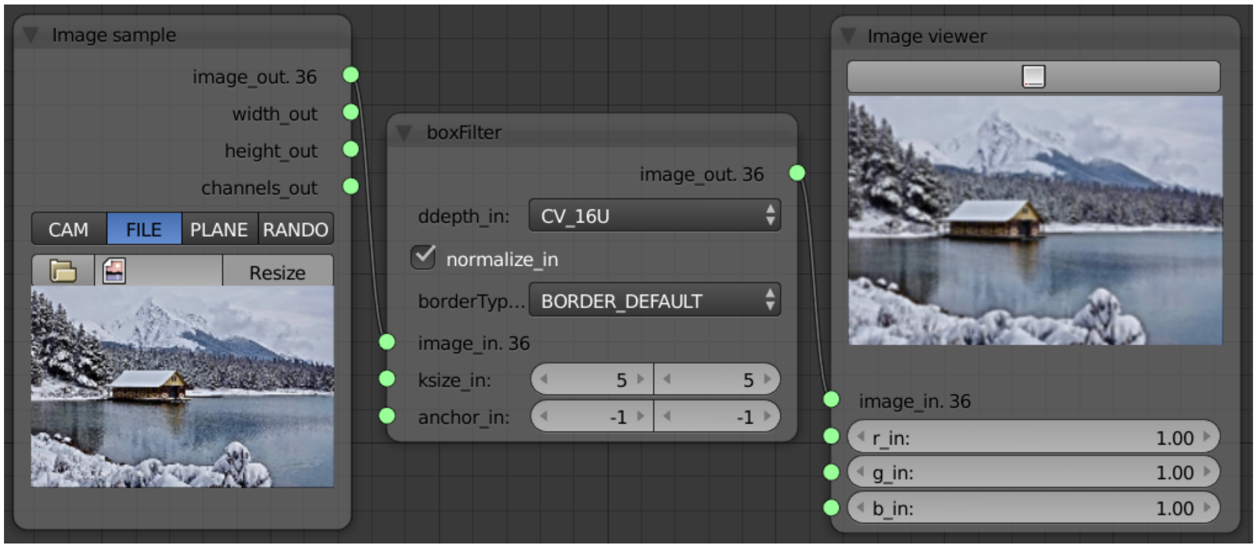
- anchor_in – Anchor point.
- borderType_in – Pixel extrapolation method, see cv::BorderTypes
- ddepth_in – The output image depth.
- image_in – Input image.
- ksize_in – Blurring kernel size.
- normalize_in – Flag, specifying whether the kernel is normalized by its area or not.

Outputs

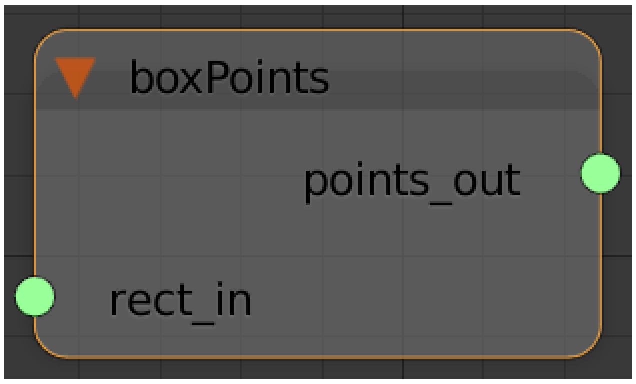
- image_out – Output image.

Locals

Examples



9.2.15 boxPoints



Functionality

Finds the four vertices of a rotated rect. Useful to draw the rotated rectangle.

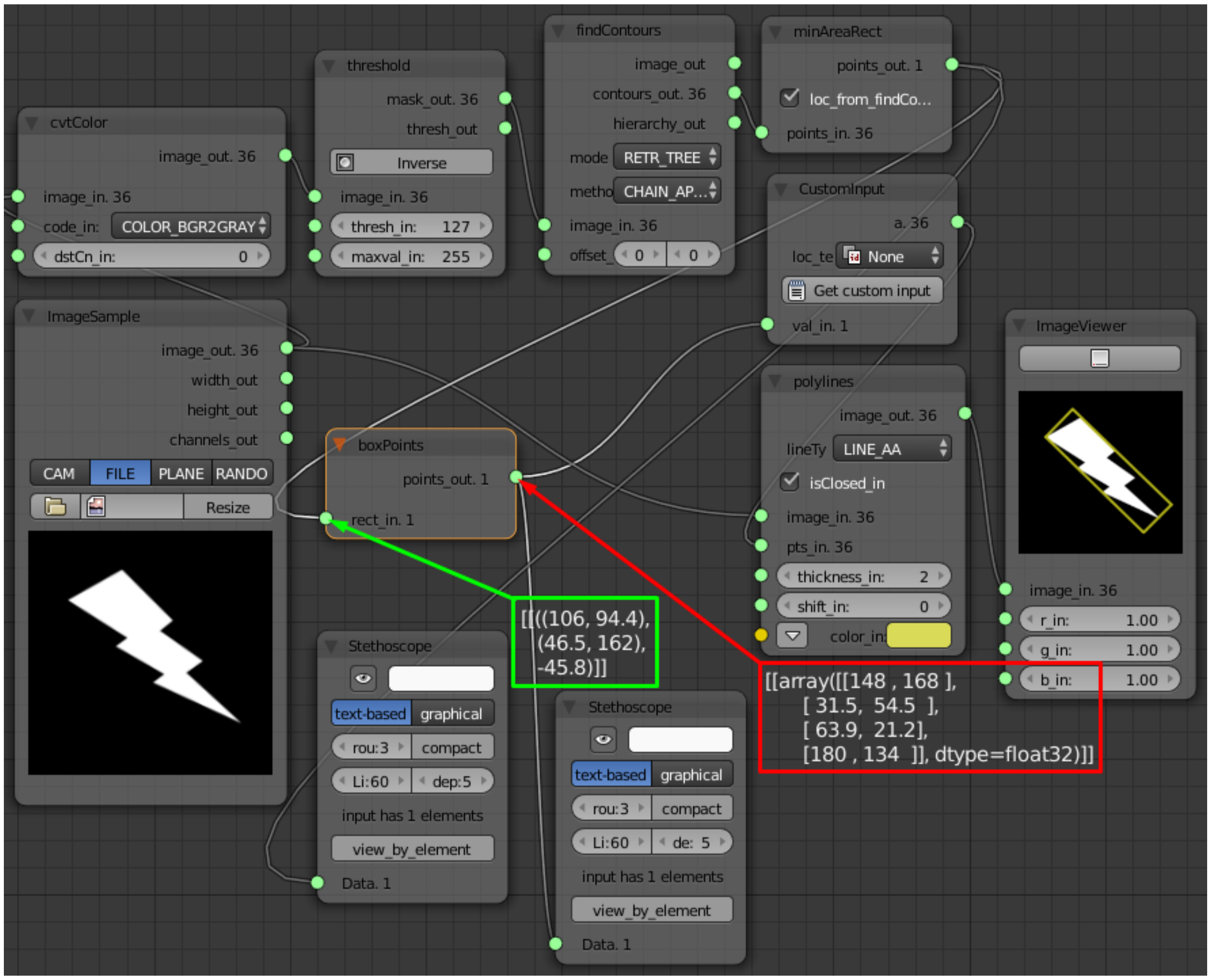
Inputs

- rect_in – Points and angle in one list.

Outputs

Locals

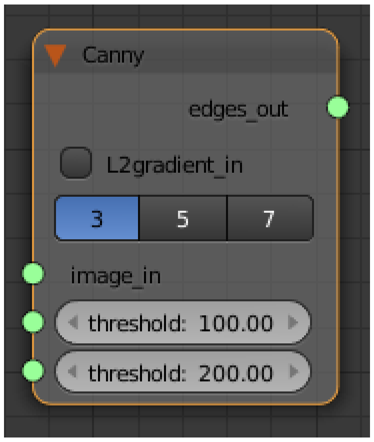
Examples



```
[[(106, 94.4),
 (46.5, 162),
-45.8)]]
```

```
[[array([[148 , 168 ],
       [ 31.5,  54.5  ],
       [ 63.9,  21.2],
       [180 , 134  ]], dtype=float32)]]
```

9.2.16 Canny



Functionality

Finds edges in an image using the [Canny86] algorithm.

Inputs

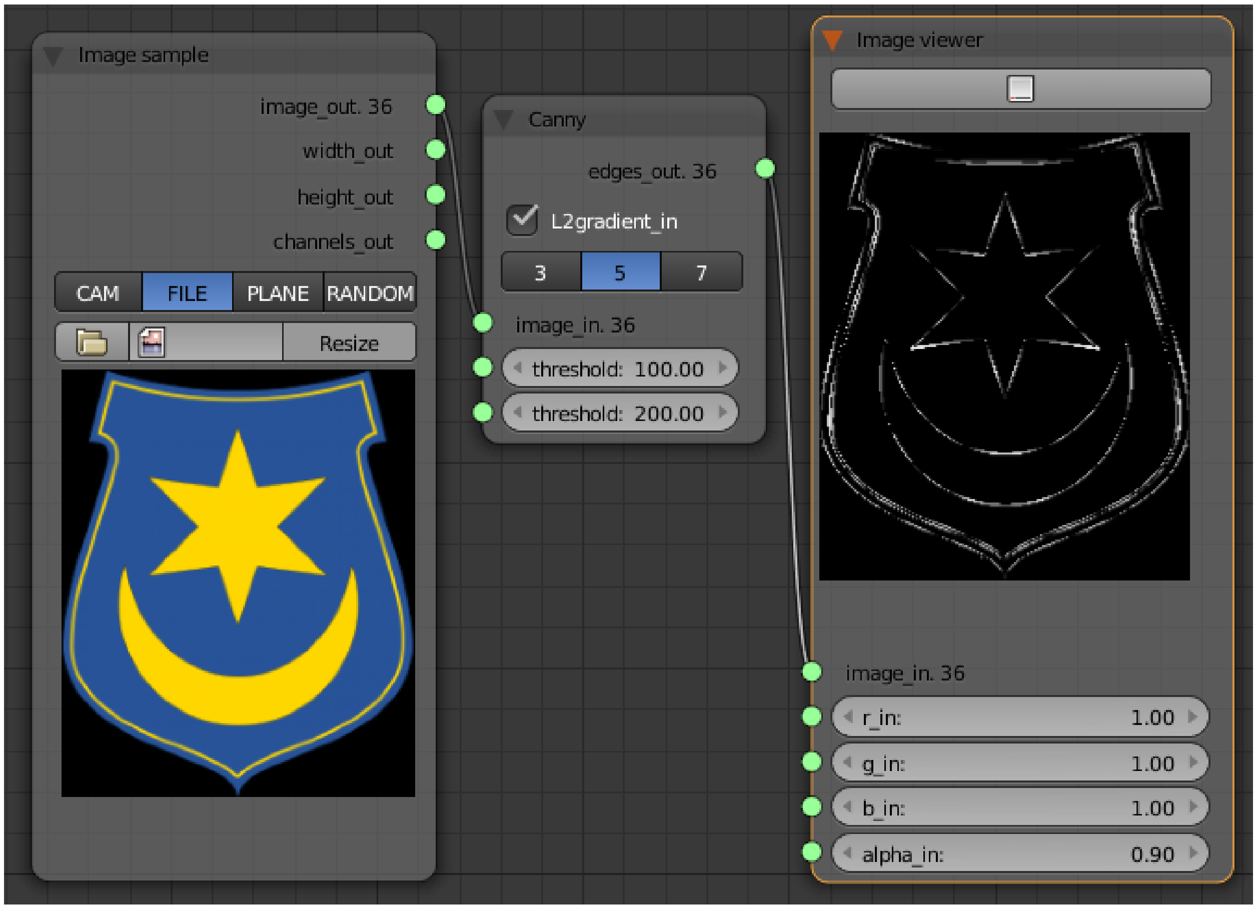
- L2gradient_in – Flag, indicating whether a more accurate.
- apertureSize_in – Aperture size for the Sobel operator.
- image_in – 8-bit input image.
- threshold1_in – First threshold for the hysteresis procedure.
- threshold2_in – Second threshold for the hysteresis procedure.

Outputs

- edges_out – Output edge map. Single channels 8-bit image, which has the same size as image.

Locals

Examples



✓ L2gradient_in

3 [✓ 5] 7

threshold: 100.00

threshold: 200.00

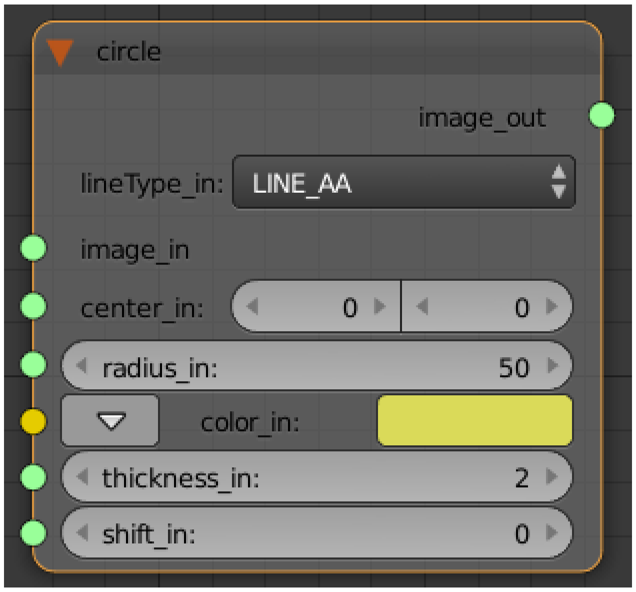
r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90

9.2.17 circle



Functionality

Draws a circle.

Inputs

- center_in – Center of the circle.
- color_in – Circle color.
- image_in – Input image.
- lineType_in – Type of the circle boundary. See the line description.
- radius_in – Radius of the circle.
- shift_in – Number of fractional bits in the coordinates of the center and in the radius value.
- thickness_in – Thickness of the circle outline, if positive. Negative thickness means that a filled circle is to be drawn.

Outputs

- image_out – Output image.

Locals

Examples

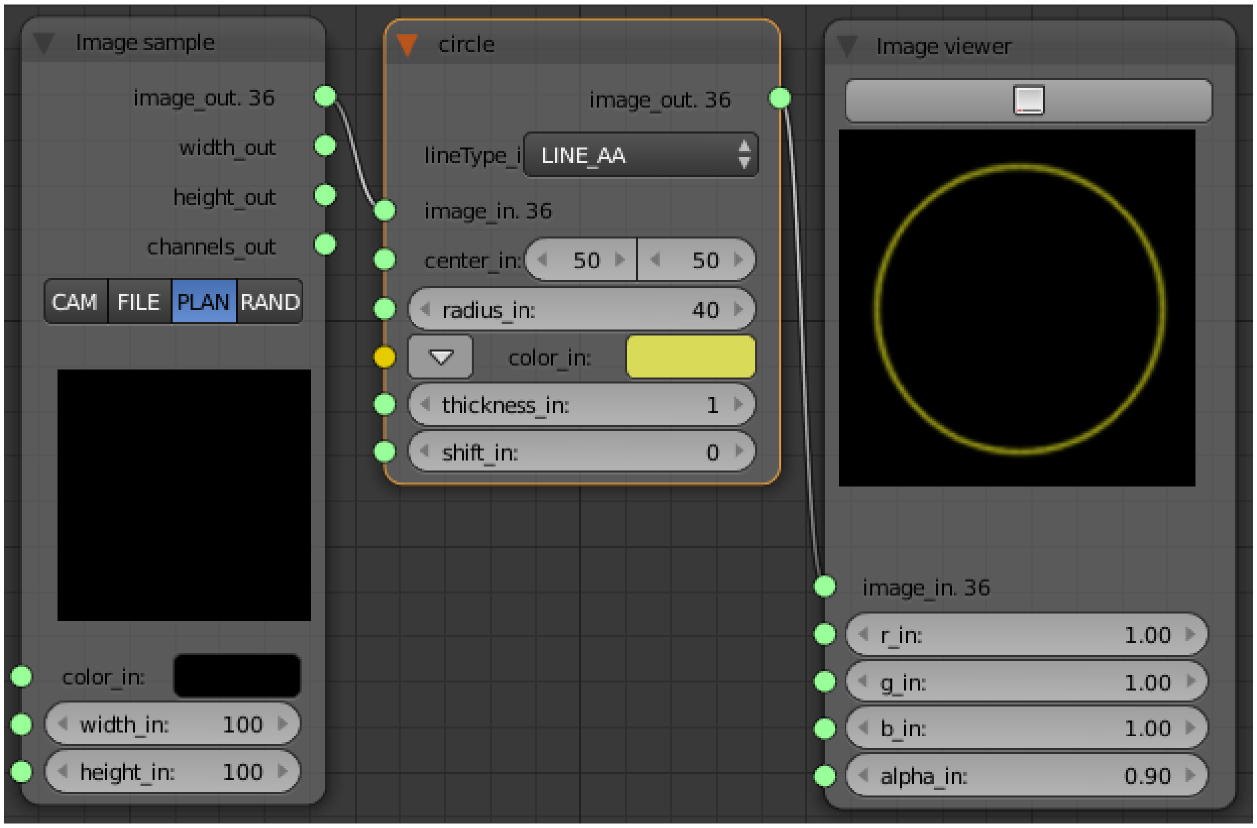
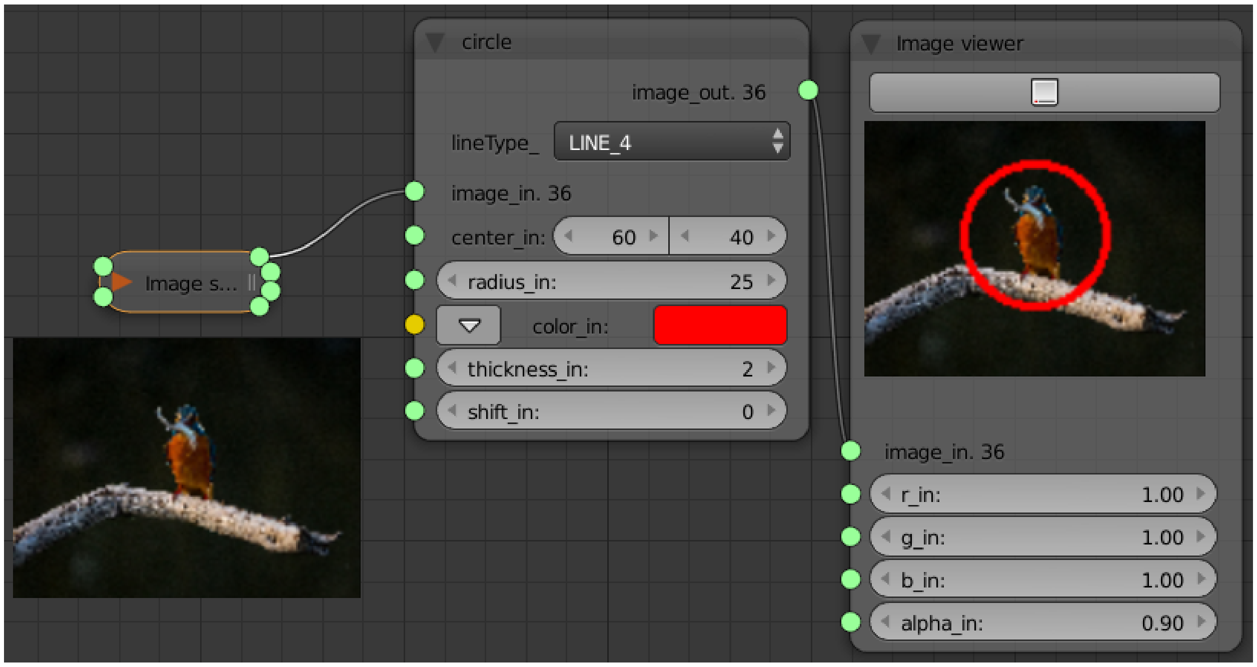


Image sample
image_out. 36
width_out
height_out
channels_out
CAM
FILE
PLAN
RAND
color_in:
width_in: 100
height_in: 100
circle
image_out. 36
lineType_i LINE_AA
image_in. 36
center_in: 50 50
radius_in: 40
color_in:
thickness_in: 1
shift_in: 0
Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90


circle
image_out. 36
lineType_ LINE_4
image_in. 36
center_in: 60 40
radius_in: 25
color_in:
thickness_in: 2
shift_in: 0
Image s...
Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

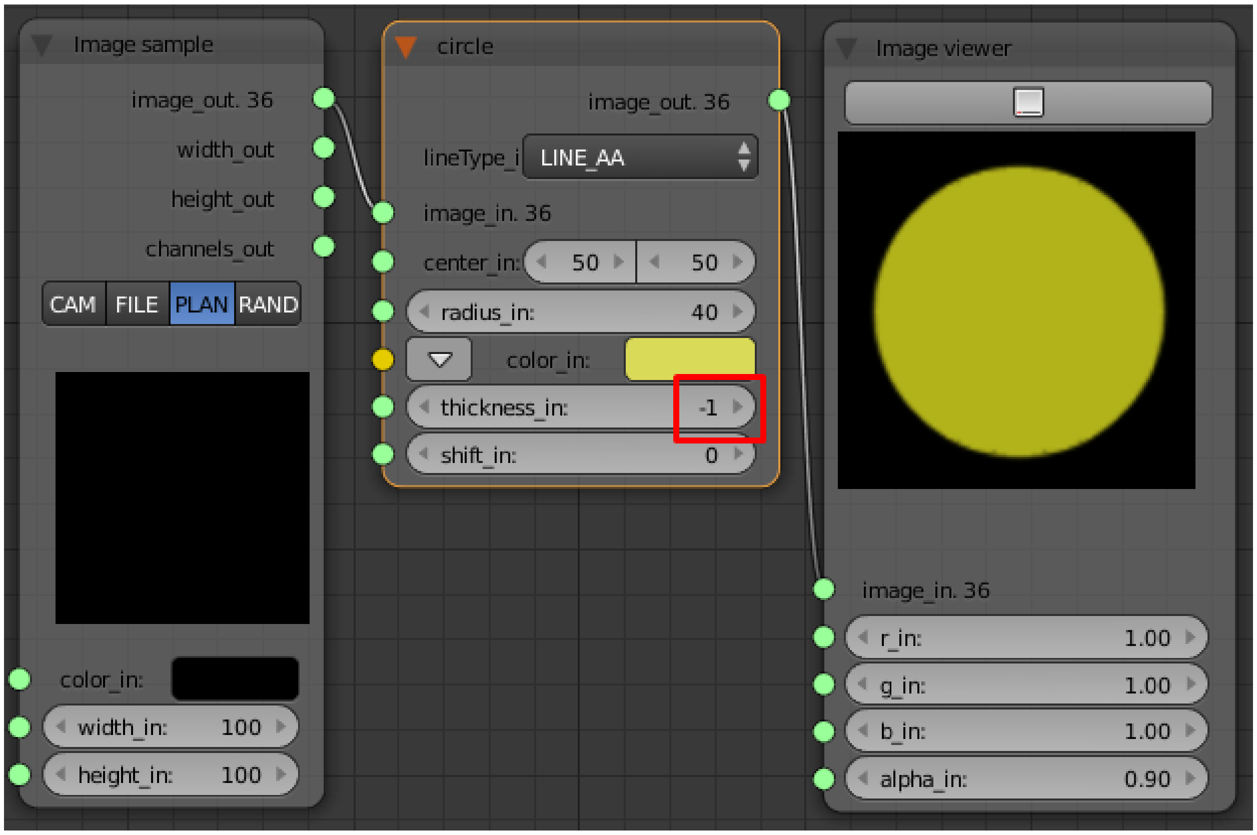


Image sample: color_in: 🟥 (black), width_in: 100, height_in: 100

circle: lineType_i LINE_AA, center_in: 50 50, radius_in: 40, color_in: 🟨, thickness_in: -1, shift_in: 0

Image viewer: r_in: 1.00, g_in: 1.00, b_in: 1.00, alpha_in: 0.90

9.2.18 clipLine

Functionality

Clips the line against the image rectangle.

Inputs

- imgRect_in – Image rectangle.
- pt1_in – First point of the line segment.
- pt2_in – Second point of the line segment.

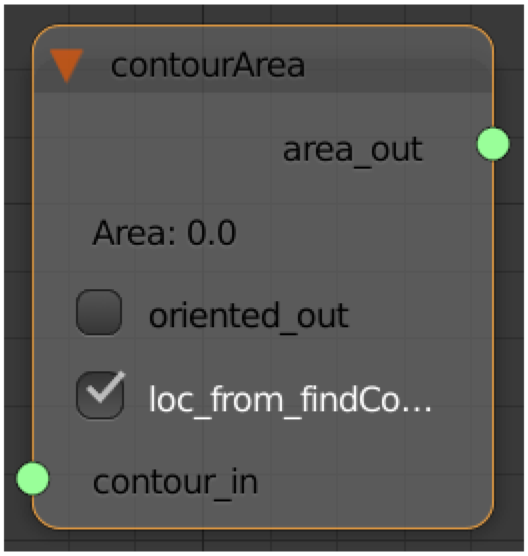
Outputs

- pt1_out – Pt1 output.
- pt2_out – Pt2 output.
- retval_out – Return value.

Locals

Examples

9.2.19 contourArea



Functionality

Calculates a contour area.

Inputs

- contour_in – Input vector of 2D points (contour vertices), stored in std::vector or Mat.

