PyFlowOpenCv

Release 0.0.1

Jul 12, 2020

Contents

CHAPTER 1

PyFlowOpenCv

PyFlowOpenCv is a easy to use rapid prototyping GUI tool for OpenCV. **PyFlowOpenCV** enable you learn Computer vision without writing a single line of code, which is great for rapid prototyping and learning. Plenty of OpenCV functions are available as building blocks in PyFlowOpenCv that can be combined in a graphical user interface with just a few mouse clicks. A quick demo on how **PyFlowOpenCv** works for a face detection.

PyFlow is a general-purpose Visual Dataflow Programming library. Nodes represent algorithms with certain inputs and outputs. Connections transfer data from the output (source) of the first node to the input (sink) of the second one. **PyFlowOpencv** is a visual scripting extension for PyFlow for OpenCV.

1.1 Goal

Learning OpenCV is quite challenging for most of the beginners. PyFlowOpenCv make the learning curve of OpenCv much smoother. You do not need to write any code, just drag and drop the diagram.

OpenCV comes with GUI tools like Highui and OpenCVGUI, but they are far from user friendly. You still need to write a lot of code to use them. With **PyFlowOpenCV**, user can focus on build the computer vision pipeline and fine tune the parameters, instead of writing boilerplate source code.

CHAPTER 2

Installation

PyFlowOpenCv is NOT a standalone software, it is an extension package of PyFlow. PyFlow has to be installed first. You can refer to PyFlow to install PyFlow.

The easy way to install PyFlow is:

```
pip install git+https://github.com/wonderworks-software/PyFlow.git@master
```

After PyFlow installed through pip or setup.py. Clone or download PyFlowOpenCV repository to a local folder:

```
git clone https://github.com/wonderworks-software/PyFlowOpenCv
```

Go to the source code folder and install requirements for your use case:

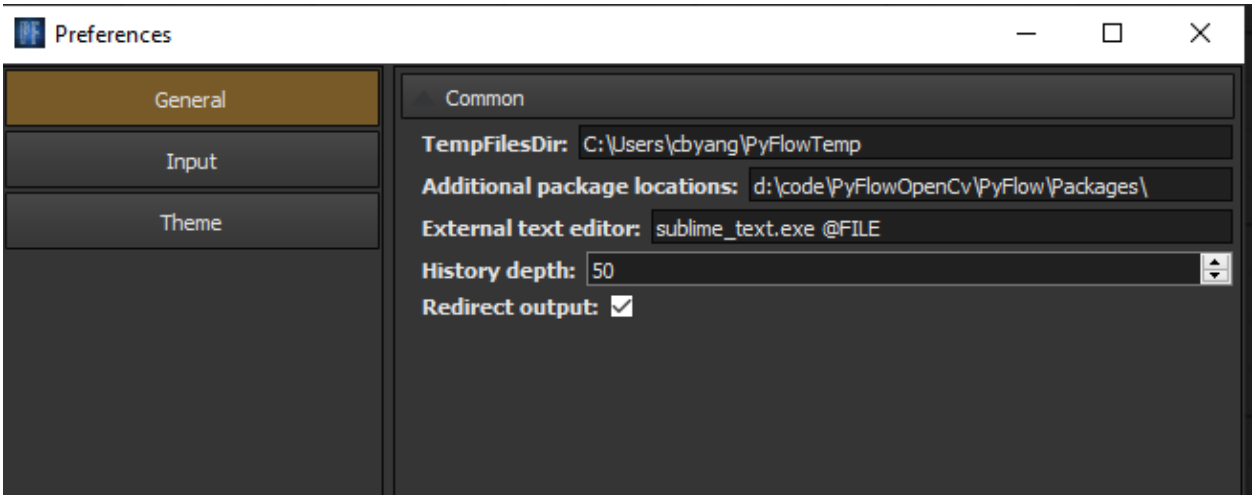
```
cd PyFlowOpenCv
pip install -r requirements.txt
```