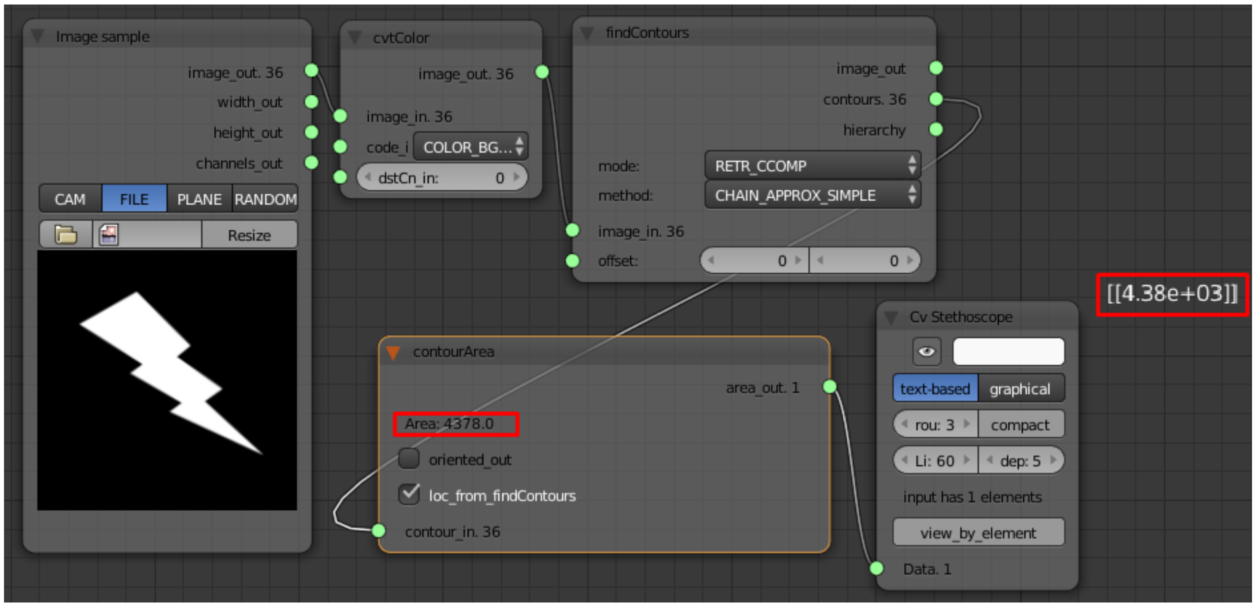
Outputs

- area_out – Area of contour.
- oriented_out – Oriented area flag. If it is true, the function returns a signed area value, depending on the contour orientation (clockwise or counter-clockwise).

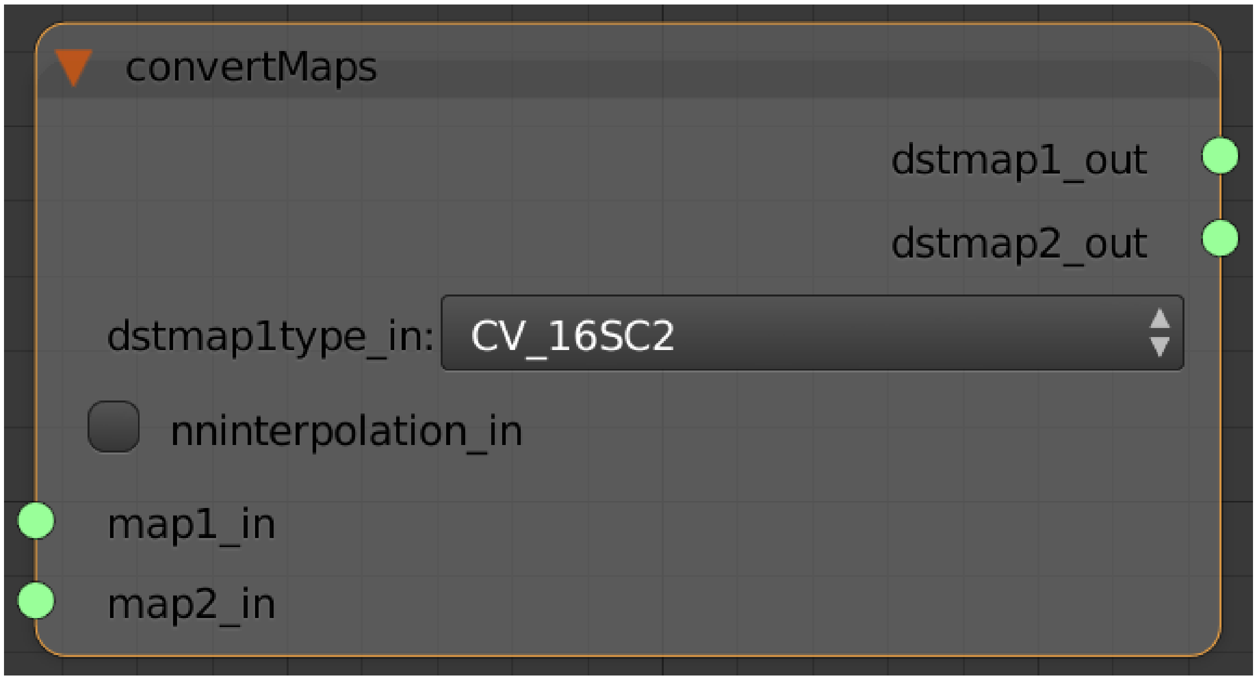
Locals

- loc_from_findContours – If linked with findContour node switch to True

Examples



9.2.20 convertMaps



Functionality

Converts image transformation maps from one representation to another.

Inputs

- dstmap1type_in – Type of the first output map that should be.
- map1_in – The first input map of type CV_16SC2 , CV_32FC1 , or CV_32FC2 .
- map2_in – The second input map of type CV_16UC1 , CV_32FC1 , or none (empty matrix), respectively.
- nninterpolation_in – Flag indicating whether the fixed-point maps are used for the nearest-neighbor or for a more complex interpolation.

Outputs

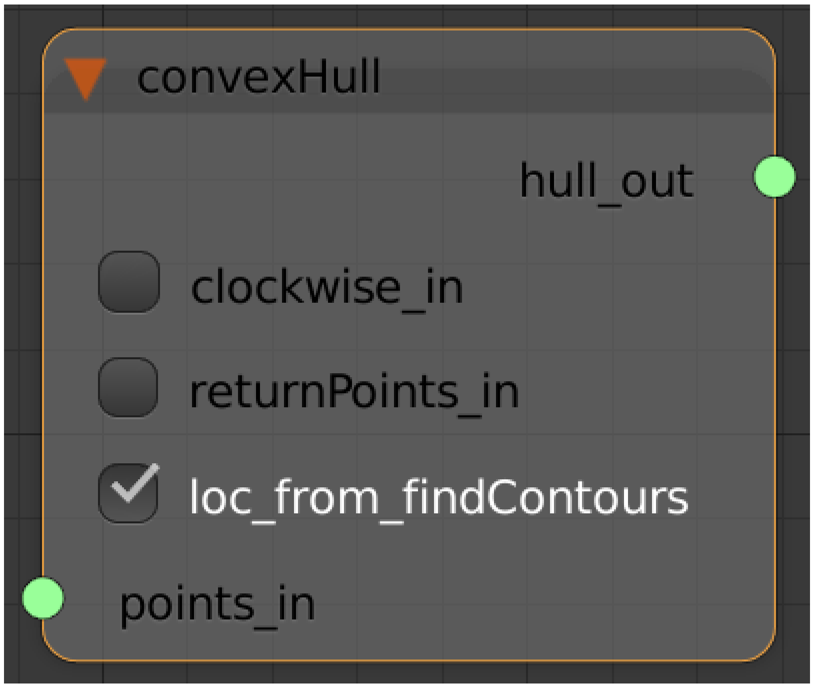
- dstmap1_out – The first output map that has the type dstmap1type and the same size as src .
- dstmap2_out – The second output map.

Locals

Examples

9.2.21 convertScaleAbs

9.2.22 convexHull



Functionality

Finds the convex hull of a point set.

Inputs

- clockwise_in – Orientation flag. If it is true, the output convex hull is oriented clockwise.
- points_in – Input 2D point set, stored in std::vector or Mat.
- returnPoints_in – Operation flag. In case of a matrix, when the flag is true, the function returns convex hull points.

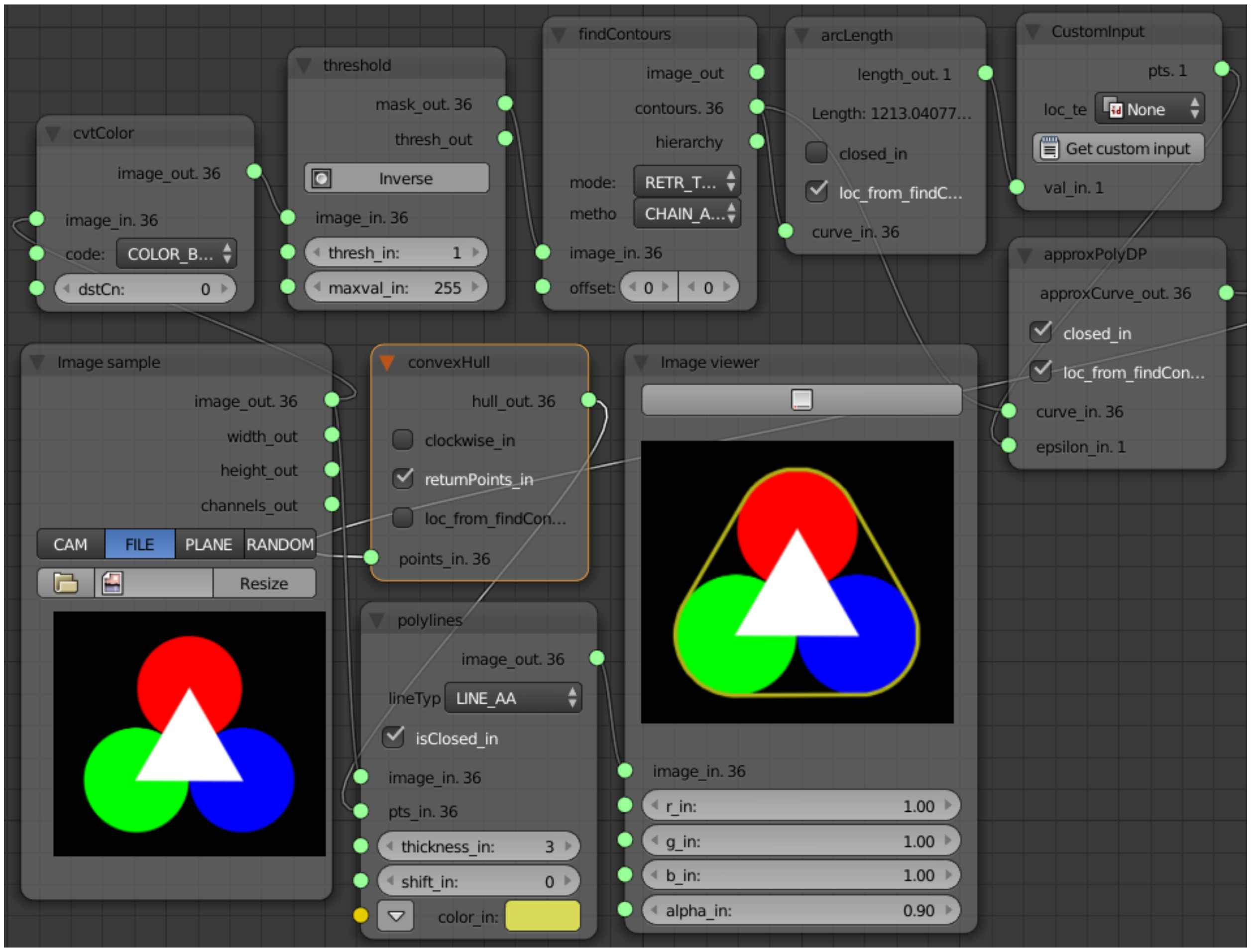
Outputs

- hull_out – Output convex hull. It is either an integer vector of indices or vector of points.

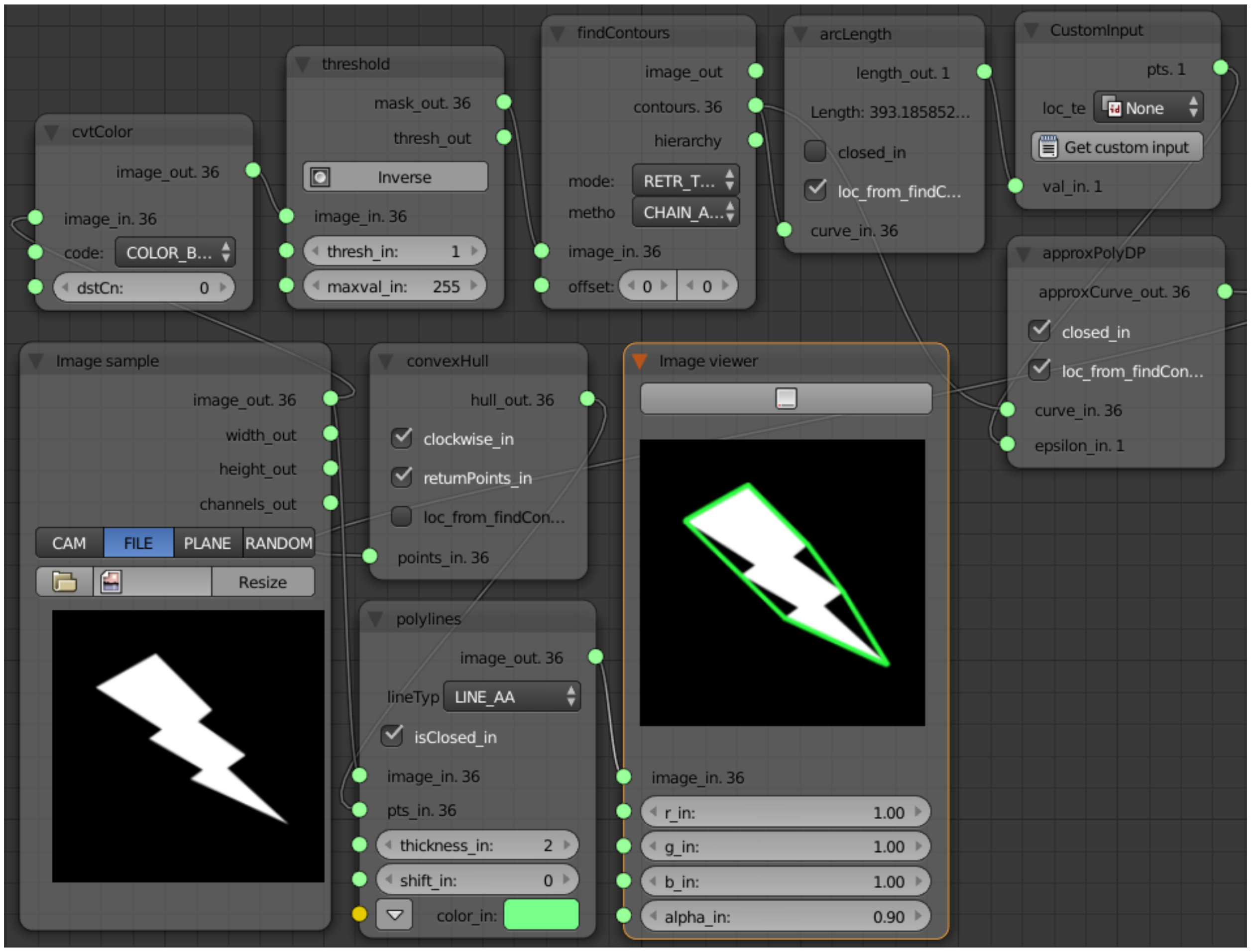
Locals

- loc_from_findContours – If linked with findContour node switch to True

Examples



code: COLOR_B...
dstCn: 0
thresh_in: 1
maxval_in: 255
mode: RETR_T...
metho CHAIN_A...
offset: 0 0
Length: 1213.04077...
closed_in ☐
loc_from_findC... ✓
loc_te None
closed_in ✓
loc_from_findCon... ✓
clockwise_in ☐
returnPoints_in ✓
loc_from_findCon... ☐
lineTyp LINE_AA
isClosed_in ✓
thickness_in: 3
shift_in: 0
color_in:
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90



cvtColor
image_out. 36
image_in. 36
code: COLOR_B...
dstCn: 0

threshold
mask_out. 36
thresh_out
Inverse
image_in. 36
thresh_in: 1
maxval_in: 255

findContours
image_out
contours. 36
hierarchy
mode: RETR_T...
metho CHAIN_A...
image_in. 36
offset: 0 0

arcLength
length_out. 1
Length: 393.185852...
closed_in
loc_from_findC...
curve_in. 36

CustomInput
pts. 1
loc_te None
Get custom input
val_in. 1

approxPolyDP
approxCurve_out. 36
closed_in
loc_from_findCon...
curve_in. 36
epsilon_in. 1

Image sample
image_out. 36
width_out
height_out
channels_out
CAM FILE PLANE RANDOM
Resize

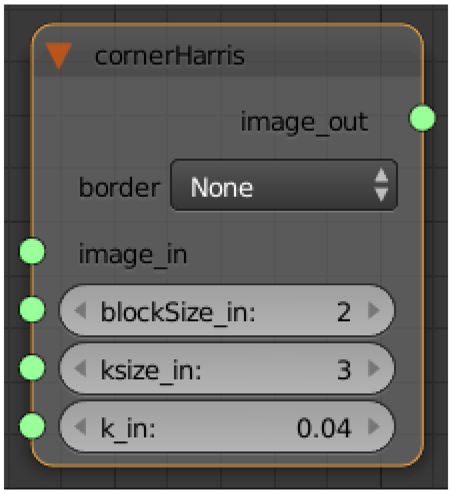
convexHull
hull_out. 36
clockwise_in
returnPoints_in
loc_from_findCon...
points_in. 36

polylines
image_out. 36
lineTyp LINE_AA
isClosed_in
image_in. 36
pts_in. 36
thickness_in: 2
shift_in: 0
color_in:

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.23 copyMakeBorder

9.2.24 cornerHarris



cornerHarris
image_out
border None
image_in
blockSize_in: 2
ksize_in: 3
k_in: 0.04

Functionality

Harris corner detector.

Inputs

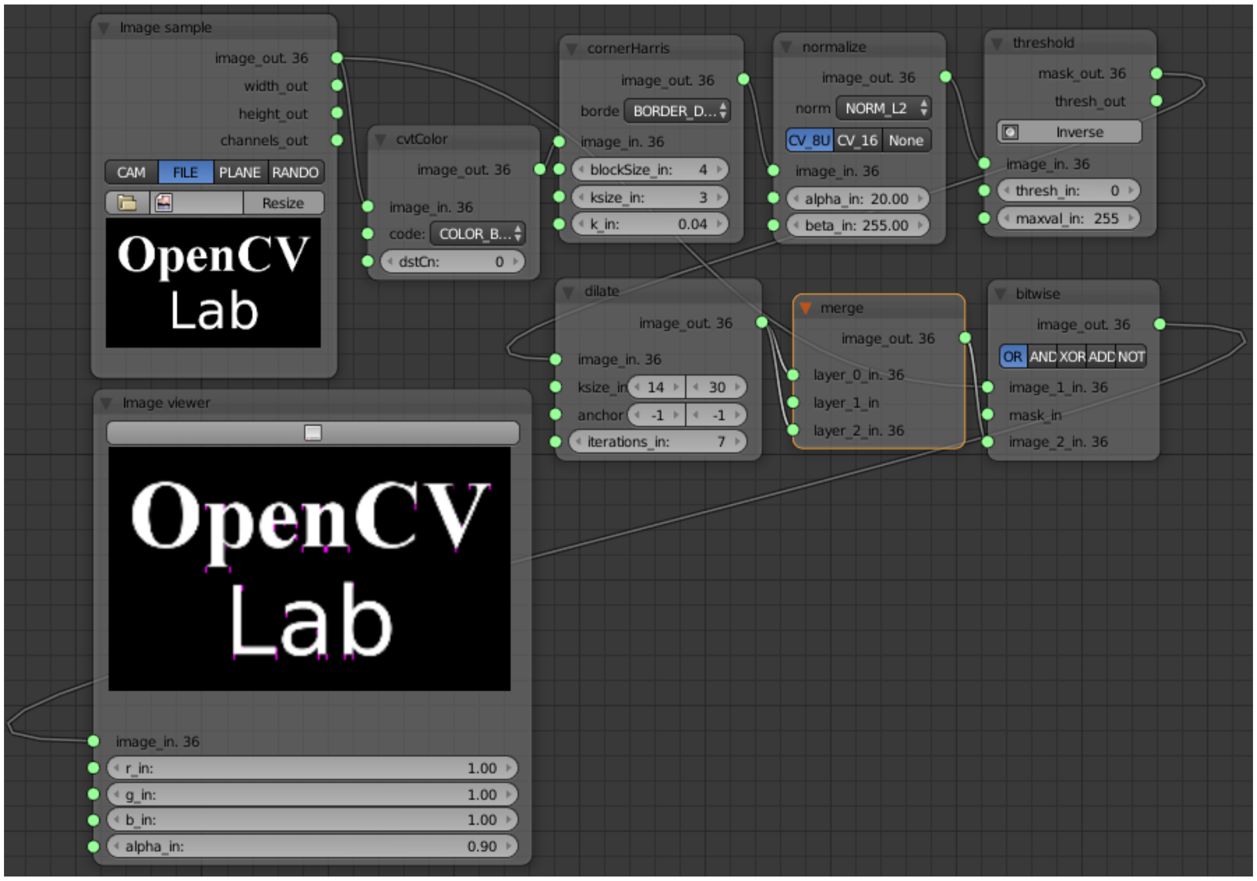
- blockSize_in – Neighborhood size (see the details on cornerEigenValsAndVecs).
- borderType_in – Pixel extrapolation method. See cv::BorderTypes.
- image_in – Input single-channel 8-bit or floating-point image.
- k_in – Harris detector free parameter. See the formula below.
- ksize_in – Aperture parameter for the Sobel operator.

Outputs

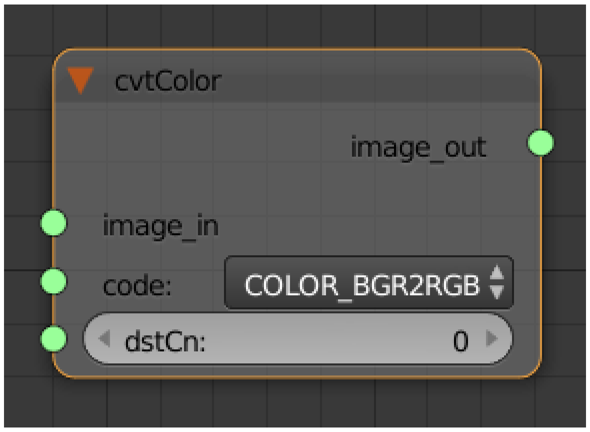
- image_out – Image to store the Harris detector responses.

Locals

Examples



9.2.25 cvtColor



Functionality

Converts an image from one color space to another.

Inputs

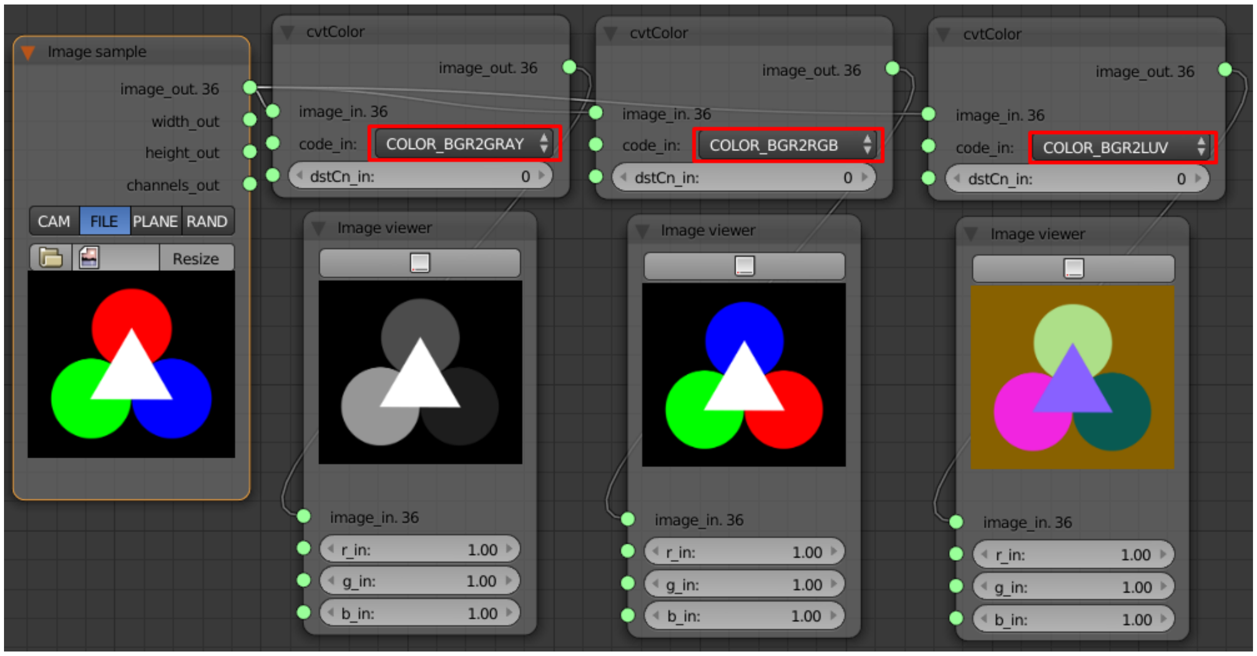
- code_in – Color space conversion code (see cv::ColorConversionCodes).
- dstCn_in – Number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from input image and code.
- image_in – Input image: 8-bit unsigned, 16-bit unsigned (CV_16UC…), or single-precision floating-point.

Outputs

- image_out – Output image of the same size and depth as input image.

Locals

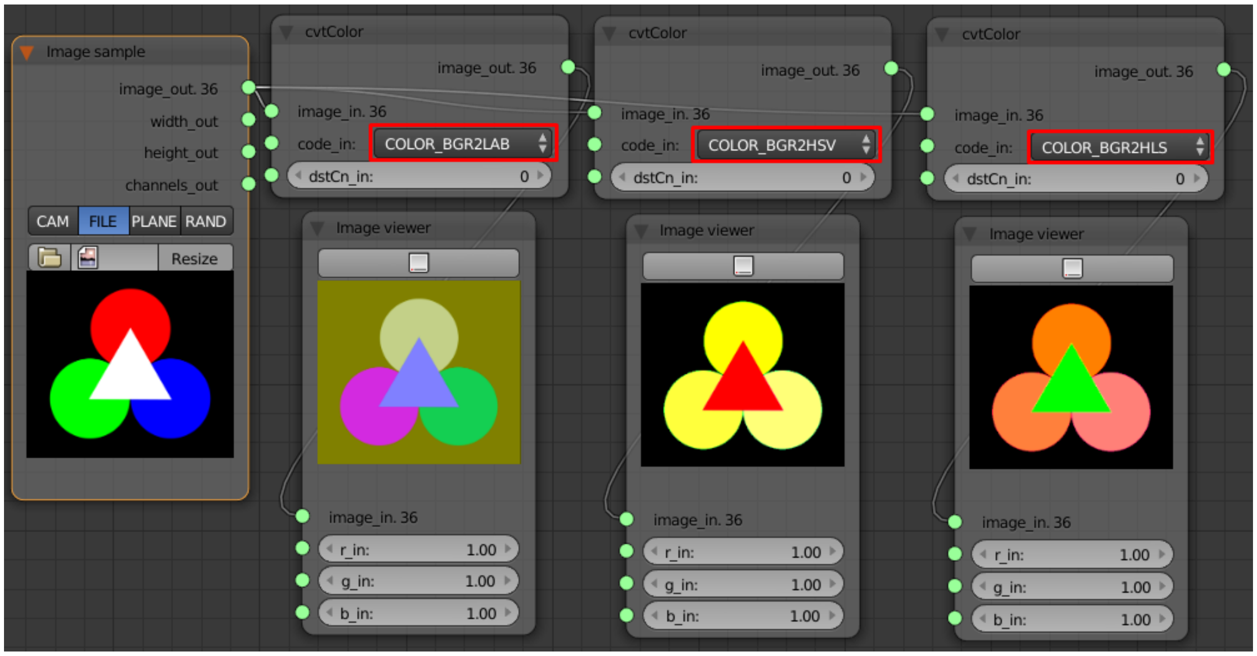
Examples



code_in: COLOR_BGR2GRAY
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00

code_in: COLOR_BGR2RGB
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00

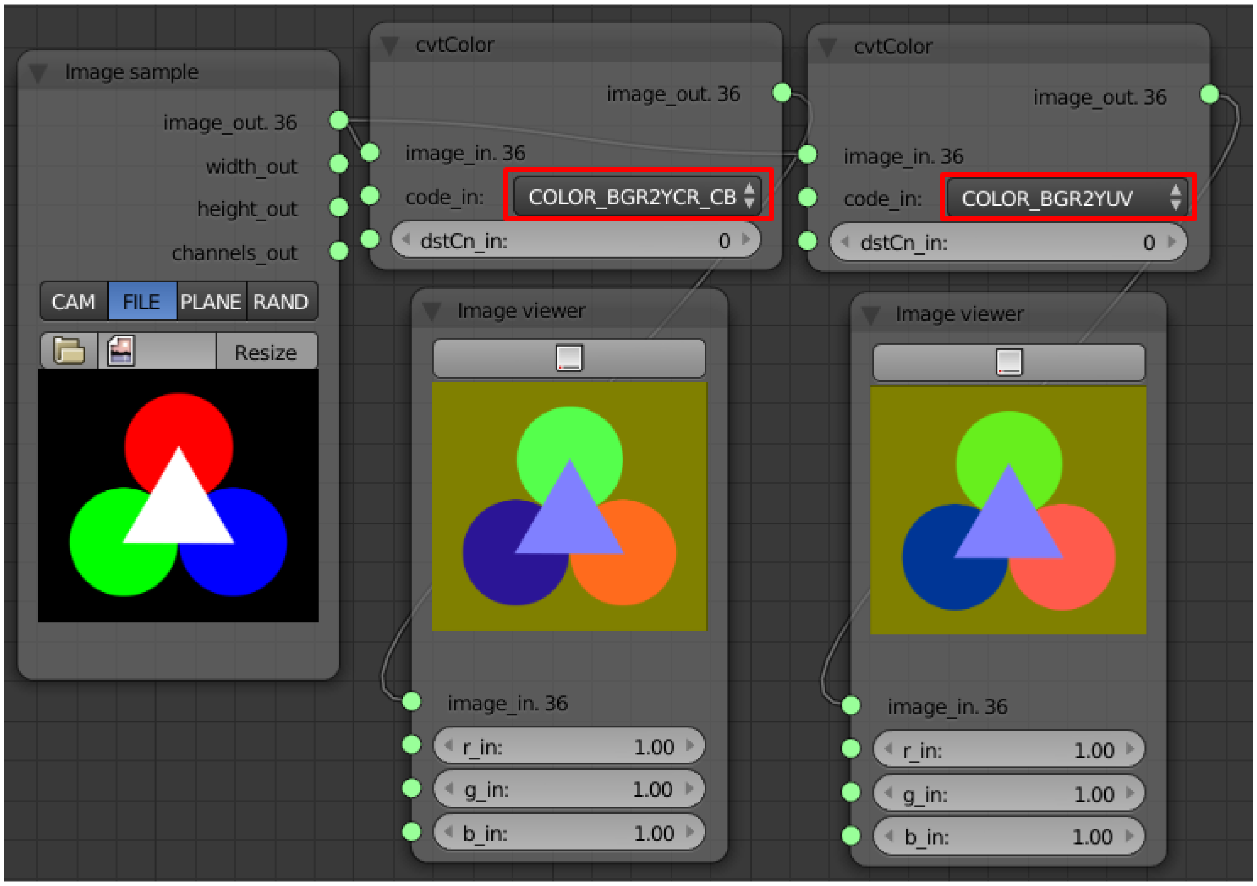
code_in: COLOR_BGR2LUV
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00



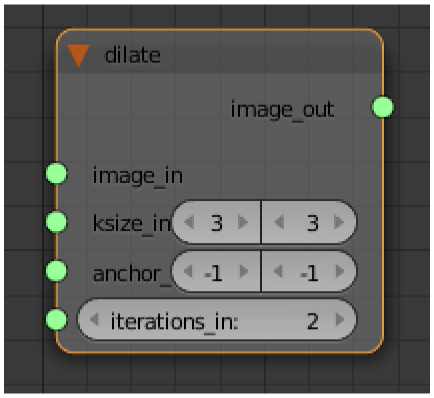
code_in: COLOR_BGR2LAB
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00

code_in: COLOR_BGR2HSV
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00

code_in: COLOR_BGR2HLS
dstCn_in: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00



9.2.26 dilate



Functionality

Dilates an image by using a specific structuring element.

Inputs

- anchor_in – Position of the anchor within the element.

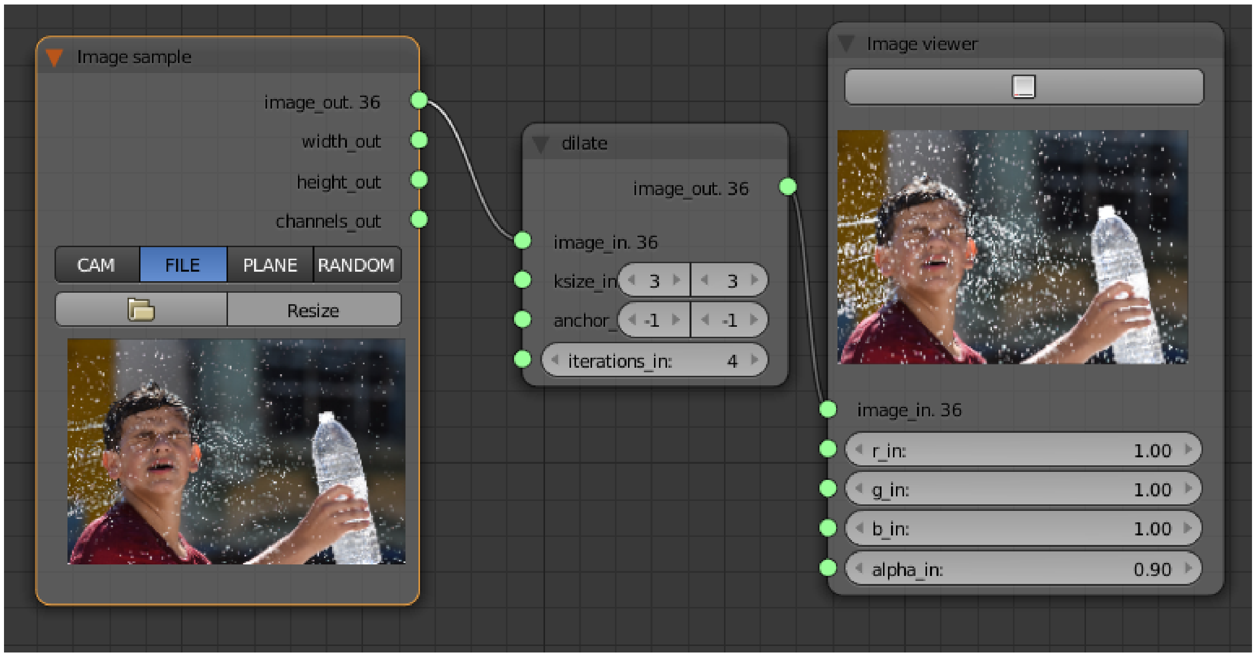
- image_in – Input image.
- iterations_in – Number of times erosion is applied.
- ksize_in – Structuring element used for erosion.

Outputs

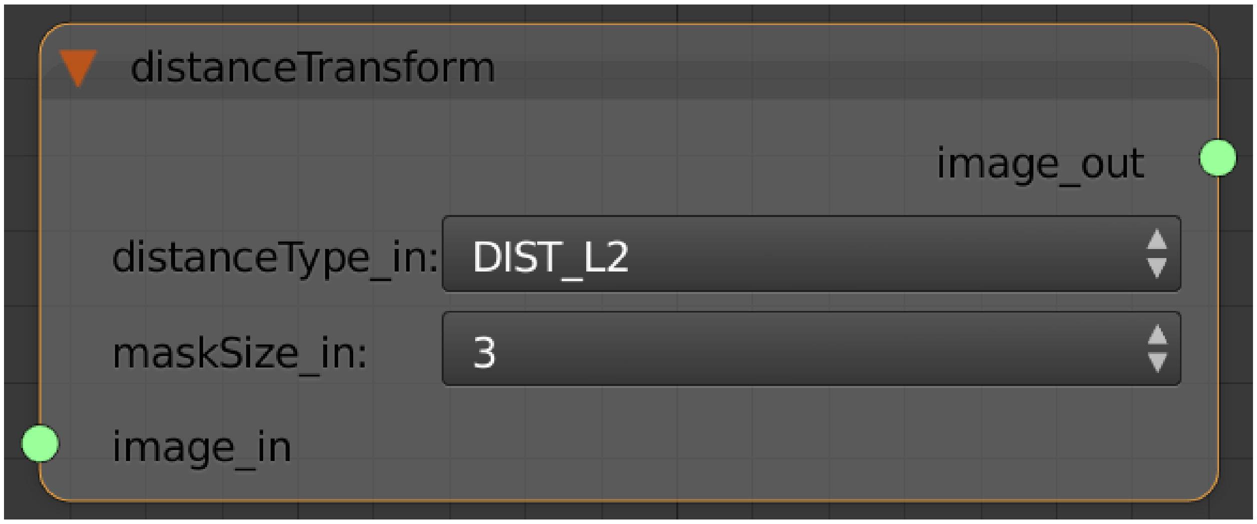
- image_out – Output image.

Locals

Examples



9.2.27 distanceTransform



Functionality

Calculates the distance to the closest zero pixel for each pixel of the source image.

Inputs

- distanceType_in – Type of distance. It can be CV_DIST_L1, CV_DIST_L2 , or CV_DIST_C.
- image_in – 8-bit, single-channel (binary) source image.
- maskSize_in – Size of the distance transform mask.

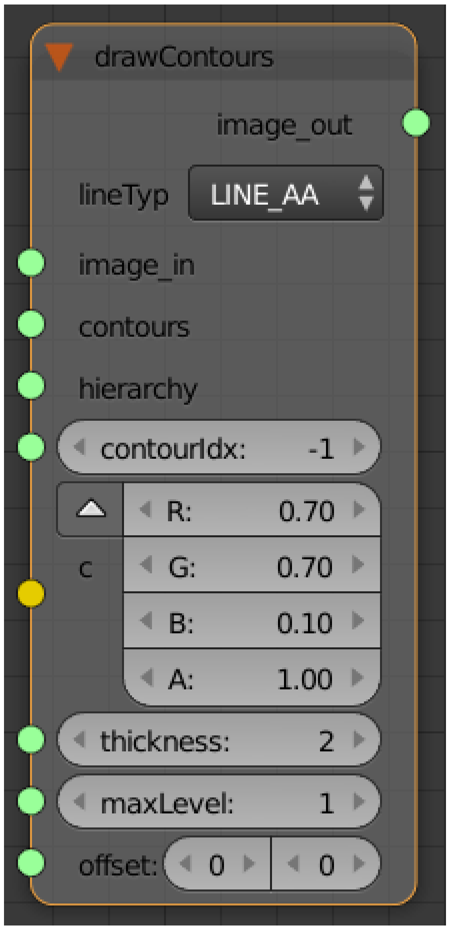
Outputs

- image_out – Output image with calculated distances.

Locals

Examples

9.2.28 drawContours



lineTyp LINE_AA

contourIdx: -1

R: 0.70

G: 0.70

B: 0.10

A: 1.00

thickness: 2

maxLevel: 1

offset: 0 0

Functionality

Draws contours outlines or filled contours.

Inputs

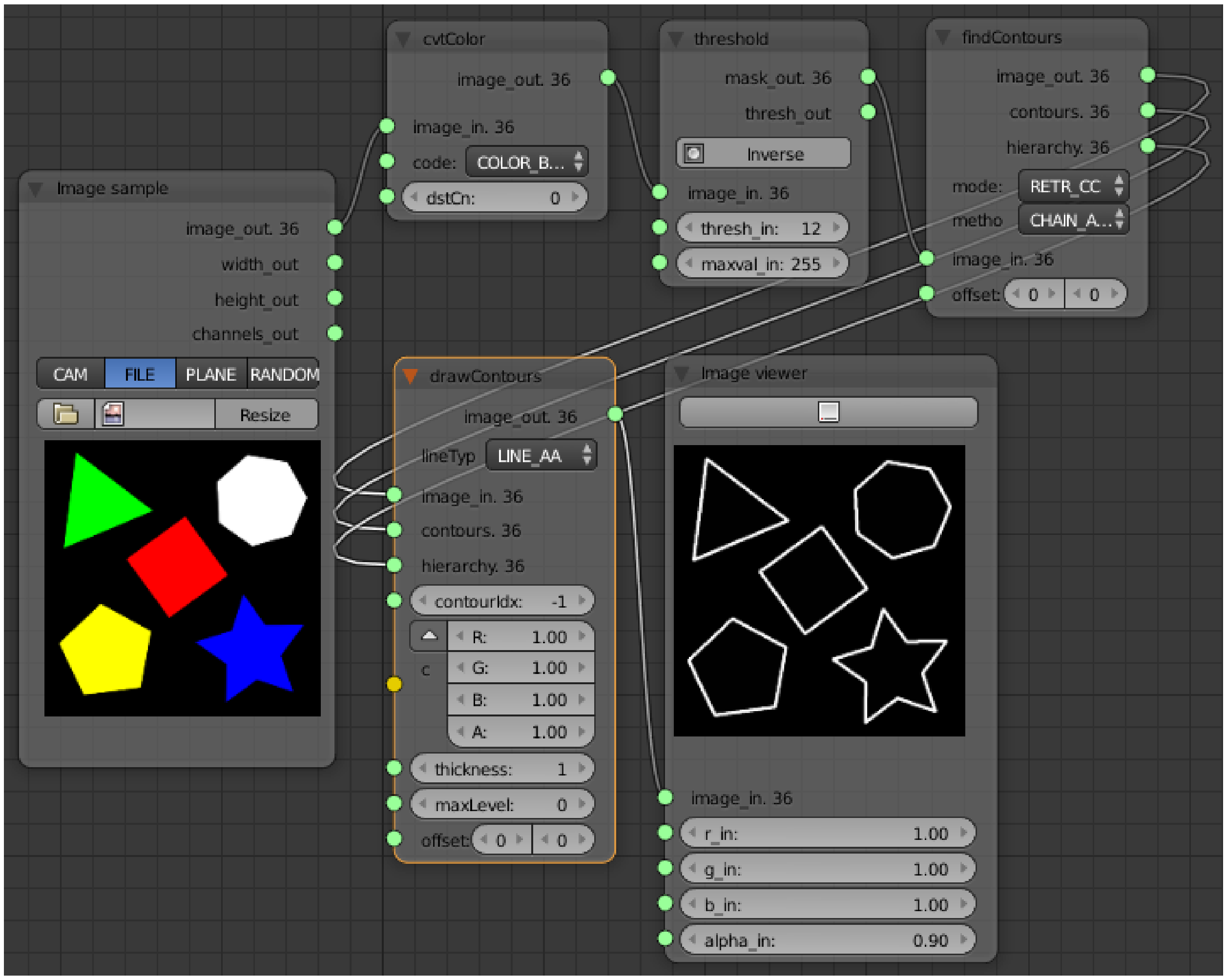
- image_in – Input image.

Outputs

- image_out – Output image.

Locals

Examples



cvtColor
image_out. 36
image_in. 36
code: COLOR_B...
dstCn: 0

threshold
mask_out. 36
thresh_out
Inverse
image_in. 36
thresh_in: 12
maxval_in: 255

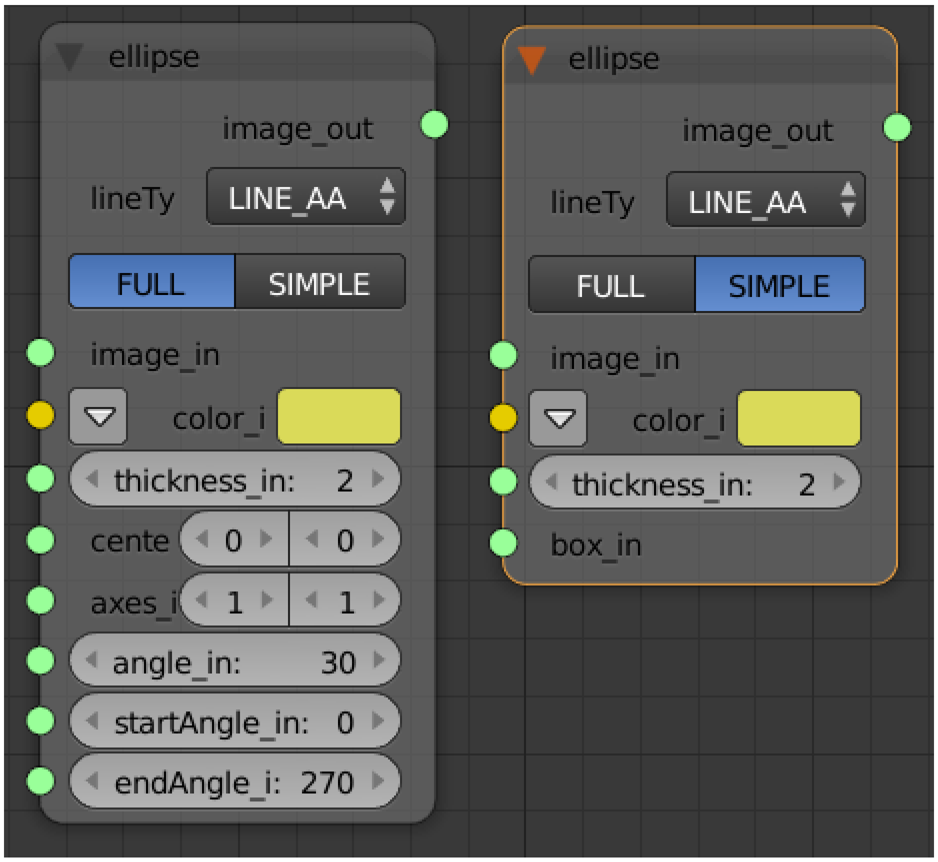
findContours
image_out. 36
contours. 36
hierarchy. 36
mode: RETR_CC
metho CHAIN_A...
image_in. 36
offset: 0 0

Image sample
image_out. 36
width_out
height_out
channels_out
CAM FILE PLANE RANDOM
Resize

drawContours
image_out. 36
lineTyp LINE_AA
image_in. 36
contours. 36
hierarchy. 36
contourIdx: -1
R: 1.00
G: 1.00
B: 1.00
A: 1.00
c
thickness: 1
maxLevel: 0
offset: 0 0

Image viewer
image_in. 36
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.29 ellipse



Functionality

Draws a simple or thick elliptic arc or fills an ellipse sector.

Inputs

- angle_in – Ellipse rotation angle in degrees.
- axes_in – Half of the size of the ellipse main axes.
- box_in – Alternative ellipse representation via RotatedRect. This means that the function draws an ellipse inscribed in the rotated rectangle.
- center_in – Center of the ellipse.
- color_in – Ellipse color.
- endAngle_in – Ending angle of the elliptic arc in degrees
- image_in – Input image.
- lineType_in – Type of the ellipse boundary. See the line description.
- startAngle_in – Starting angle of the elliptic arc in degrees.
- thickness_in – Thickness of the ellipse arc outline, if positive. Otherwise, this indicates that a filled ellipse sector is to be drawn.

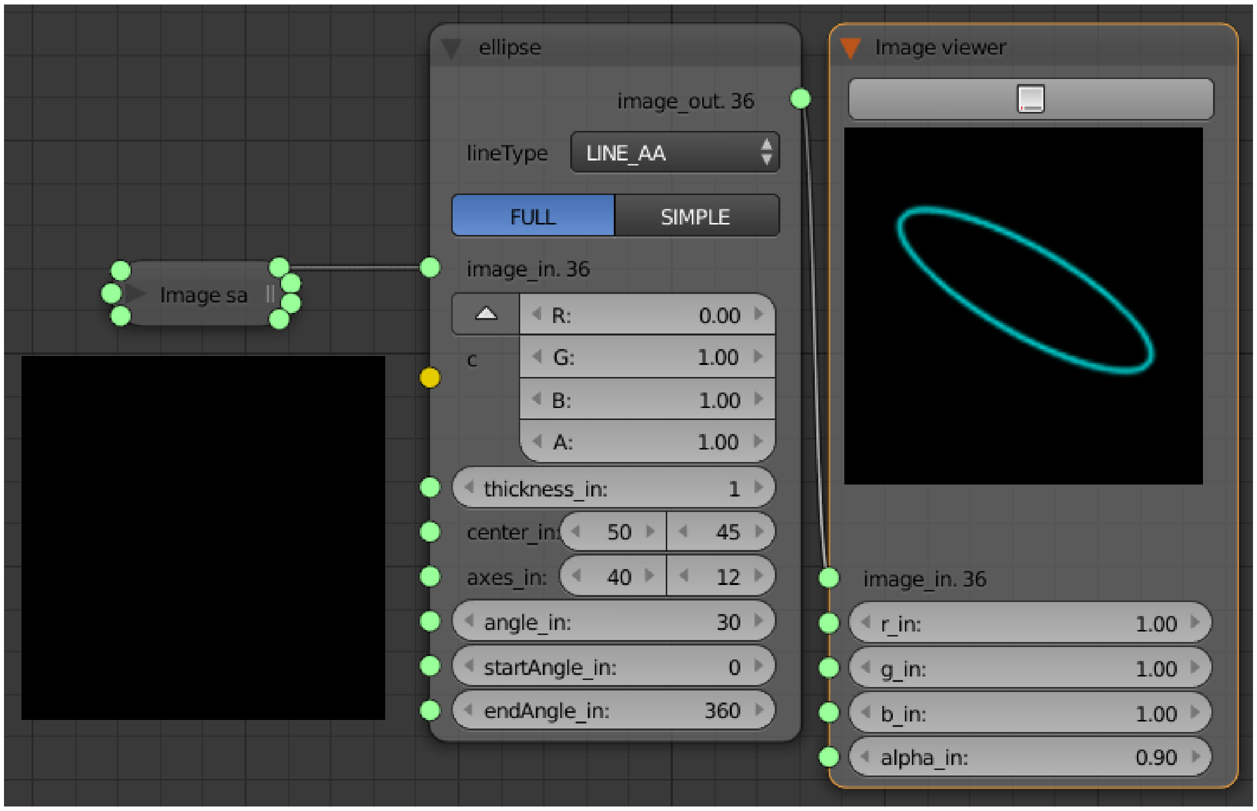
Outputs

- image_out – Output image.

Locals

- loc_input_mode – Input mode.

Examples



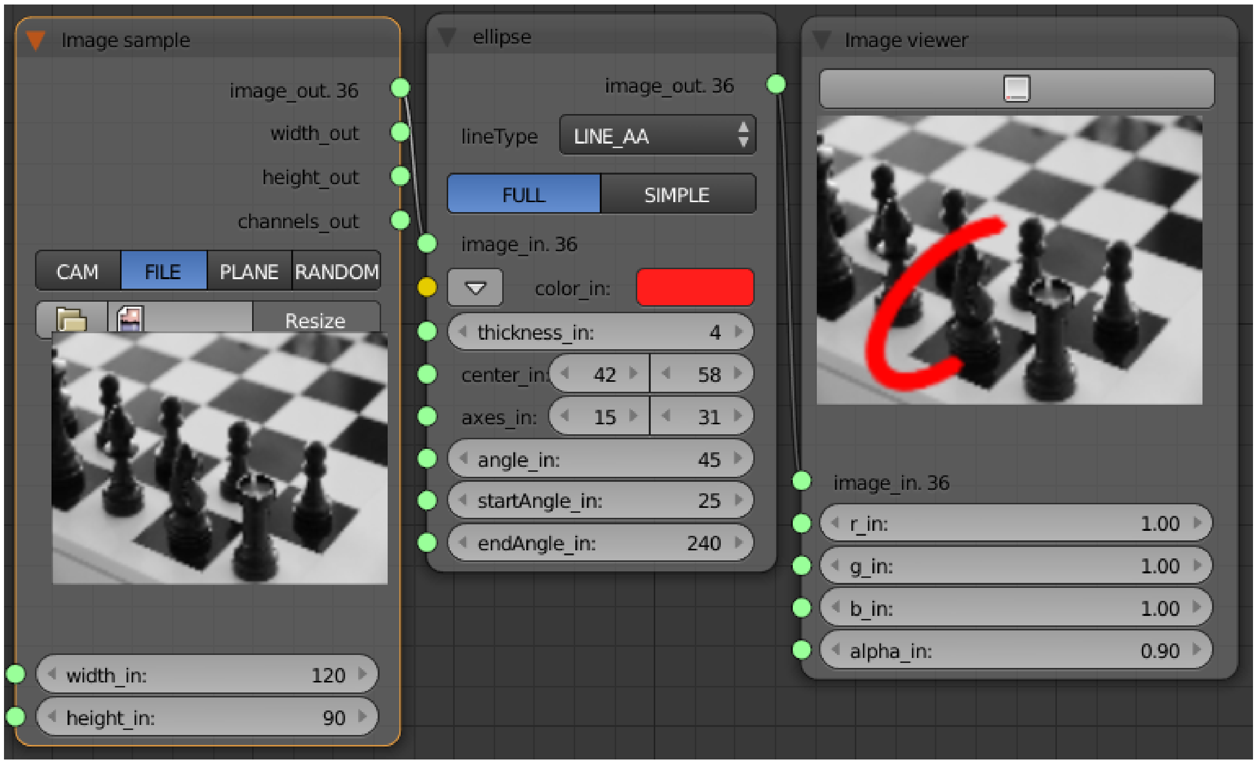


Image sample: width_in: 120, height_in: 90

ellipse: image_out. 36, lineType LINE_AA, FULL, image_in. 36, color_in: 🟥, thickness_in: 4, center_in: 42 58, axes_in: 15 31, angle_in: 45, startAngle_in: 25, endAngle_in: 240

Image viewer: image_in. 36, r_in: 1.00, g_in: 1.00, b_in: 1.00, alpha_in: 0.90

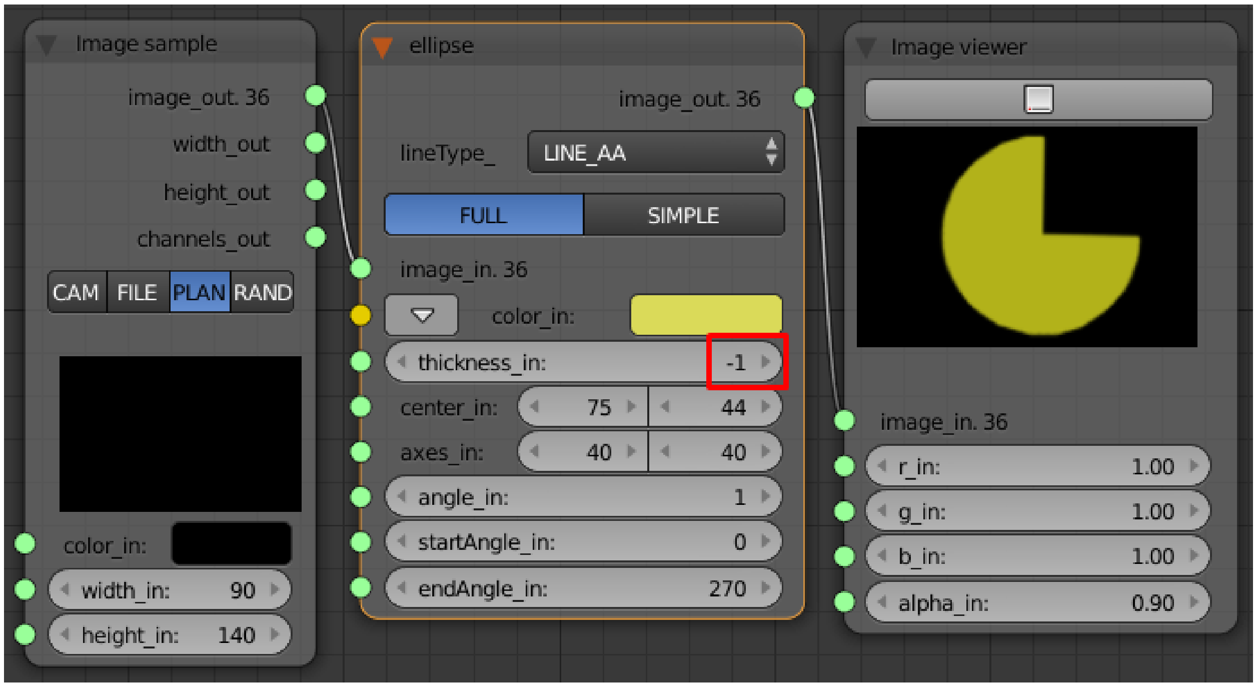


Image sample: color_in: ⬛, width_in: 90, height_in: 140

ellipse: image_out. 36, lineType_ LINE_AA, FULL, image_in. 36, color_in: 🟨, thickness_in: -1, center_in: 75 44, axes_in: 40 40, angle_in: 1, startAngle_in: 0, endAngle_in: 270

Image viewer: image_in. 36, r_in: 1.00, g_in: 1.00, b_in: 1.00, alpha_in: 0.90

9.2.30 ellipse2Poly

Functionality

Approximates an elliptic arc with a polyline.

Inputs

- angle_in – Ellipse rotation angle in degrees.
- arcEnd_in – Ending angle of the elliptic arc in degrees.
- arcStart_in – Starting angle of the elliptic arc in degrees.
- axes_in – Half of the size of the ellipse main axes.
- center_in – Center of the ellipse.
- delta_in – Angle between the subsequent polyline vertices. It defines the approximation accuracy.

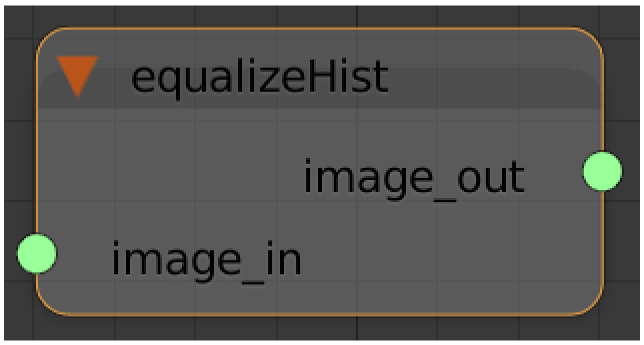
Outputs

- pts_out – Output vector of polyline vertices.

Locals

Examples

9.2.31 equalizeHist



Functionality

Equalizes the histogram of a grayscale image.

Inputs

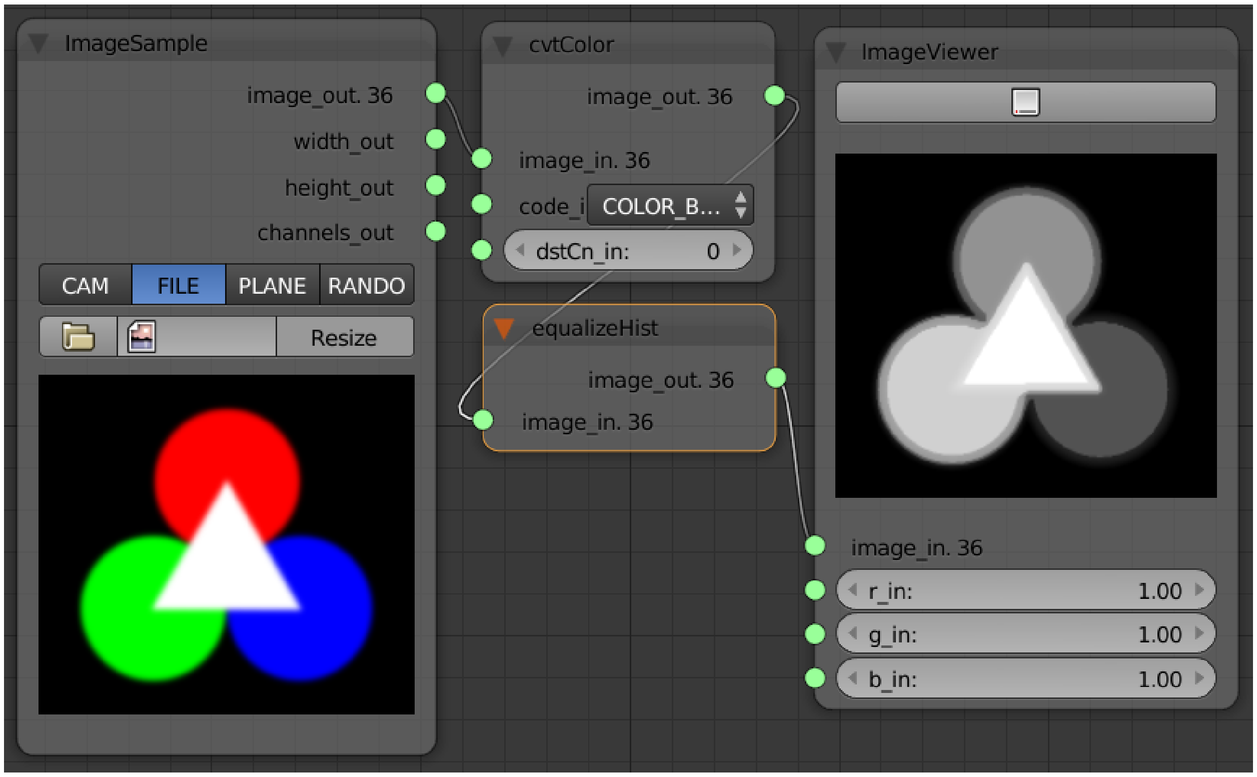
- image_in – Source 8-bit single channel image.

Outputs

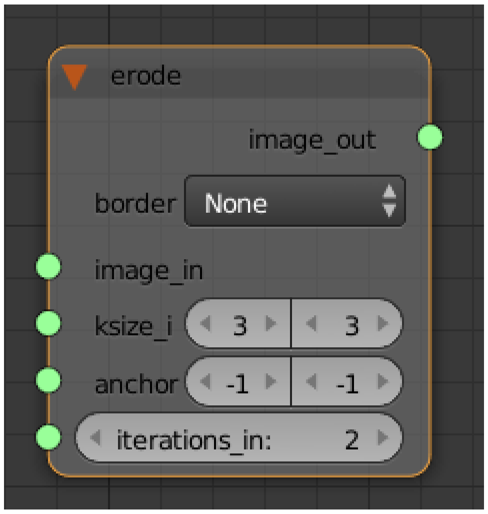
- image_out – Output image.

Locals

Examples



9.2.32 erode



Functionality

Erodes an image by using a specific structuring element.

Inputs

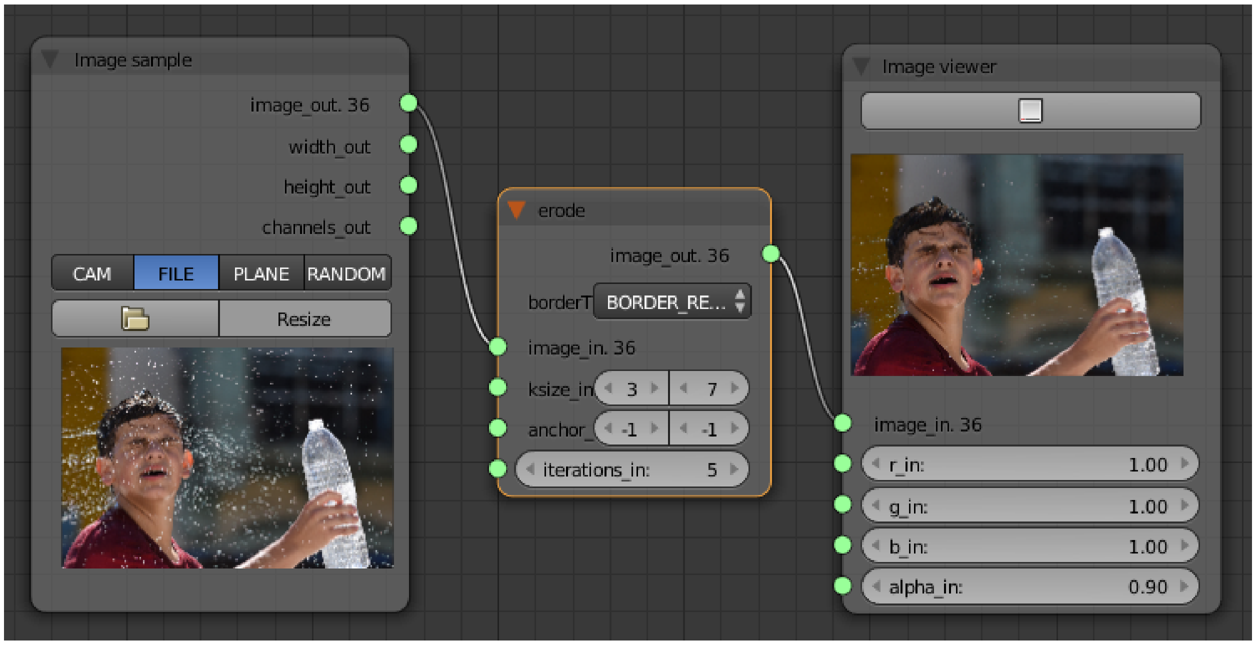
- anchor_in – Position of the anchor within the element.
- borderType_in – border mode used to extrapolate pixels outside of the image, see cv::BorderTypes
- image_in – Input image.
- iterations_in – Number of times erosion is applied.
- ksize_in – Structuring element used for erosion.

Outputs

- image_out – Output image.

Locals

Examples



ksize_in 3 7
anchor -1 -1
iterations_in: 5
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.33 filter2d

Functionality

Convolves an image with the kernel.

Inputs

- image_in – Input image.

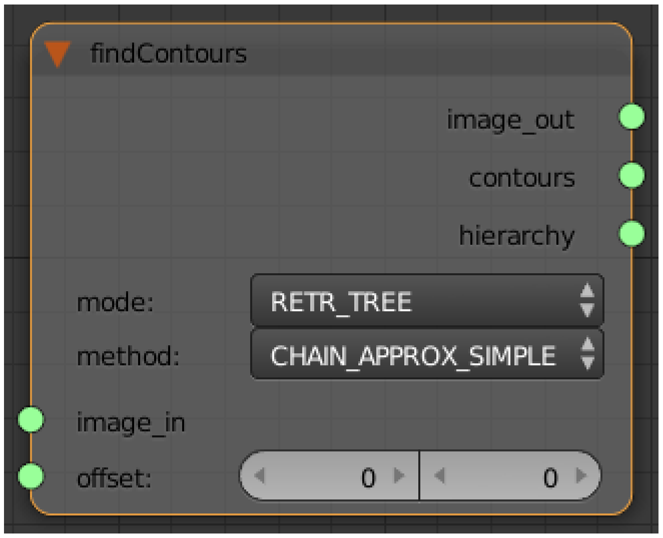
Outputs

- image_out – Output image.

Locals

Examples

9.2.34 findContours



mode: RETR_TREE
method: CHAIN_APPROX_SIMPLE
offset: 0 0

Functionality

Finds contours in a binary image.

Inputs

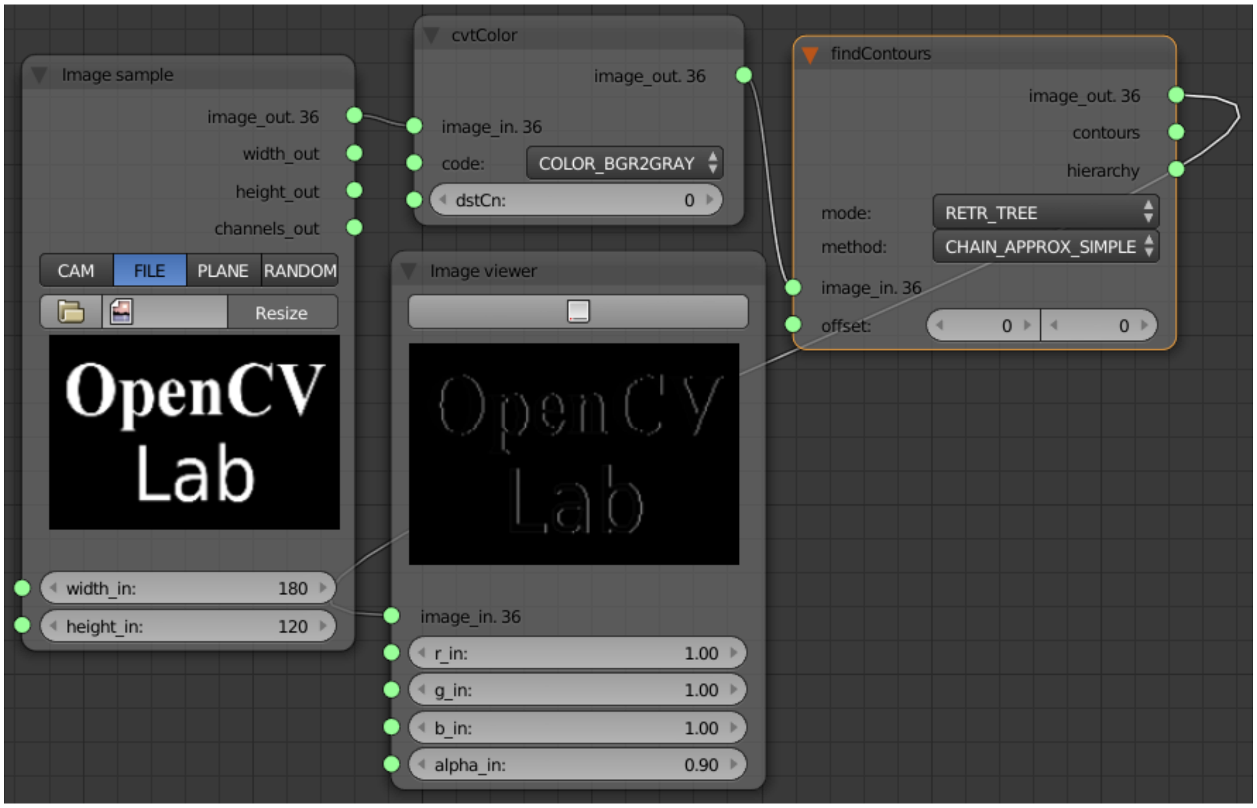
- image_in – Input image.
- method_in – Contour approximation method, see cv::ContourApproximationModes
- mode_in – Contour retrieval mode, see cv::RetrievalModes
- offset_in – Optional offset by which every contour point is shifted. This is useful if the.

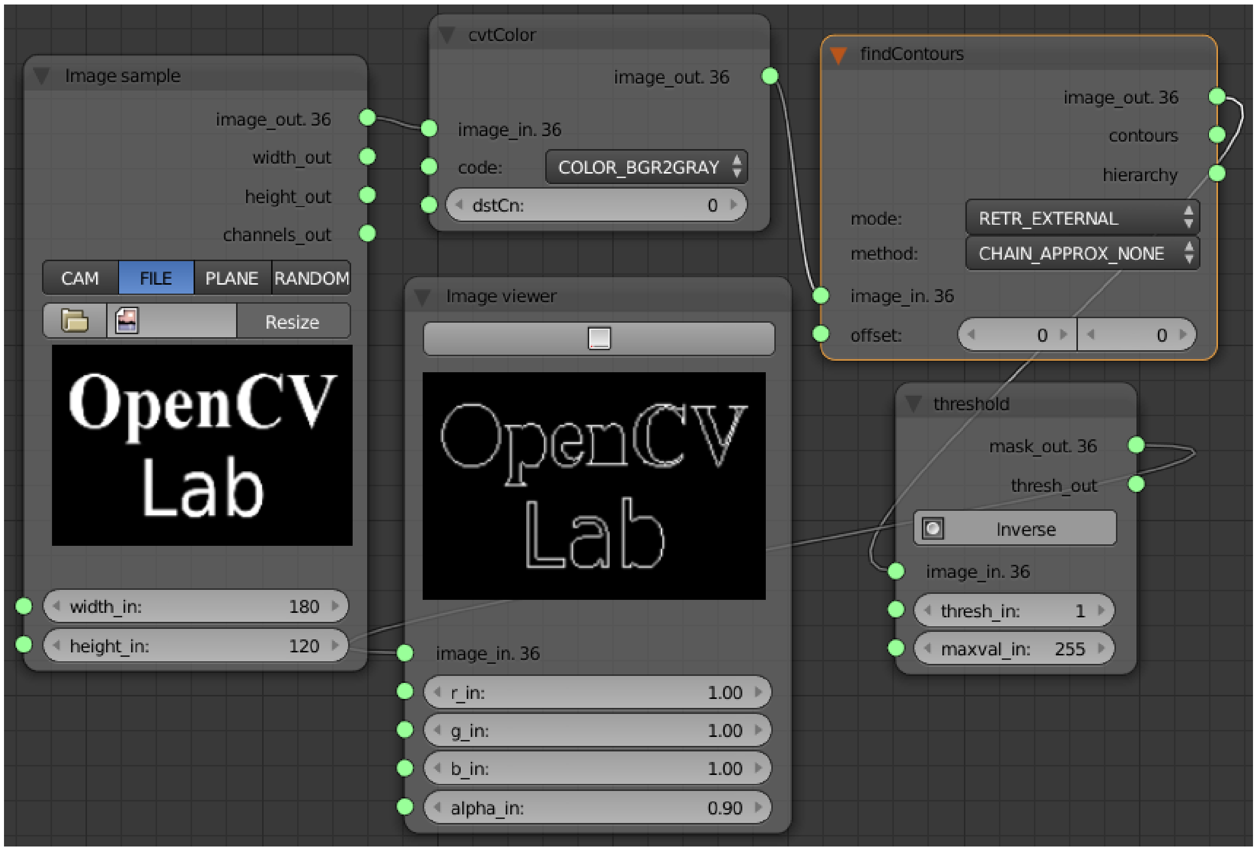
Outputs

- contours_out – Detected contours. Each contour is stored as a vector of points.
- hierarchy_out – Optional output vector, containing information about the image topology. It has as many elements as the number of contours.
- image_out – Output image.

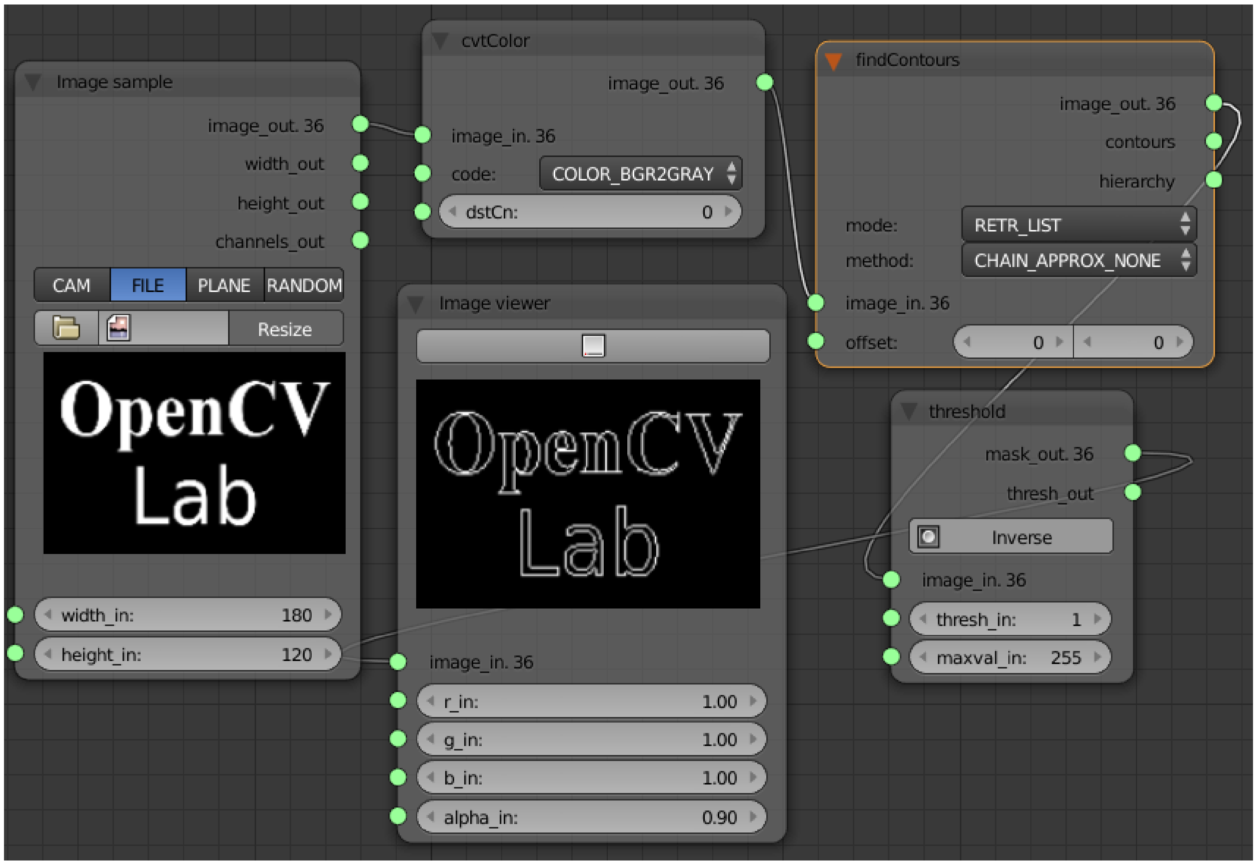
Locals

Examples

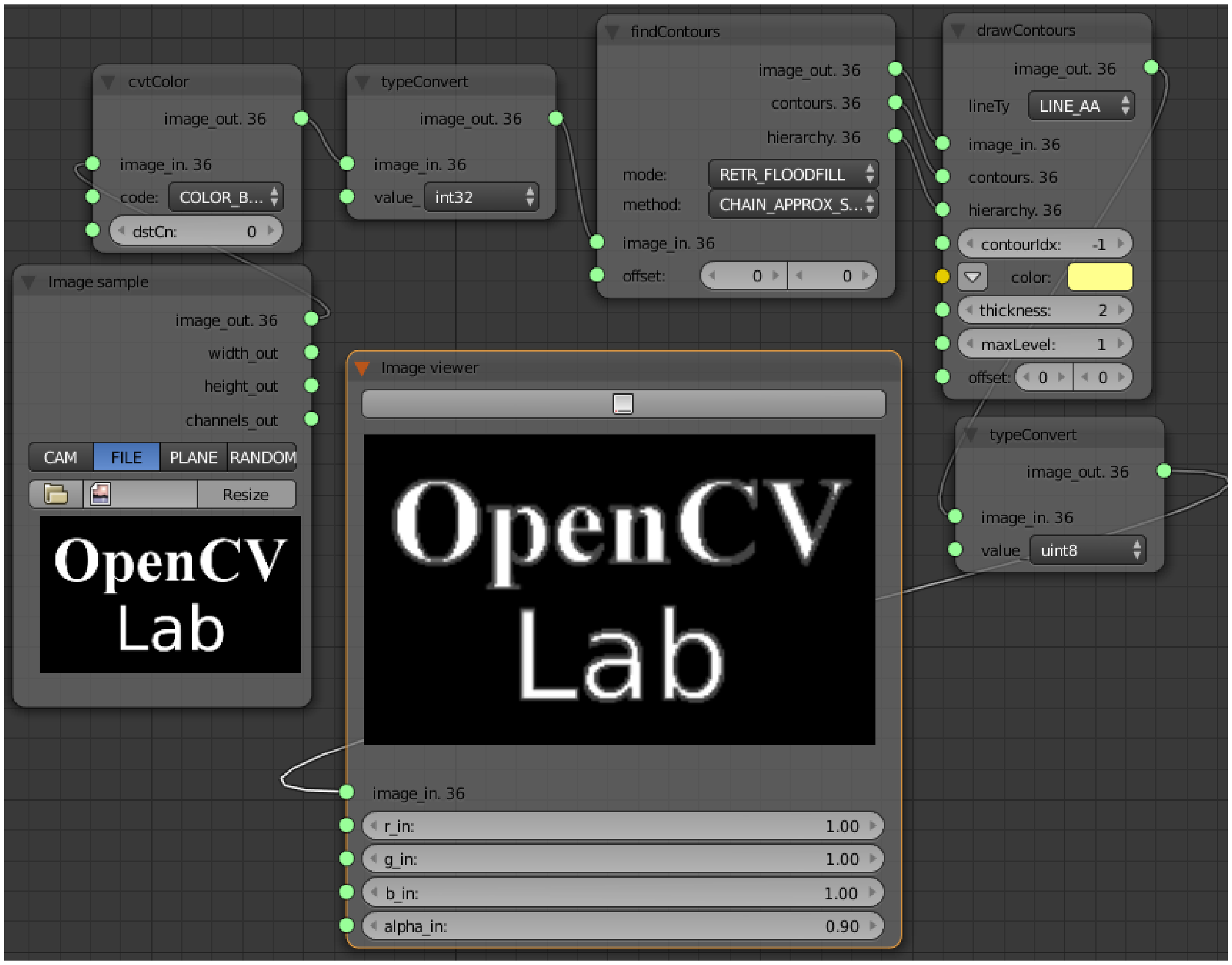




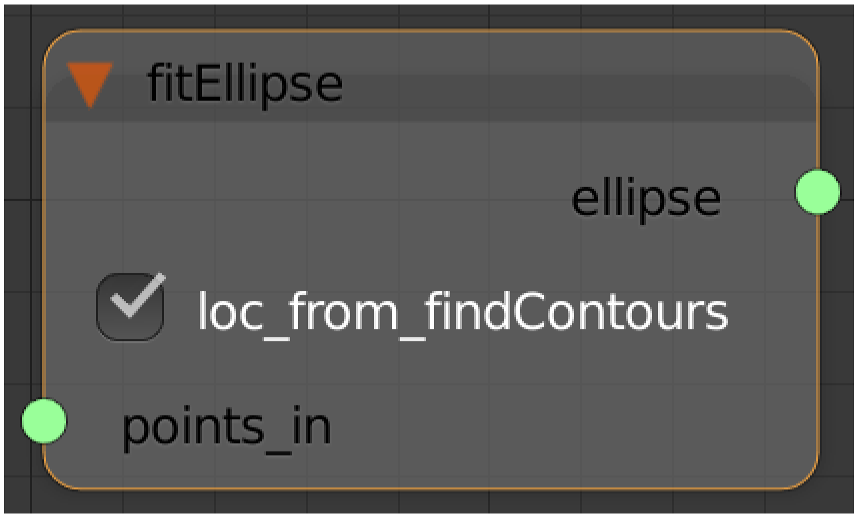
width_in: 180
height_in: 120
code: COLOR_BGR2GRAY
dstCn: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90
mode: RETR_EXTERNAL
method: CHAIN_APPROX_NONE
offset: 0 0
thresh_in: 1
maxval_in: 255



width_in: 180
height_in: 120
code: COLOR_BGR2GRAY
dstCn: 0
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90
mode: RETR_LIST
method: CHAIN_APPROX_NONE
offset: 0 0
thresh_in: 1
maxval_in: 255



9.2.35 fitEllipse



Functionality

Fits an ellipse around a set of 2D points.

Inputs

- points_in – Input vector of 2D points, stored in std::vector<> or Mat

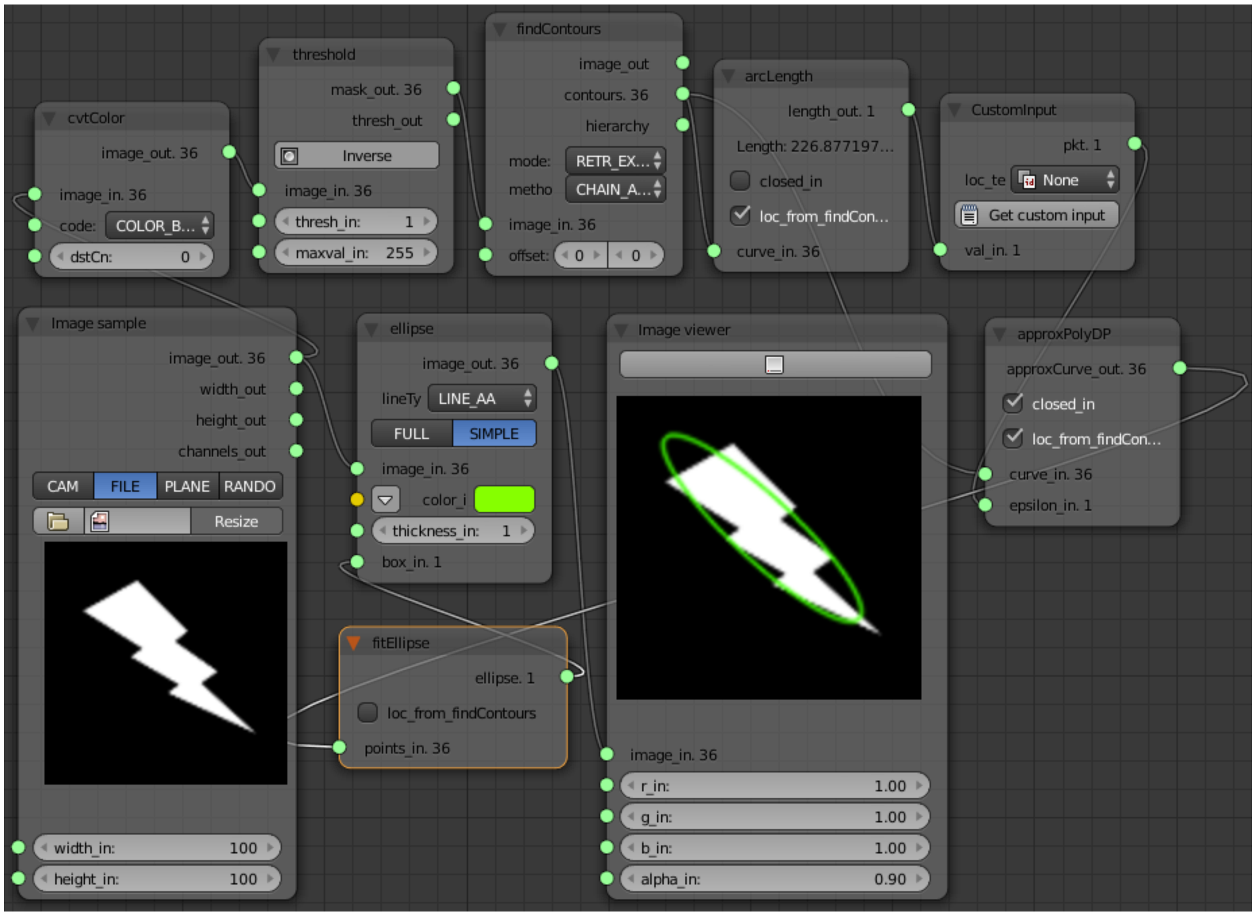
Outputs

- ellipse_out – Output ellipse.

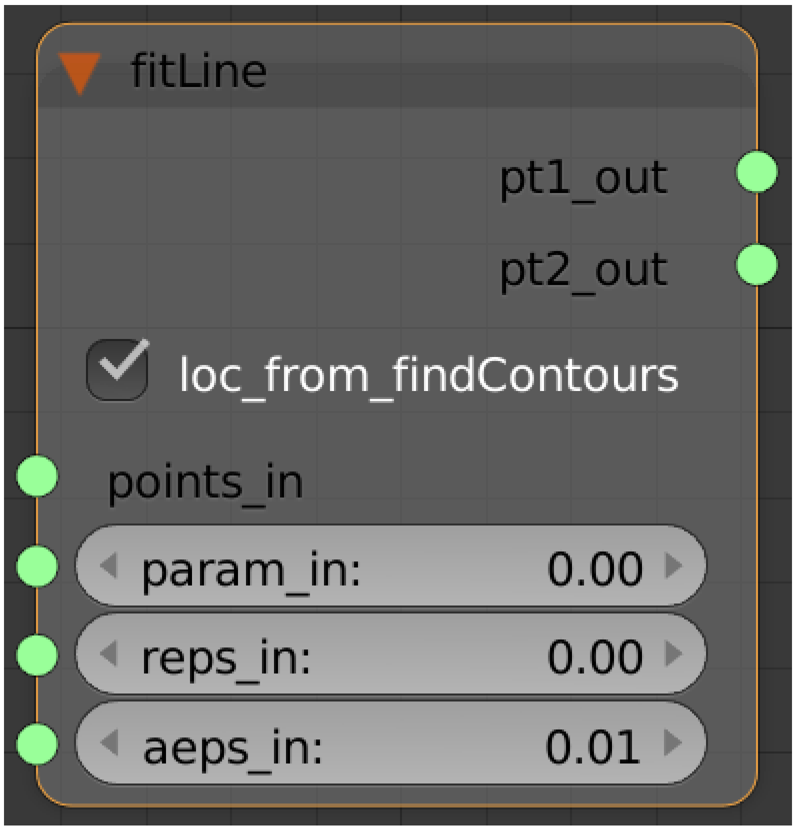
Locals

- loc_from_findContours – If linked with findContour node switch to True

Examples



9.2.36 fitLine



Functionality

Fits a line to a 2D or 3D point set.

Inputs

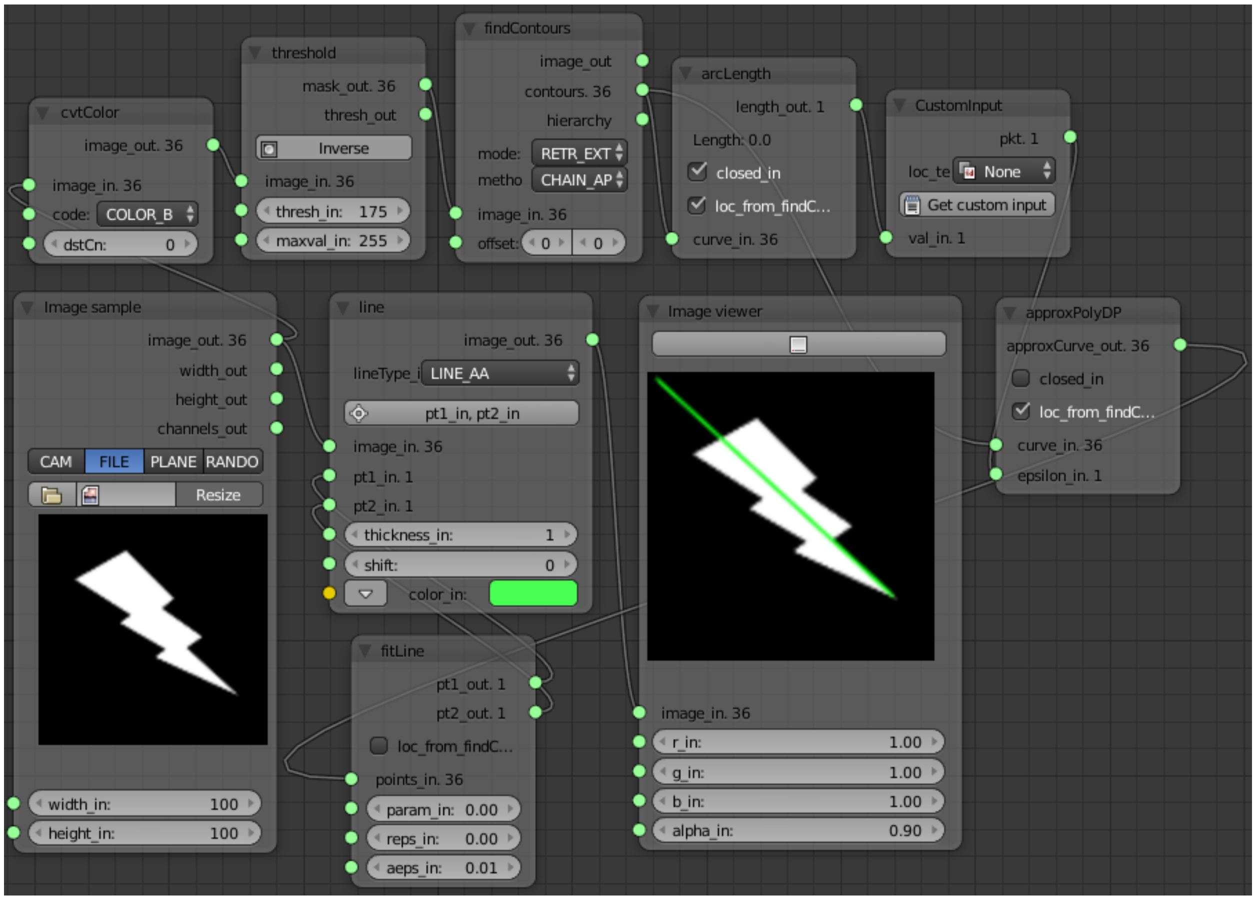
- aeps_in – Sufficient accuracy for the angle. 0.01 would be a good default value for reps and aeps.
- distType_in – Distance used by the M-estimator, see cv::DistanceTypes.
- param_in – Numerical parameter (C) for some types of distances. If it is 0, an optimal value is chosen.
- points_in – Input vector of 2D points, stored in std::vector<> or Mat
- reps_in – Sufficient accuracy for the radius (distance between the coordinate origin and the line).

Outputs

Locals

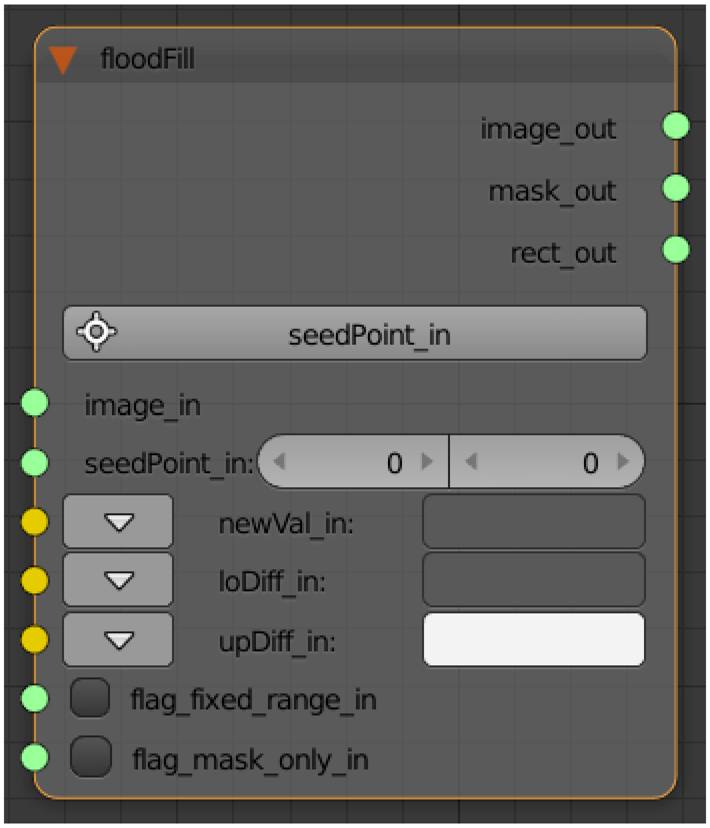
- loc_from_findContours – If linked with findContour node switch to True

Examples



code: COLOR_B
dstCn: 0
thresh_in: 175
maxval_in: 255
mode: RETR_EXT
metho CHAIN_AP
offset: 0 0
Length: 0.0
closed_in ✓
loc_from_findC... ✓
loc_te None
lineType_i LINE_AA
thickness_in: 1
shift: 0
color_in: 🟩
width_in: 100
height_in: 100
loc_from_findC...
param_in: 0.00
reps_in: 0.00
aeps_in: 0.01
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90
closed_in
loc_from_findC... ✓

9.2.37 floodFill



Functionality

Fills a connected component with the given color.

Inputs

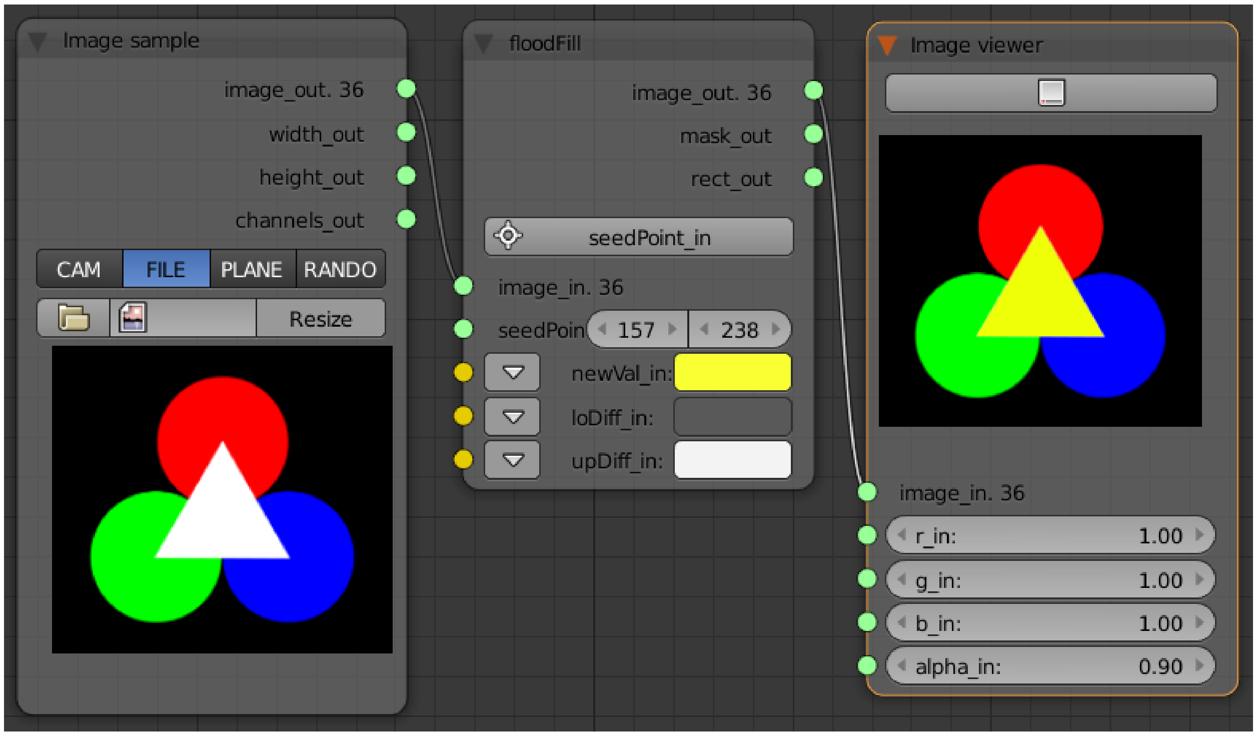
- flag_fixed_range_in – If set, the difference between the current pixel and seed pixel is considered. Otherwise, the difference between neighbor pixels is considered (that is, the range is floating).
- flag_mask_only_in – If set, the function does not change the image (newVal is ignored), and only fills the mask with the value specified in bits 8-16 of flags as described above.
- image_in – Source 8-bit single-channel image.
- loDiff_in – Maximal lower brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.
- newVal_in – New value of the repainted domain pixels.
- seedPoint_in – Starting point.
- upDiff_in – Maximal upper brightness/color difference between the currently observed pixel and one of its neighbors belonging to the component, or a seed pixel being added to the component.

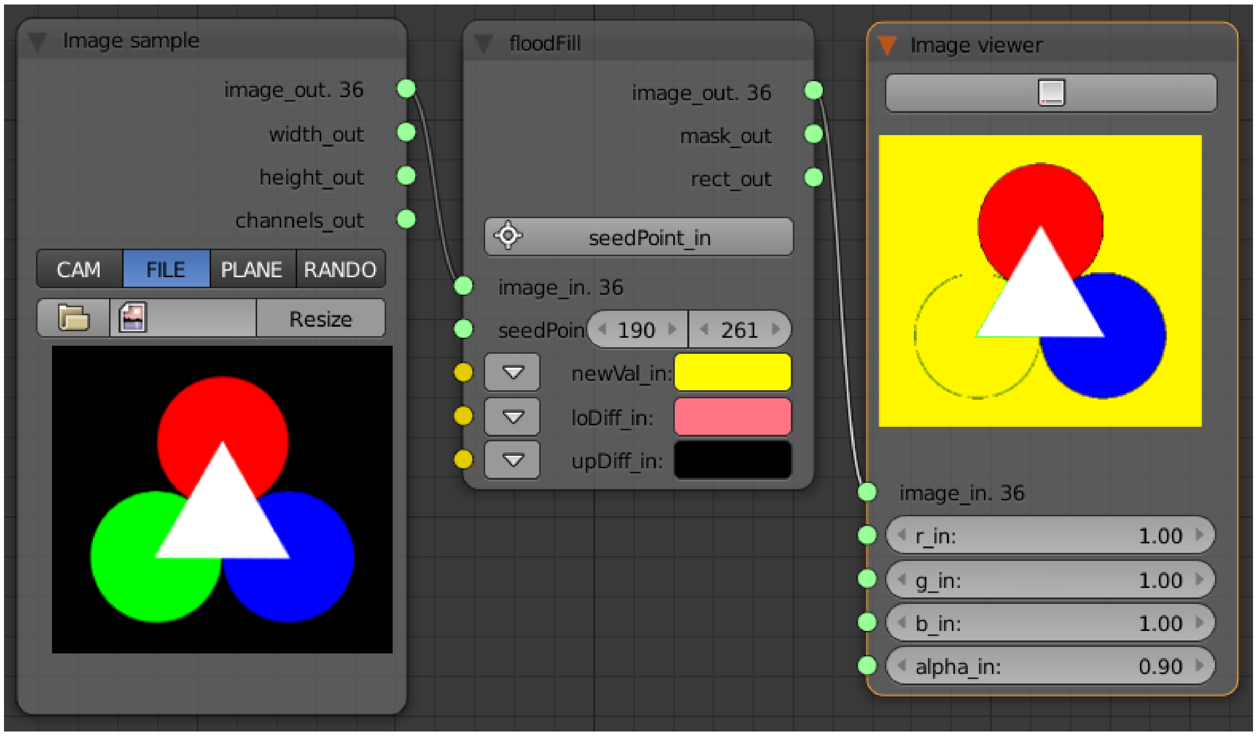
Outputs

- image_out – Destination image of the same size and the same type as src.
- rect_out – Rect output.

Locals

Examples





9.2.38 getAffineTransform

Functionality

Calculates an affine transform from three pairs of the corresponding points.

Inputs

- pts1_in – Pts1 input.
- pts2_in – Pts2 input.

Outputs

- matrix_out – Output matrix.

Locals

Examples

9.2.39 getDefaultNewCameraMatrix

Functionality

Returns the default new camera matrix.

Inputs

- cameraMatrix_in – Input camera matrix.
- centerPrincipalPoint_in – Location of the principal point in the new camera matrix. The parameter indicates whether this location should be at the image center or not.
- imgsize_in – Camera view image size in pixels.

Outputs

- retval_out – Return value.

Locals

Examples

9.2.40 GetDerivKernels

Functionality

Returns filter coefficients for computing spatial image derivatives.

Inputs

- dx_in – Derivative order in respect of x.
- dy_in – Derivative order in respect of y.
- ksize_in – Aperture size. It can be CV_SCHARR, 1, 3, 5, or 7.
- ktype_in – Type of filter coefficients. It can be CV_32f or CV_64F.
- normalize_in – Flag indicating whether to normalize (scale down) the filter coefficients or not.

Outputs

- kernel_out – Output kernel.
- kx_out – Output matrix of row filter coefficients. It has the type ktype .
- ky_out – Output matrix of column filter coefficients. It has the type ktype .

Locals

Examples

9.2.41 getGaussianKernel

Functionality

Returns Gaussian filter coefficients.

Inputs

- ksize_in – Aperture size. It should be odd.
- ktype_in – Type of filter coefficients. It can be CV_32f or CV_64F.
- sigma_in – Gaussian standard deviation.

Outputs

- kernel_out – Output kernel.

Locals

Examples

9.2.42 getPerspectiveTransform

Functionality

Calculates a perspective transform from four pairs of the corresponding points.

Inputs

- pts1_in – Input pts1.
- pts2_in – Input pts2.

Outputs

- matrix_out – Output matrix.

Locals

Examples

9.2.43 getRectSubPix

Functionality

Retrieves a pixel rectangle from an image with sub-pixel accuracy.

Inputs

- center_in – Floating point coordinates of the center of the extracted rectangle.
- image_in – Source image.
- patchSize_in – Size of the extracted patch.
- patchType_in – Depth of the extracted pixels. By default, they have the same depth as src.

Outputs

- patch_out – Patch out

Locals

Examples

9.2.44 getRotationMatrix2D

Functionality

Calculates an affine matrix of 2D rotation.

Inputs

- angle_in – Rotation angle in degrees.
- center_in – Center of the rotation in the source image.
- scale_in – Isotropic scale factor.

Outputs

- map_matrix_out – The output affine transformation, 2x3 floating-point matrix.

Locals

Examples

9.2.45 getTextSize

Functionality

Calculates the width and height of a text string.

Inputs

- fontFace_in – Font type, see cv::HersheyFonts.
- fontScale_in – Scale factor that is multiplied by the font-specific base size.
- text_in – Text string to be drawn.
- thickness_in – Thickness of the lines used to draw a text.

Outputs

- baseLine_out – Output parameter - y-coordinate of the baseline relative to the bottom-most text point.
- retval_out – Return value.

Locals

Examples

9.2.46 initUndistortRectifyMap

Functionality

Computes the undistortion and rectification transformation map.

Inputs

- R_in – Optional rectification transformation in the object space (3x3 matrix). R1 or R2 , computed by stereoRectify() can be passed here.
- cameraMatrix_in – Input camera matrix A
- distCoeffs_in – Input vector of distortion coefficients (k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]) of 4, 5, or 8 elements. If the vector is NULL/empty, the zero distortion coefficients are assumed.
- m1type_in – Type of the first output map that can be CV_32FC1 or CV_16SC2.
- size_in – Undistorted image size.

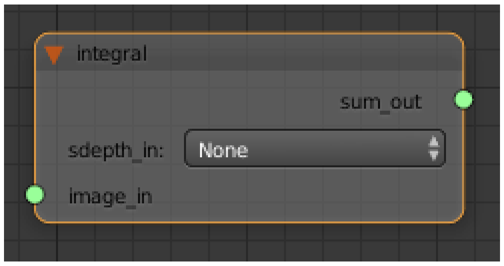
Outputs

- map1_out – First output map
- map2_out – Second output map

Locals

Examples

9.2.47 integral



sdepth_in: None

Functionality

Calculates the integral of an image.

Inputs

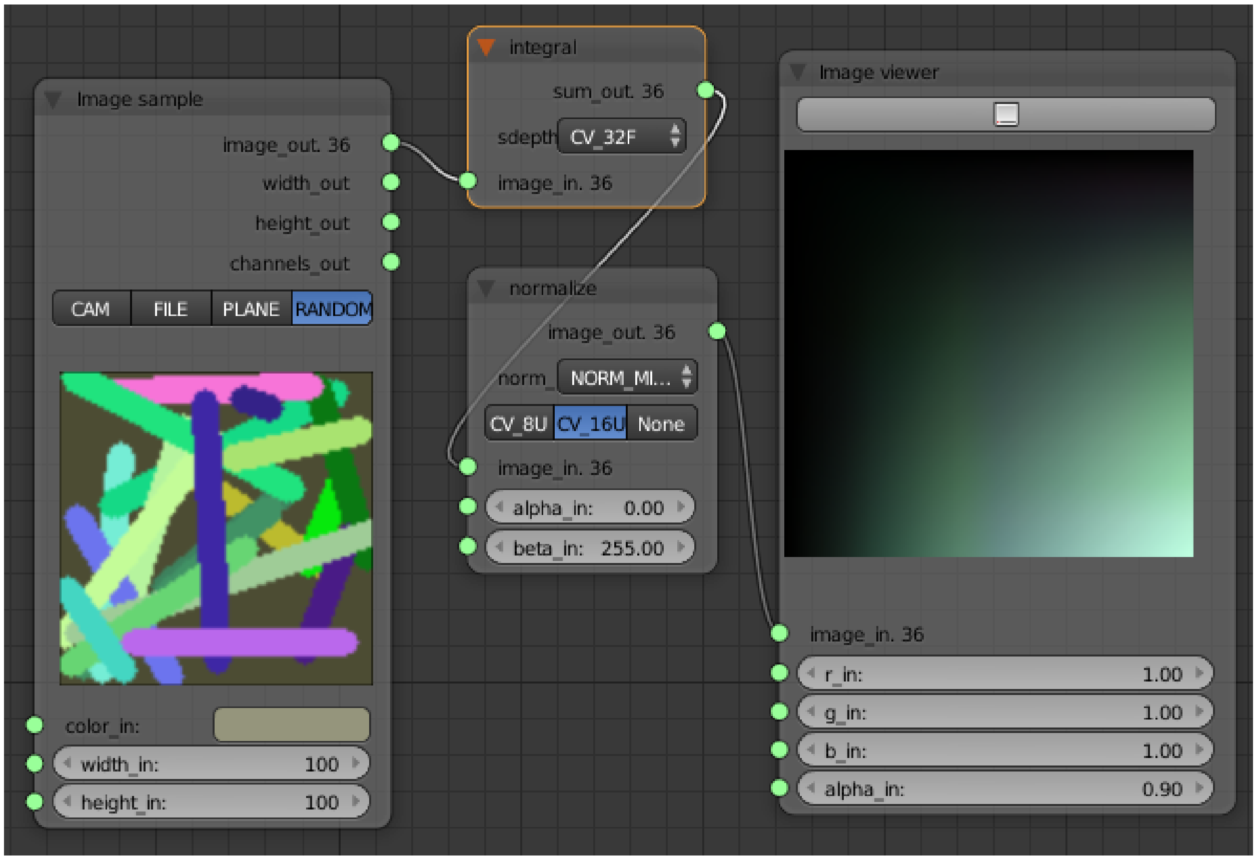
- image_in – Input image as W x H, 8-bit or floating-point (32f or 64f).
- sdepth_in – Desired depth of the integral and the tilted integral images, CV_32S, CV_32F, or CV_64F.

Outputs

- sum_out – Integral image as (W+1) x (H+1) , 32-bit integer or floating-point (32f or 64f).

Locals

Examples



width_in: 100
height_in: 100
sdepth CV_32F
norm NORM_MI...
alpha_in: 0.00
beta_in: 255.00
r_in: 1.00
g_in: 1.00
b_in: 1.00
alpha_in: 0.90

9.2.48 integral2

Functionality

Calculates the integral of an image.

Inputs

- image_in – Input image as W x H, 8-bit or floating-point (32f or 64f).

- sdepth_in – Desired depth of the integral and the tilted integral images, CV_32S, CV_32F, or CV_64F.
- sqdepth_in – Desired depth of the integral and the tilted integral images, CV_32S, CV_32F, or CV_64F.

Outputs

- sqsum_out – integral image for squared pixel values; it is (W+1) x (H+1), double-precision floating-point (64f) array.
- sum_out – Integral image as (W+1) x (H+1) , 32-bit integer or floating-point (32f or 64f).

Locals

Examples

9.2.49 integral3

Functionality

Calculates the integral of an image.

Inputs

- image_in – Input image as W x H, 8-bit or floating-point (32f or 64f).
- sdepth_in – Desired depth of the integral and the tilted integral images, CV_32S, CV_32F, or CV_64F.
- sqdepth_in – Desired depth of the integral and the tilted integral images, CV_32S, CV_32F, or CV_64F.

Outputs

- sqsum_out – integral image for squared pixel values; it is (W+1) x (H+1), double-precision floating-point (64f) array.
- sum_out – Integral image as (W+1) x (H+1) , 32-bit integer or floating-point (32f or 64f).
- tilted_out – Integral for the image rotated by 45 degrees; it is (W+1) x (H+1) array with the same data type as sum.

Locals

Examples

9.2.50 invertAffineTransform

Functionality

Inverts an affine transformation.

Inputs

- matrix_invert_in – Original affine transformation.

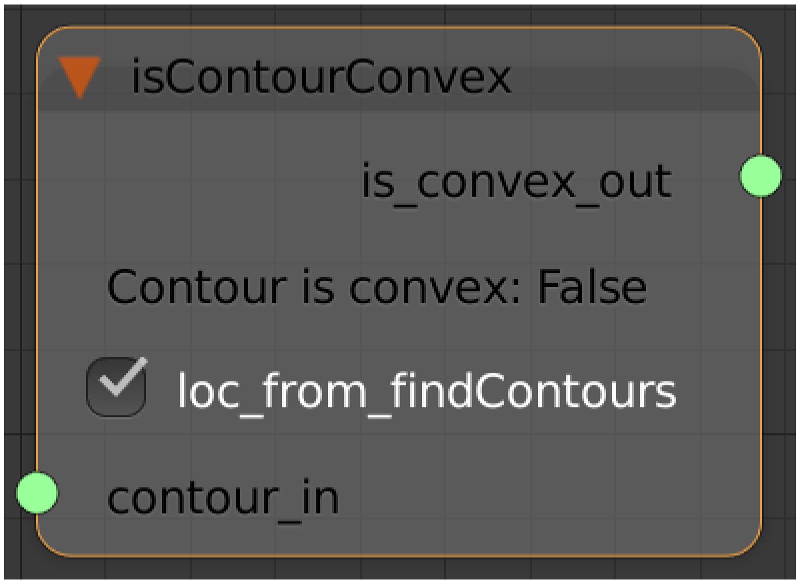
Outputs

- matrix_invert_out – Output reverse affine transformation.

Locals

Examples

9.2.51 isContourConvex



Functionality

Tests a contour convexity.

Inputs

- contour_in – Input vector of 2D points, stored in std::vector<> or Mat

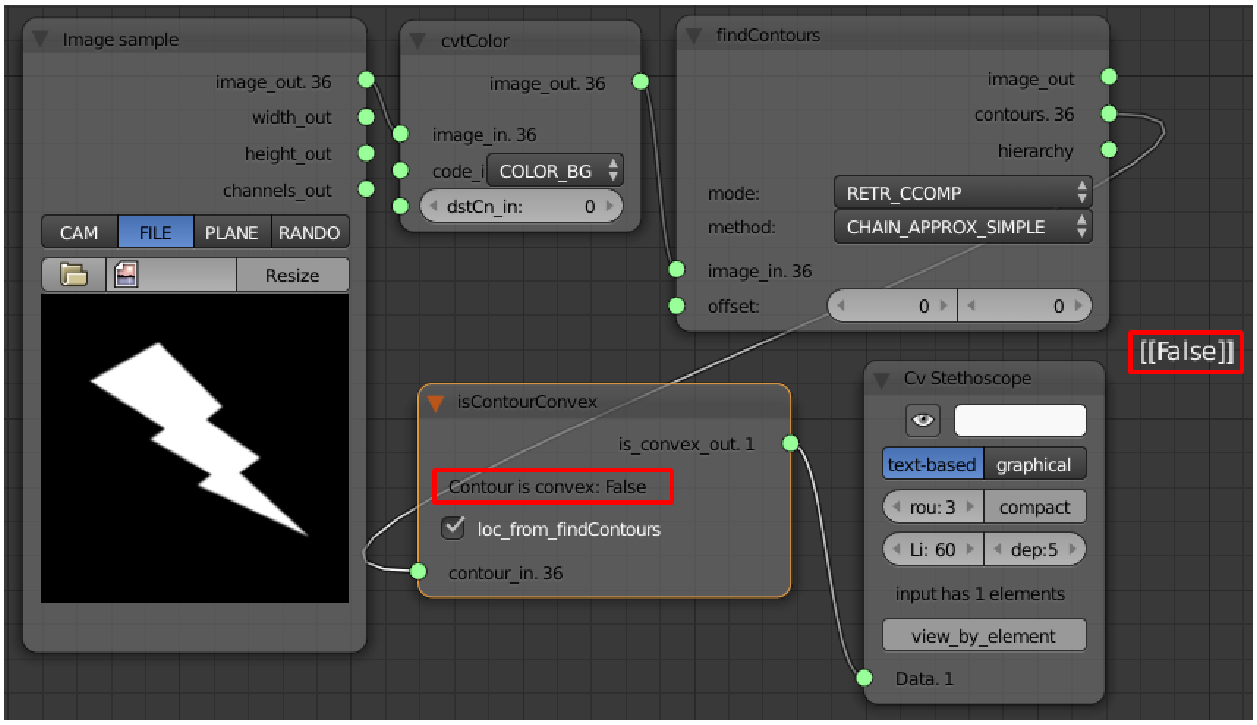
Outputs

- is_convex_out – True if contour is convex

Locals

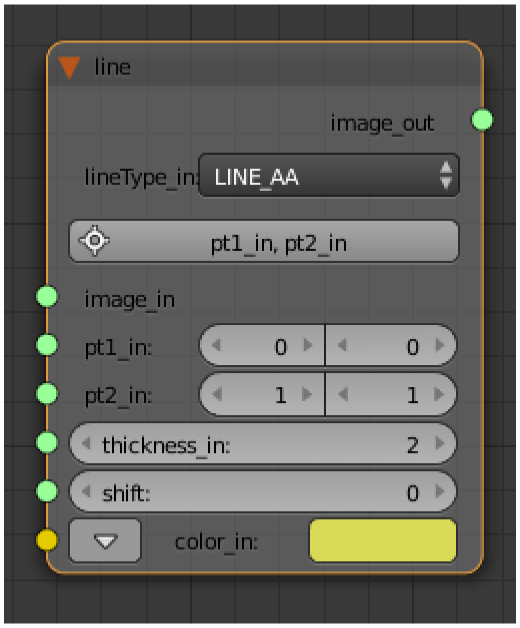
- loc_from_findContours – If linked with findContour node switch to True

Examples



code_i COLOR_BG
dstCn_in: 0
mode: RETR_CCOMP
method: CHAIN_APPROX_SIMPLE
offset: 0 0
Contour is convex: False
loc_from_findContours
rou: 3
Li: 60
dep:5
input has 1 elements
[[False]]

9.2.52 line



lineType_in: LINE_AA
pt1_in: 0 0
pt2_in: 1 1
thickness_in: 2
shift: 0
color_in:

Functionality

Draws a line segment connecting two points.

Inputs

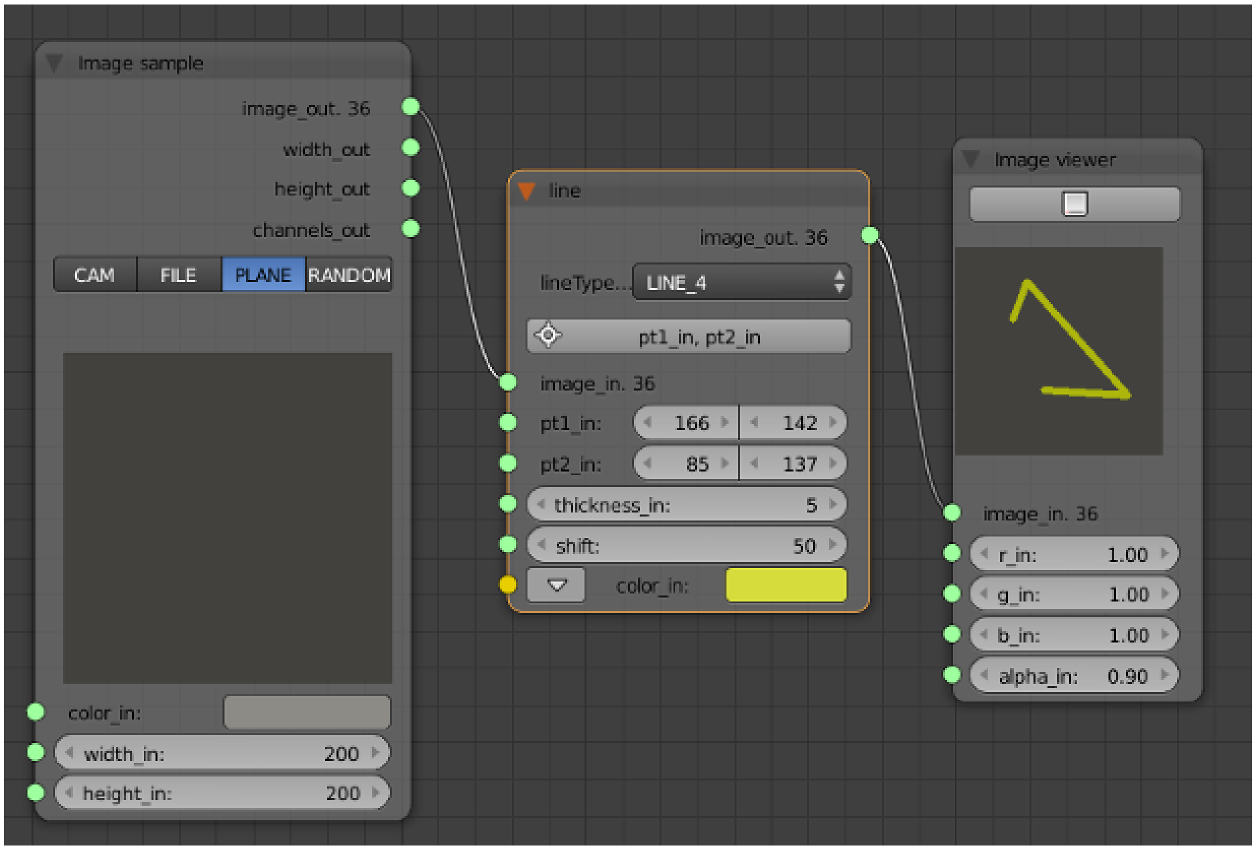
- color_in – Line color.
- image_in – Input image
- lineType_in – Line type. See the line for details.
- pt1_in – First point of the line segment.
- pt2_in – Second point of the line segment.
- thickness_in – Line thickness.

Outputs

- image_out – Output image

Locals

Examples



9.2.53 matchTemplate

Functionality

Compares a template against overlapped image regions.

Inputs

- image_in – Image where the search is running. It must be 8-bit or 32-bit floating-point.
- mask_in – Input mask.
- method_in – Parameter specifying the comparison method, see cv::TemplateMatchModes.
- templ_in – Searched template. It must be not greater than the source image and have the same data type.

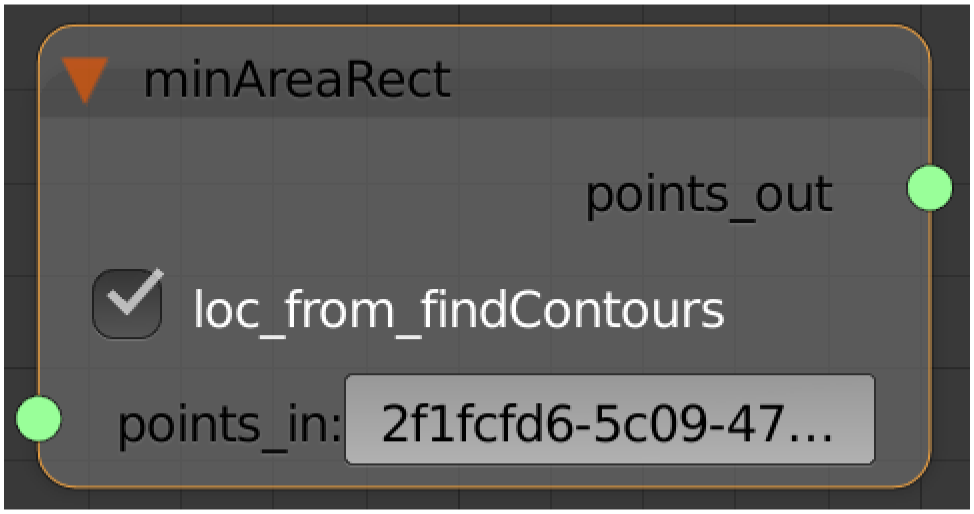
Outputs

- image_out – Output image.
- result_out – Map of comparison results. It must be single-channel 32-bit floating-point.

Locals

Examples

9.2.54 minAreaRect



Functionality

Finds a rotated rectangle of the minimum area enclosing the input 2D point set.

Inputs

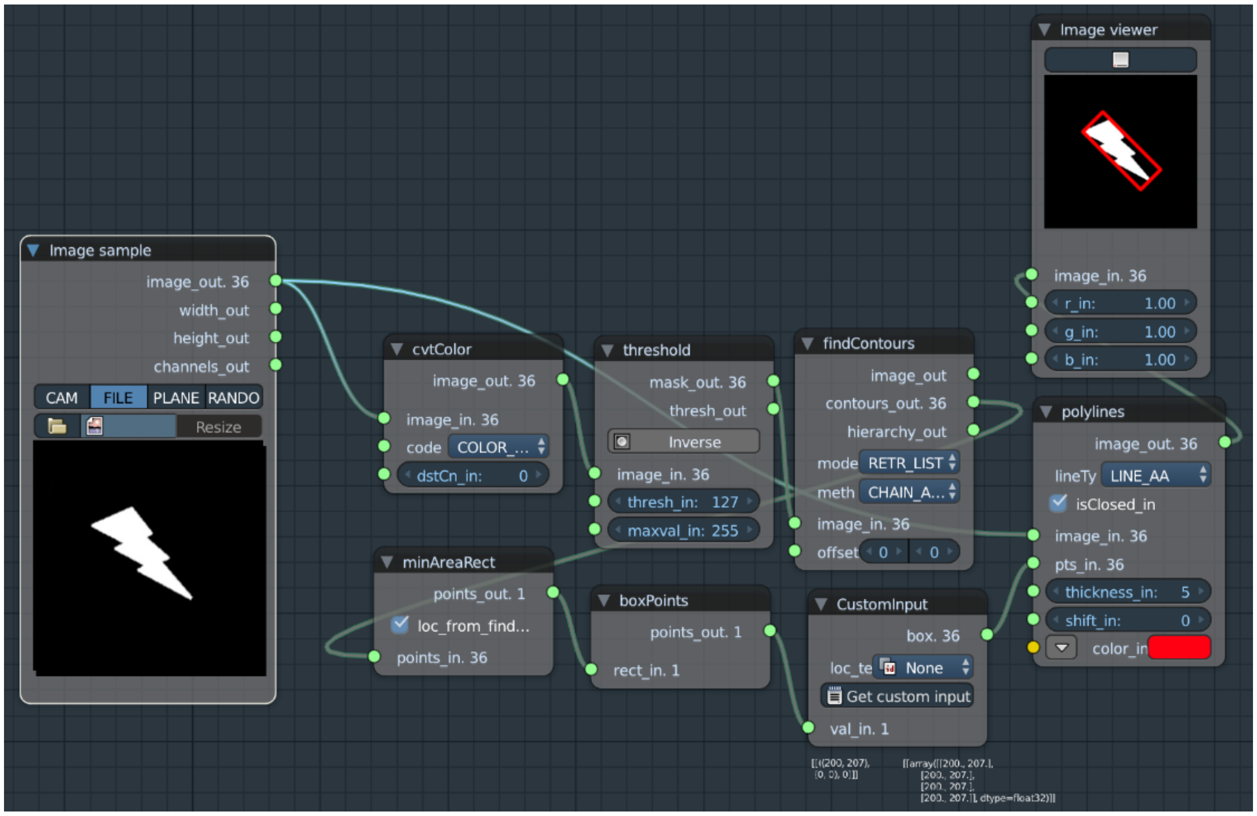
- points_in – Input vector of 2D points, stored in std::vector<> or Mat

Outputs

Locals

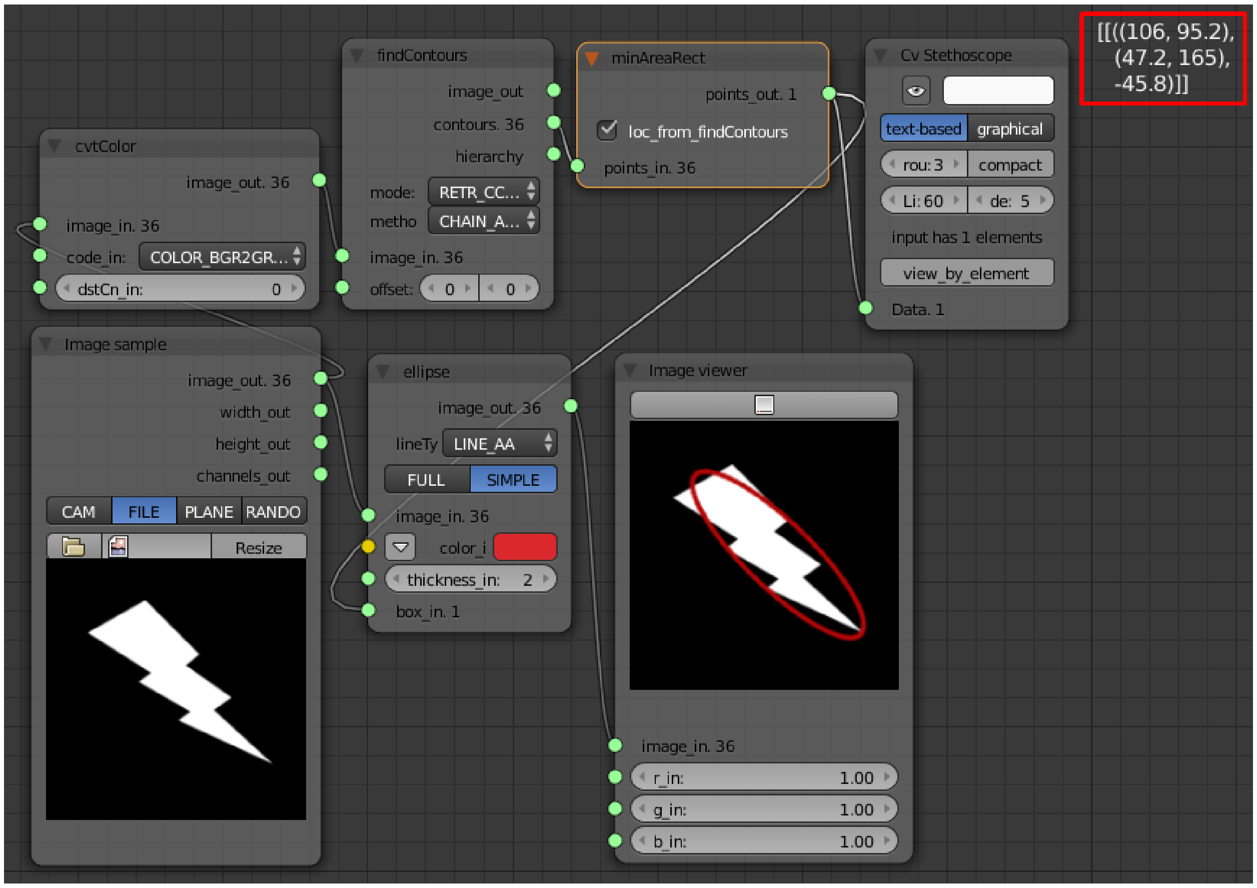
- loc_from_findContours – If linked with findContour node switch to True

Examples



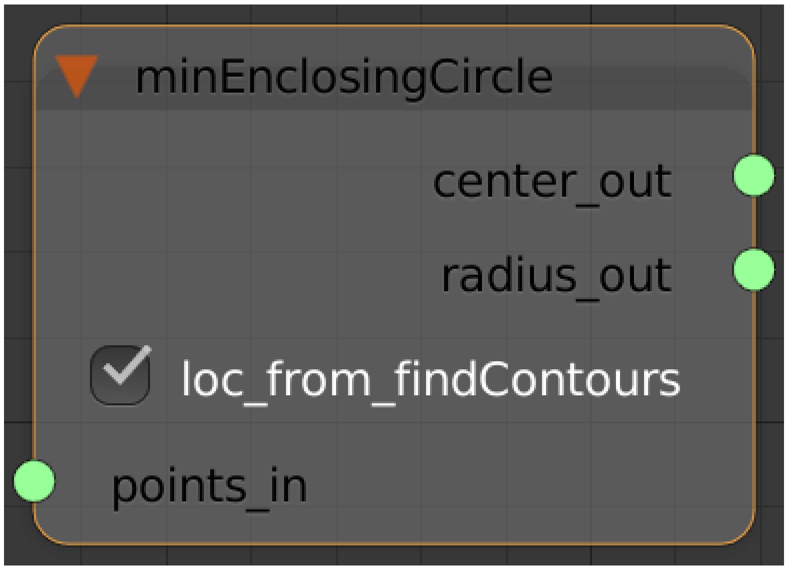
```
[[((200, 207),
  (0, 0), 0)]]
```

```
[[array([[200., 207.],
       [200., 207.],
       [200., 207.],
       [200., 207.]], dtype=float32)]]
```



```
[[((106, 95.2),
(47.2, 165),
-45.8)]]
```

9.2.55 minEnclosingCircle



Functionality

Finds a circle of the minimum area enclosing a 2D point set.

Inputs

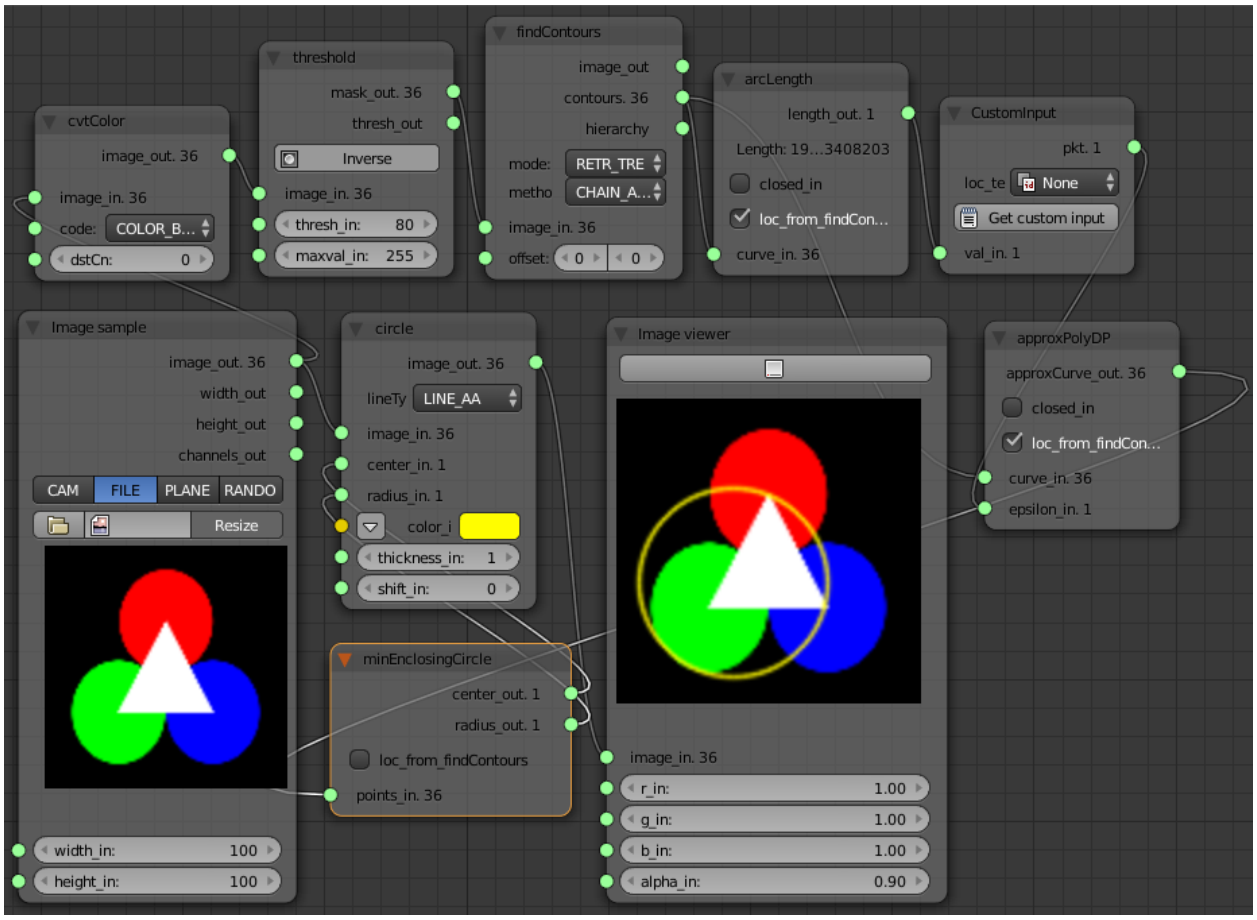
- points_in – Input vector of 2D points, stored in std::vector<> or Mat

Outputs

Locals

- loc_from_findContours – If linked with findContour node switch to True

Examples



9.2.56 moments

Functionality

Calculates all of the moments up to the third order of a polygon or rasterized shape.

Inputs

- binaryImage_in – If it is true, all non-zero image pixels are treated as 1's. The parameter is used for images only.
- image_in – Raster image (single-channel, 8-bit or floating-point 2D array) or an array

Outputs

- moments_out – Output moments.

Locals

Examples

9.2.57 morphologyEx

Functionality

Performs advanced morphological transformations.

Inputs

- anchor_in – Position of the anchor within the element.
- borderType_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes
- image_in – Source image. The number of channels can be arbitrary. The depth should be one of CV_8U, CV_16U, CV_16S, CV_32F‘ or ‘‘CV_64F.
- iterations_in – Number of times erosion is applied.
- ksize_in – Structuring element used for erosion.
- op_in – Type of a morphological operation, see cv::MorphTypes.

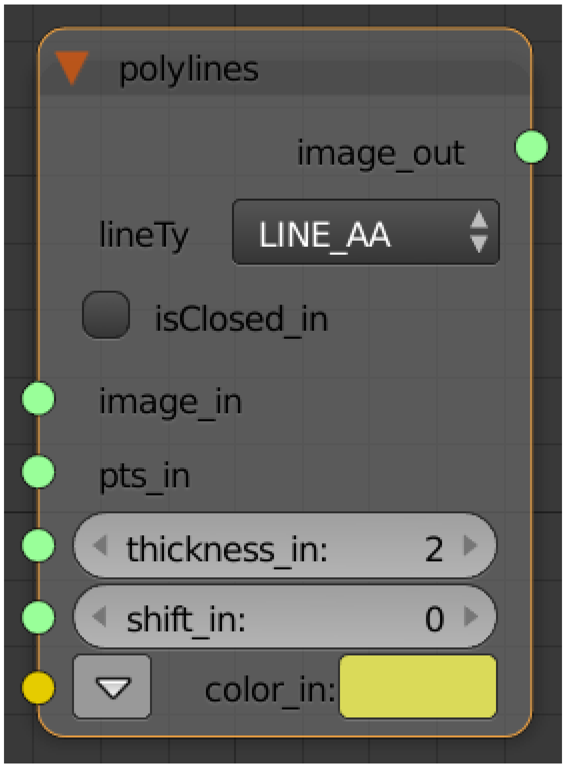
Outputs

- image_out – Destination image of the same size and type as src .

Locals

Examples

9.2.58 polylines



Functionality

Draws several polygonal curves.

Inputs

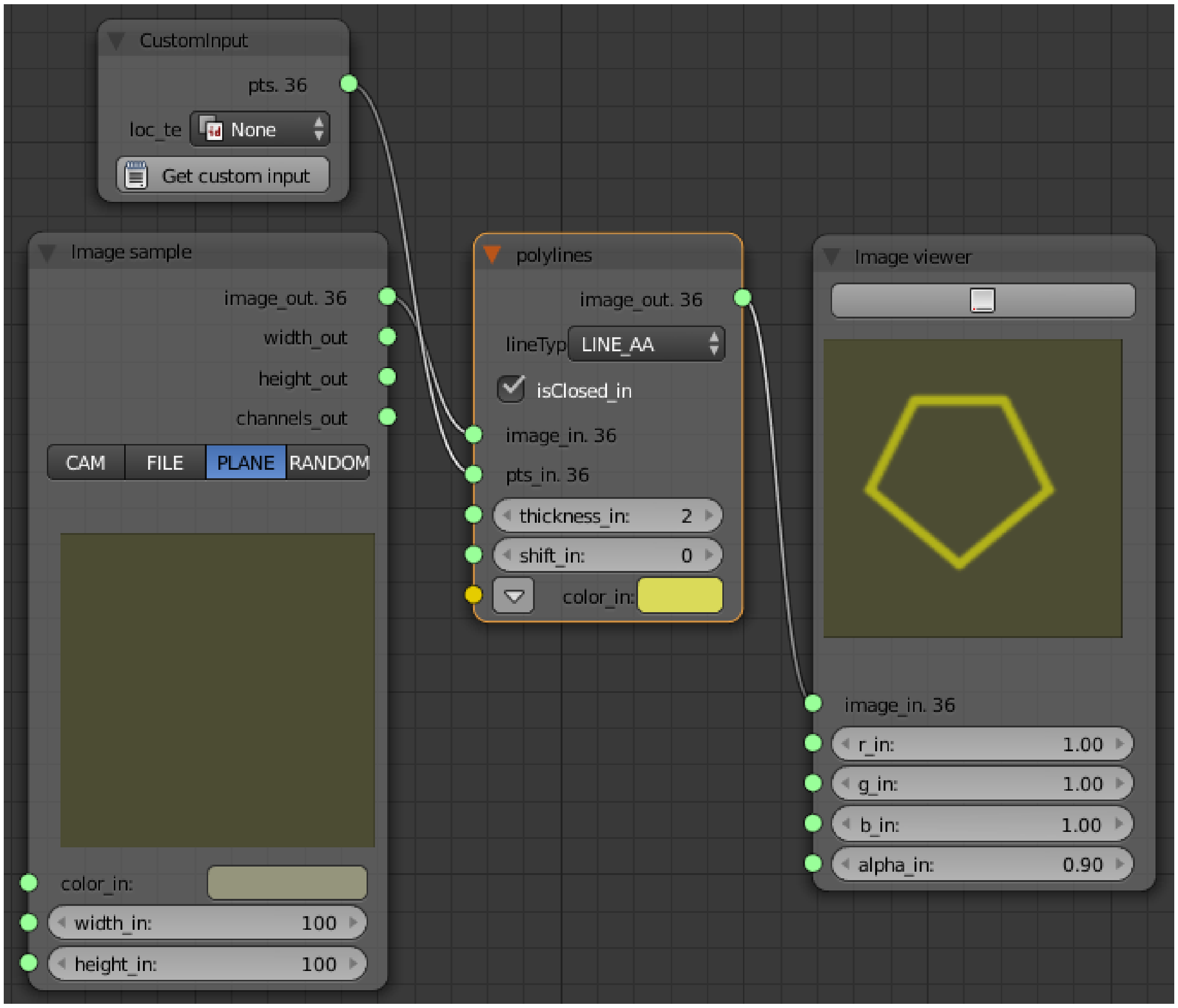
- color_in – Polyline color.
- image_in – Input image.
- isClosed_in – Flag indicating whether the drawn polylines are closed or not. If they are closed, the function draws a line from the last vertex of each curve to its first vertex.
- lineType_in – Type of the line segments. See the line description.
- pts_in – Array of polygonal curves.
- shift_in – Number of fractional bits in the vertex coordinates.
- thickness_in – Thickness of the polyline edges.

Outputs

- image_out – Output image.

Locals

Examples



loc_te None

lineTyp LINE_AA

✓ isClosed_in

thickness_in: 2

shift_in: 0

color_in:

color_in:

width_in: 100

height_in: 100

r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90

9.2.59 putText

Functionality

Draws a text string.

Inputs

- image_in – Input image.

Outputs

- image_out – Output image.

Locals

Examples

9.2.60 pyrDown

Functionality

Blurs an image and downsamples it.

Inputs

- image_in – Input image.

Outputs

- image_0_out – Image 0 output.
- image_1_out – Image 1 output.
- image_2_out – Image 2 output.
- image_3_out – Image 3 output.
- image_4_out – Image 4 output.
- image_5_out – Image 5 output.
- image_6_out – Image 6 output.
- image_7_out – Image 7 output.
- image_8_out – Image 8 output.
- image_9_out – Image 9 output.
- image_full_out – Image full output.

Locals

- loc_pyramid_size – Number levels of pyramids.

Examples

9.2.61 pyrUp

Functionality

Upsamples an image and then blurs it.

Inputs

- image_in – Input image.

Outputs

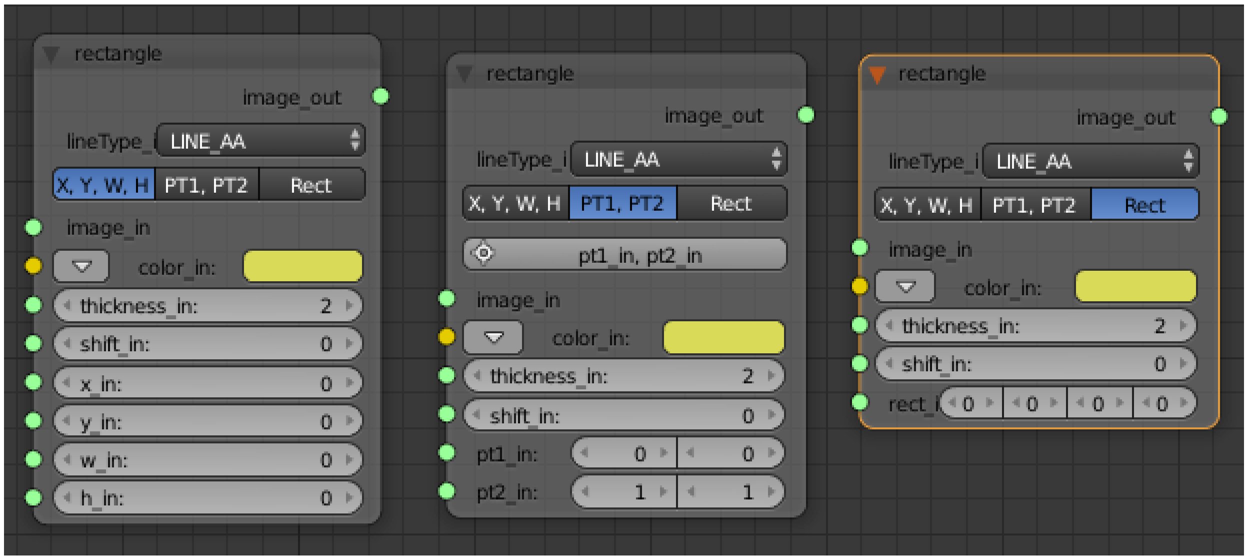
- image_0_out – Image 0 output.
- image_1_out – Image 1 output.
- image_2_out – Image 2 output.
- image_3_out – Image 3 output.
- image_4_out – Image 4 output.
- image_5_out – Image 5 output.
- image_6_out – Image 6 output.
- image_7_out – Image 7 output.
- image_8_out – Image 8 output.
- image_9_out – Image 9 output.
- image_full_out – Image full output.

Locals

- loc_pyramid_size – Number levels of pyramids.

Examples

9.2.62 rectangle



Functionality

Draws a simple, thick, or filled up-right rectangle.

Inputs

- color_in – Rectangle color or brightness (grayscale image).
- h_in – Height of rectangle.
- image_in – Input image.
- lineType_in – Type of the line. See the line description.
- pt1_in – Vertex of the rectangle.
- pt2_in – Vertex of the rectangle opposite to pt1.
- rect_in – X, Y, Weight, Height in one vector.
- shift_in – Number of fractional bits in the point coordinates.
- thickness_in – Thickness of lines that make up the rectangle. Negative values, like CV_FILLED, mean that the function has to draw a filled rectangle.
- w_in – Weight of rectangle.
- x_in – X for point of top left corner.
- y_in – Y for point of top left corner.

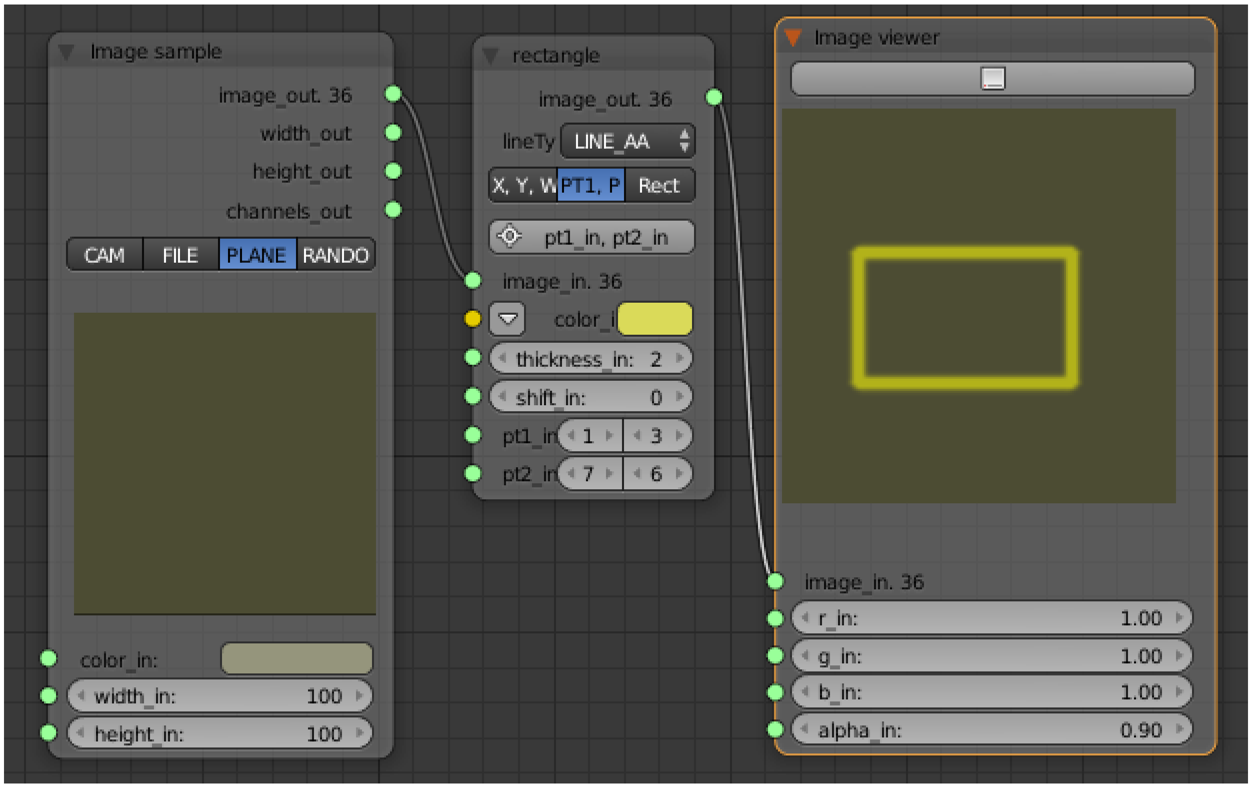
Outputs

- image_out – Output image.

Locals

- loc_input_mode – Loc input mode.

Examples



9.2.63 remap

Functionality

Applies a generic geometrical transformation to an image.

Inputs

- borderMode_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes.
- borderValue_in – Value used in case of a constant border; by default, it equals 0.
- image_in – Input image.
- interpolation_in – Interpolation method.
- map1_in – The first map of either (x,y) points or just x values having the type CV_16SC2 , CV_32FC1 , or CV_32FC2 .
- map2_in – The second map of y values having the type CV_16UC1 , CV_32FC1 , or none (empty map if map1 is (x,y) points), respectively.

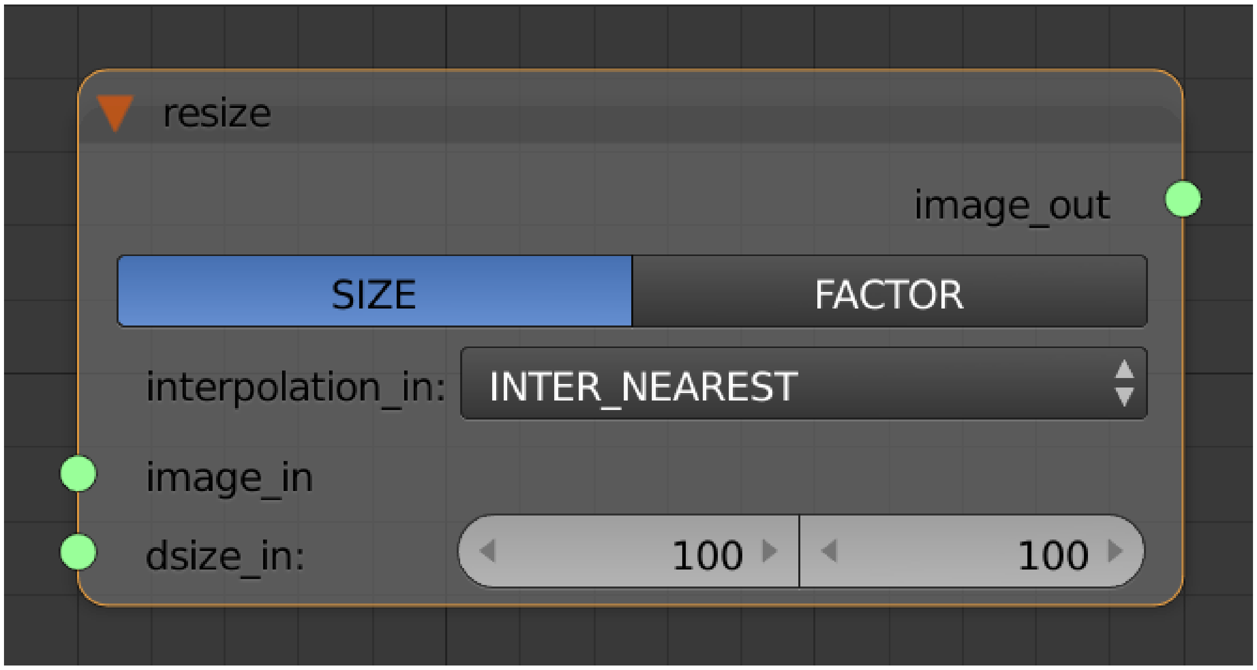
Outputs

- image_out – Output image. It has the same size as map1 and the same type as src .

Locals

Examples

9.2.64 resize



Functionality

Resizes an image.

Inputs

- dsize_in – Output image size.
- fx_in – Fx and fy and let the function compute the destination image size.
- fy_in – Fx and fy and let the function compute the destination image size.
- image_in – Input image.
- interpolation_in – Interpolation method.

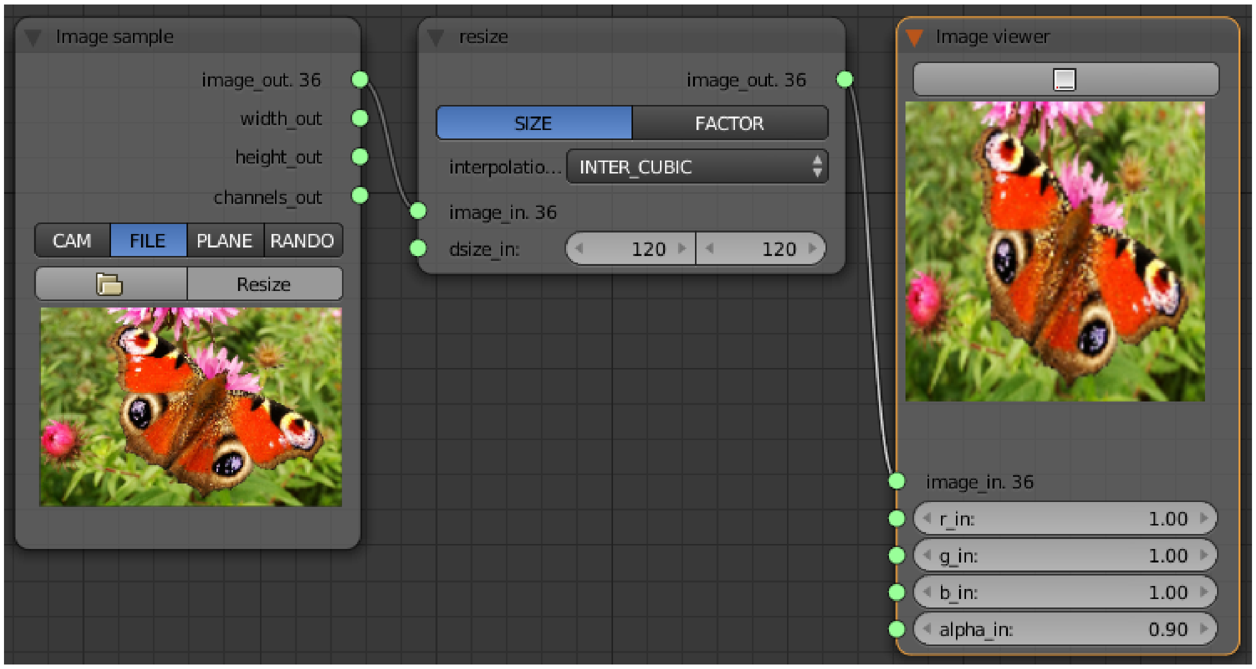
Outputs

- image_out – Output image.

Locals

- loc_resize_mode – Loc resize mode.

Examples



9.2.65 sepFilter2d

Functionality

Applies a separable linear filter to an image.

Inputs

- anchor_in – Anchor position within the kernel. The default value $(-1,-1) $ means that the anchor is at the kernel center.
- borderType_in – Pixel extrapolation method, see cv::BorderTypes
- ddepth_in – Destination image depth, see @ref filter_depths 'combinations'
- delta_in – Value added to the filtered results before storing them.
- image_in – Input image.
- kernel_size_in – Coefficients for filtering each row and column.

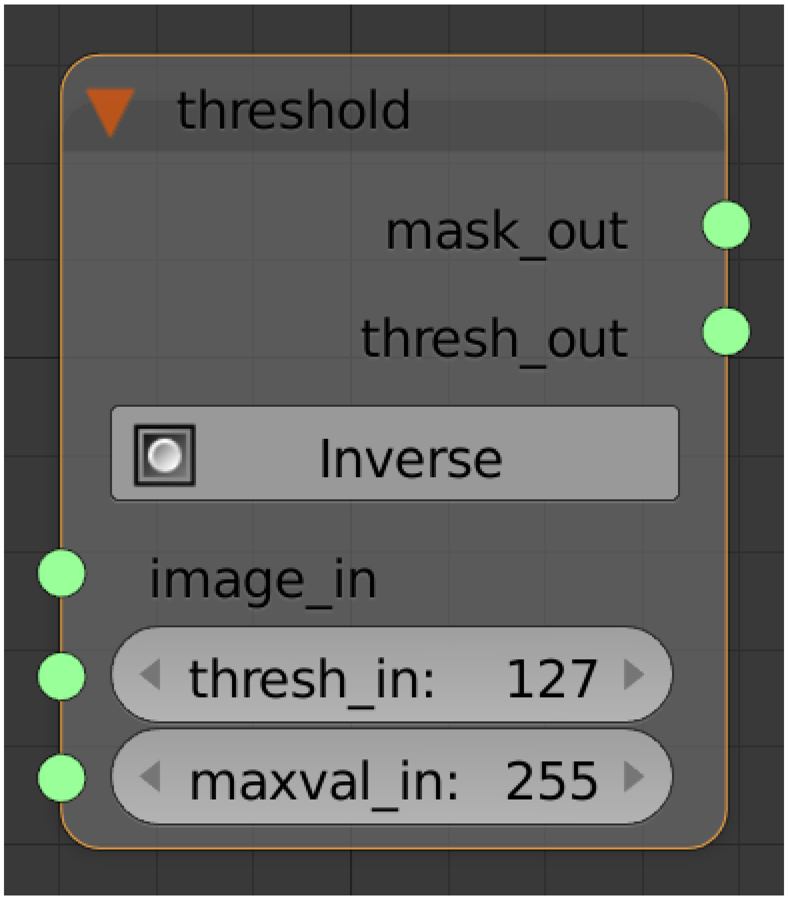
Outputs

- image_out – Output image

Locals

Examples

9.2.66 threshold



Functionality

Applies a fixed-level threshold to each array element.

Inputs

- image_in – Input array (single-channel, 8-bit or 32-bit floating point).
- maxval_in – Maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types
- thresh_in – Threshold value.
- type_in – Thresholding type (see the cv::ThresholdTypes).

Outputs

- mask_out – Output mask.
- thresh_out – Threshold value output.

Locals

- loc_invert – Invert output mask.

Examples

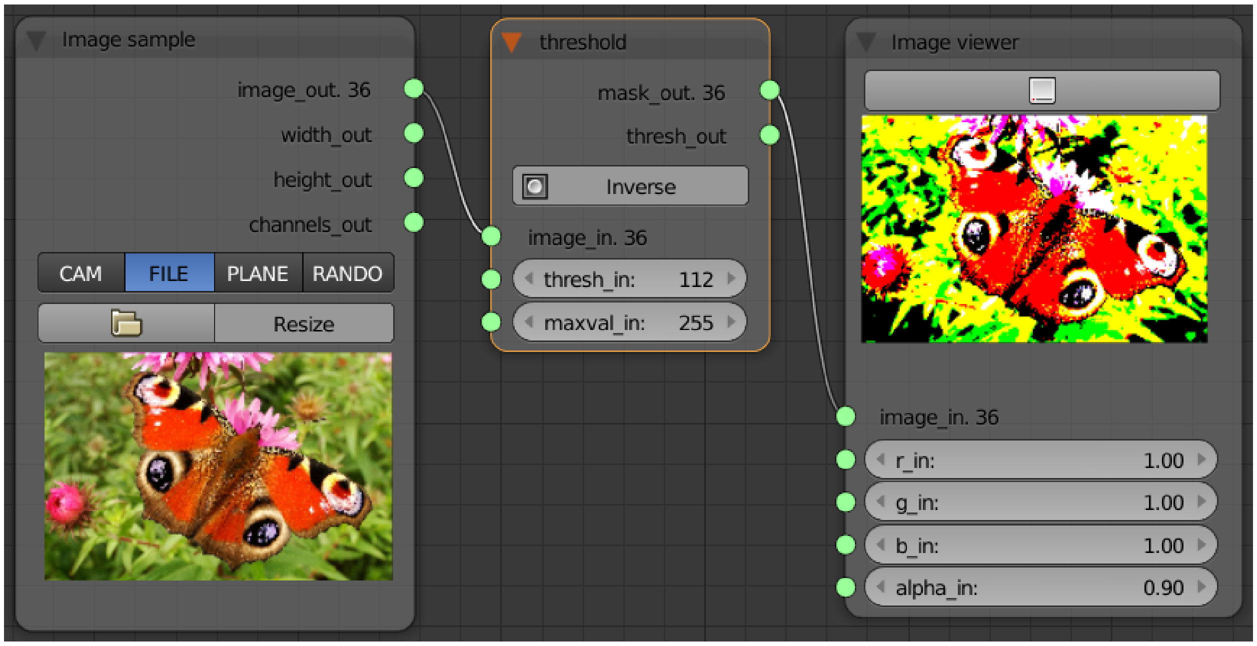


Image sample

image_out. 36

width_out

height_out

channels_out

CAM FILE PLANE RANDO

Resize

threshold

mask_out. 36

thresh_out

Inverse

image_in. 36

thresh_in: 112

maxval_in: 255

Image viewer

image_in. 36

r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90

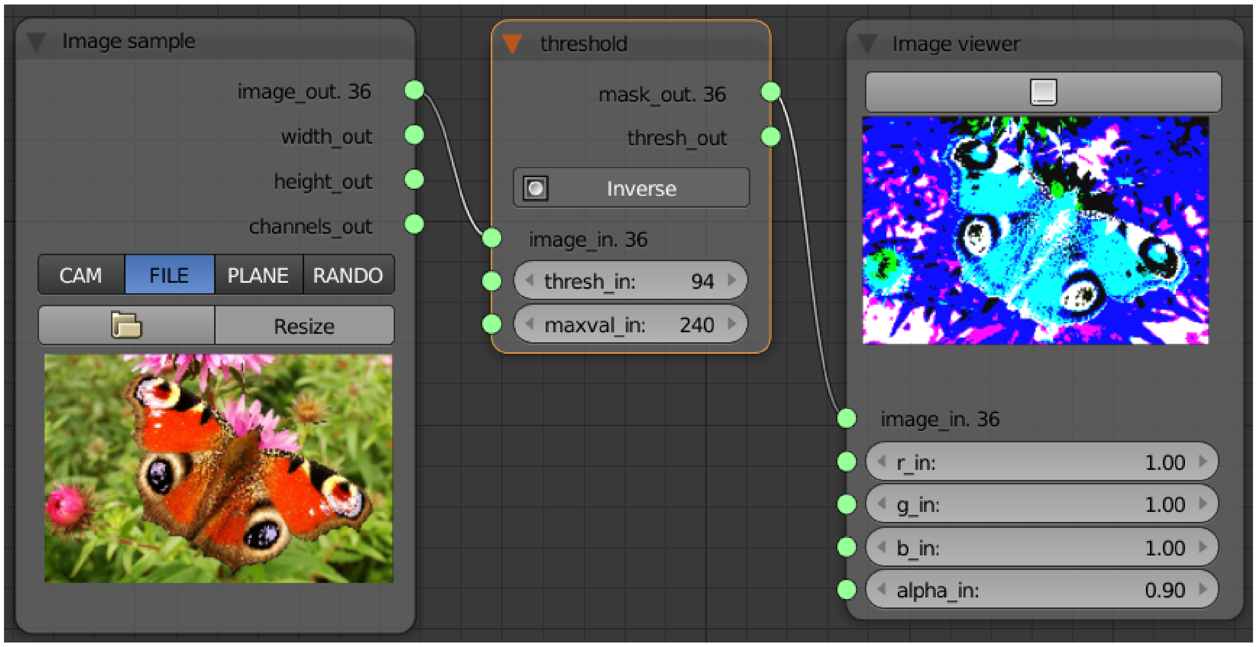


Image sample

image_out. 36

width_out

height_out

channels_out

CAM FILE PLANE RANDO

Resize

threshold

mask_out. 36

thresh_out

Inverse

image_in. 36

thresh_in: 94

maxval_in: 240

Image viewer

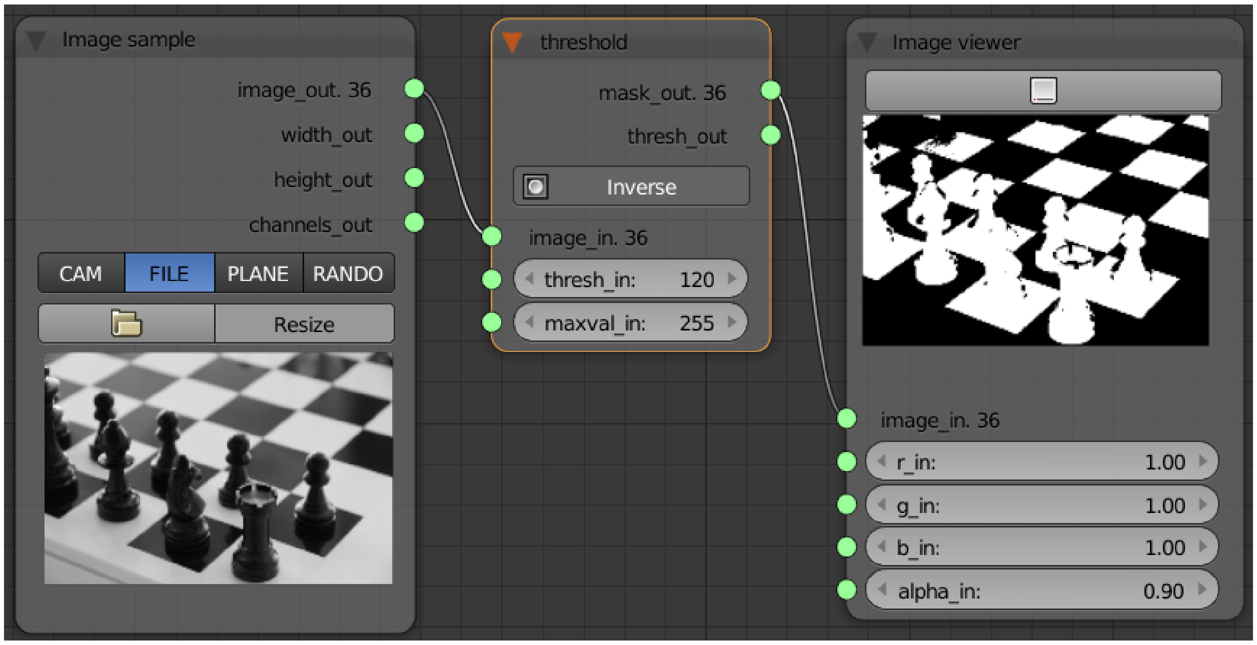
image_in. 36

r_in: 1.00

g_in: 1.00

b_in: 1.00

alpha_in: 0.90



9.2.67 undistort

Functionality

Transforms an image to compensate for lens distortion.

Inputs

- cameraMatrix_in – Input camera matrix
- distCoeffs_in – Input vector of distortion coefficients (k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]) of 4, 5, or 8 elements.
- image_in – Input (distorted) image.
- newCameraMatrix_in – Camera matrix of the distorted image. By default, it is the same as cameraMatrix but you may additionally scale and shift the result by using a different matrix.

Outputs

- image_out – Output (corrected) image.

Locals

Examples

9.2.68 undistortPoints

Functionality

Computes the ideal point coordinates from the observed point coordinates.

Inputs

- P_in – New camera matrix (3x3) or new projection matrix (3x4). P1 or P2 computed by stereoRectify() can be passed here. If the matrix is empty, the identity new camera matrix is used.
- R_in – Rectification transformation in the object space (3x3 matrix). R1 or R2 computed by stereoRectify() can be passed here. If the matrix is empty, the identity transformation is used.
- cameraMatrix_in – Camera matrix
- distCoeffs_in – Input vector of distortion coefficients (k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]) of 4, 5, or 8 elements.
- src_in – Observed point coordinates, 1xN or Nx1 2-channel (CV_32FC2 or CV_64FC2).

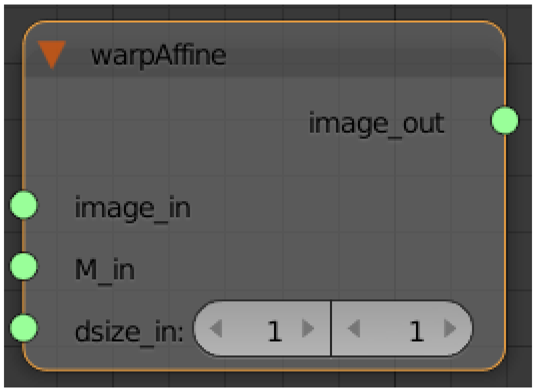
Outputs

- dst_out – Output ideal point coordinates after undistortion and reverse perspective transformation. If matrix P is identity or omitted, dst will contain normalized point coordinates.

Locals

Examples

9.2.69 warpAffine



Functionality

Applies an affine transformation to an image.

Inputs

- M_in – Transformation matrix.
- borderMode_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes
- borderValue_in – Border mode used to extrapolate pixels outside of the image, see cv::BorderTypes
- dsize_in – Size of the output image.
- flags_in – INTER_LINEAR, INTER_NEAREST, WARP_INVERSE_MAP

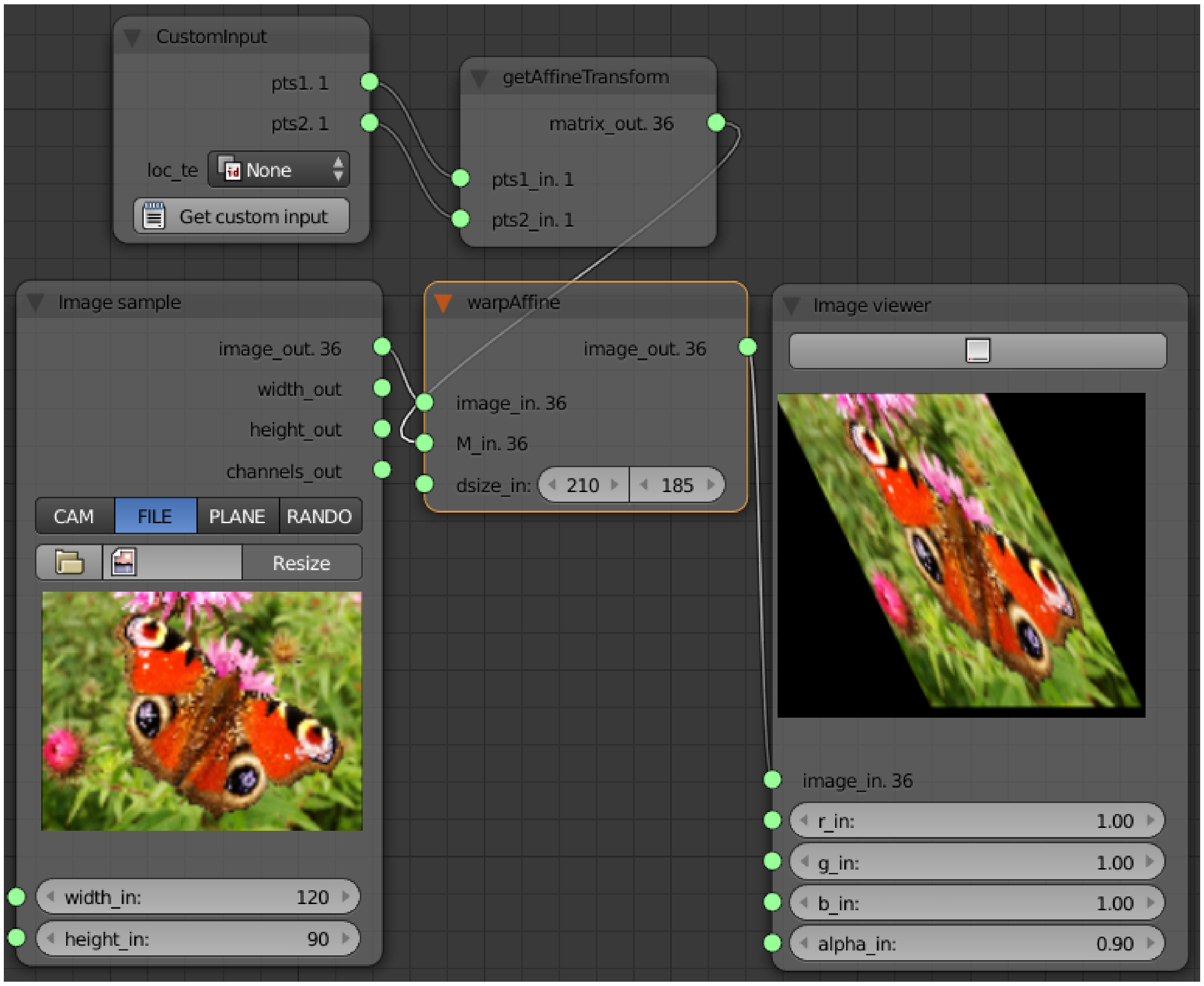
- image_in – Input image.

Outputs

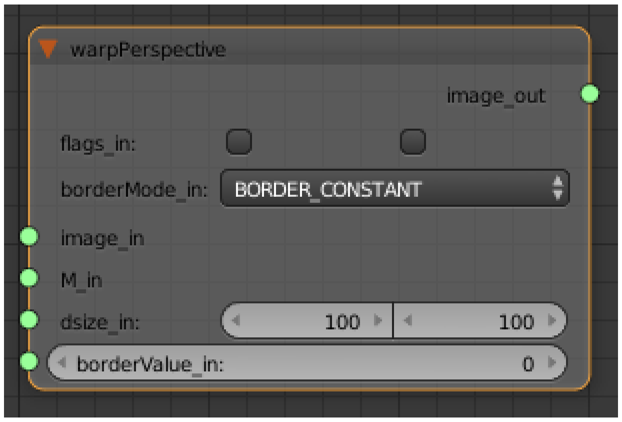
- image_out – Output image.

Locals

Examples



9.2.70 warpPerspective



Functionality

Applies a perspective transformation to an image.

Inputs

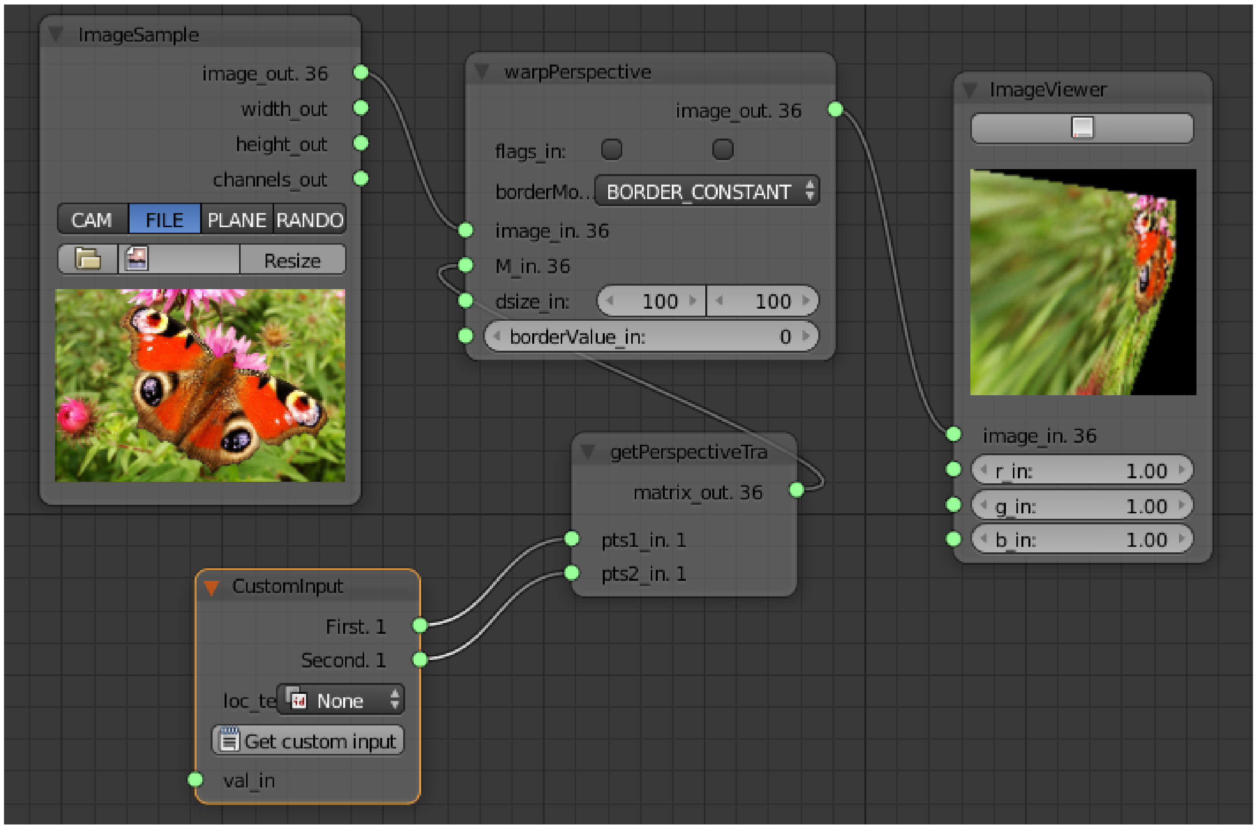
- M_in – Transformation matrix.
- borderMode_in – Pixel extrapolation method (BORDER_CONSTANT or BORDER_REPLICATE).
- borderValue_in – Value used in case of a constant border; by default, it equals 0.
- dsize_in – Size of the output image.
- flags_in – INTER_LINEAR, WARP_FILL_OUTLIERS
- image_in – Image input.

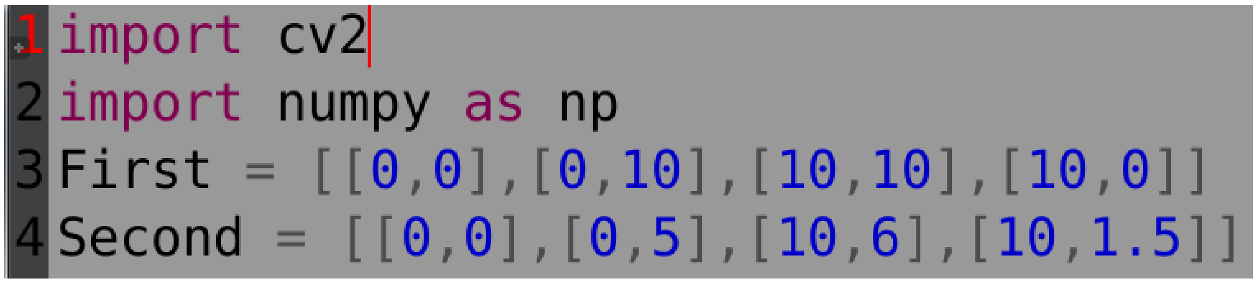
Outputs

- image_out – Image output.

Locals

Examples





```python
import cv2
import numpy as np
First = [[0,0],[0,10],[10,10],[10,0]]
Second = [[0,0],[0,5],[10,6],[10,1.5]]
```

9.3 laboratory

9.3.1 ROI

9.4 objdetect

9.4.1 CascadeClassifier

9.5 photo

9.5.1 inpaint

Functionality

doc

Inputs

- flags_in – INPAINT_NS, INPAINT_TELEA
- image_in – desc
- inpaintMask_in – Inpainting mask, 8-bit 1-channel image. Non-zero pixels indicate the area that needs to be inpainted.
- inpaintRadius_in – Radius of a circular neighborhood of each point inpainted that is considered by the algorithm.

Outputs

- image_out – desc

Locals

Examples

9.6 video

9.6.1 createBackgroundSubtractorMOG2

CHAPTER 10

Indices and tables

- genindex
- modindex
- search