Option: if you need to use OCR module, please refer to this page to install the Tesseract

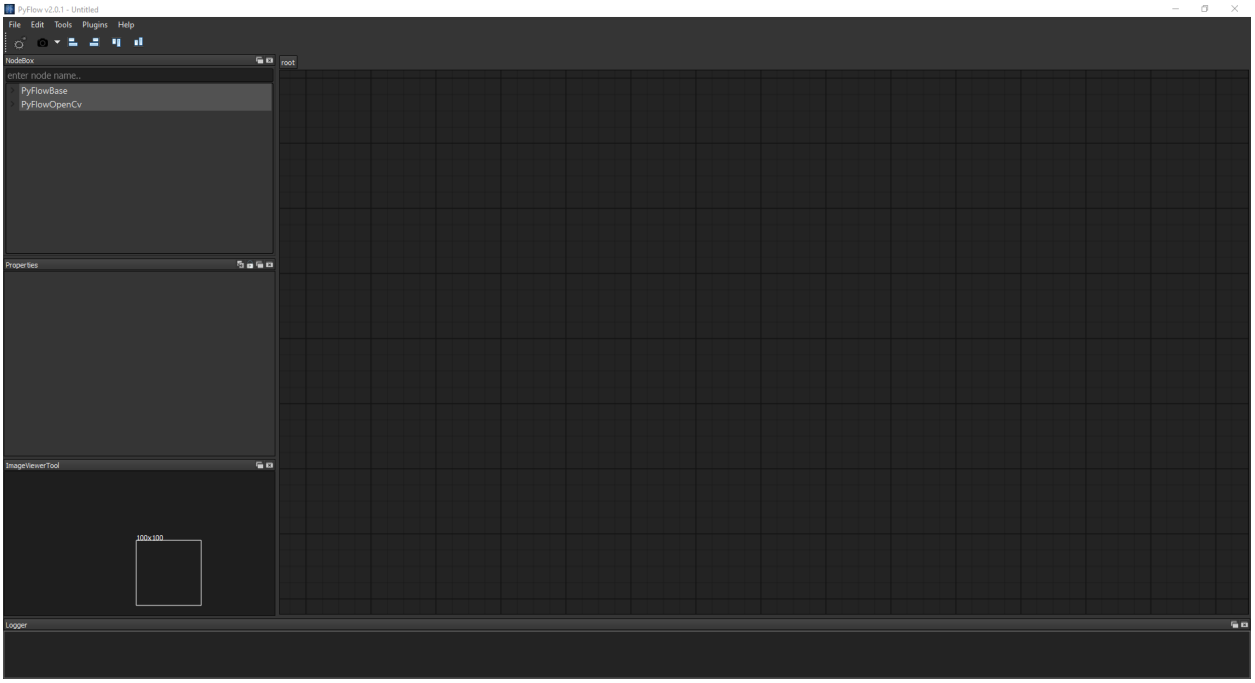
To run the program in standalone mode, you can invoke pyflow.exe on windows or pyflow on unix OS. Program is located inside PATH/TO/PYTHON/Scripts. You can also run pyflow.py in the root folder of PyFlow(not PyFlowOpenCV) project.

You can enable the PyFlowOpenCv package by one the of following ways.

- Copy the PyFlowOpenCv package to .PyFlow/Packages
- User can add location of package to env variable for PYTHONPATH
- Paths listed in PYFLOW_PACKAGES_PATHS env variable (; separated string)
- addition package on preferences dialog. Make sure you add path of PyFlow/Packages under PyFlowOpenCv project to the 'additional package location' edit.



If everything works out, you should able to see 'PyFlowOpenCv' in your NodeBox dialog of the GUI.



CHAPTER 3

Getting Started

We have documentation, and check out the examples in samples folder.

CHAPTER 4

Authors

Pedro Cabrera - Pedro Cabrera

Changbo Yang - Changbo Yang

See also the list of contributors who participated in this project.

CHAPTER 5

Discussion

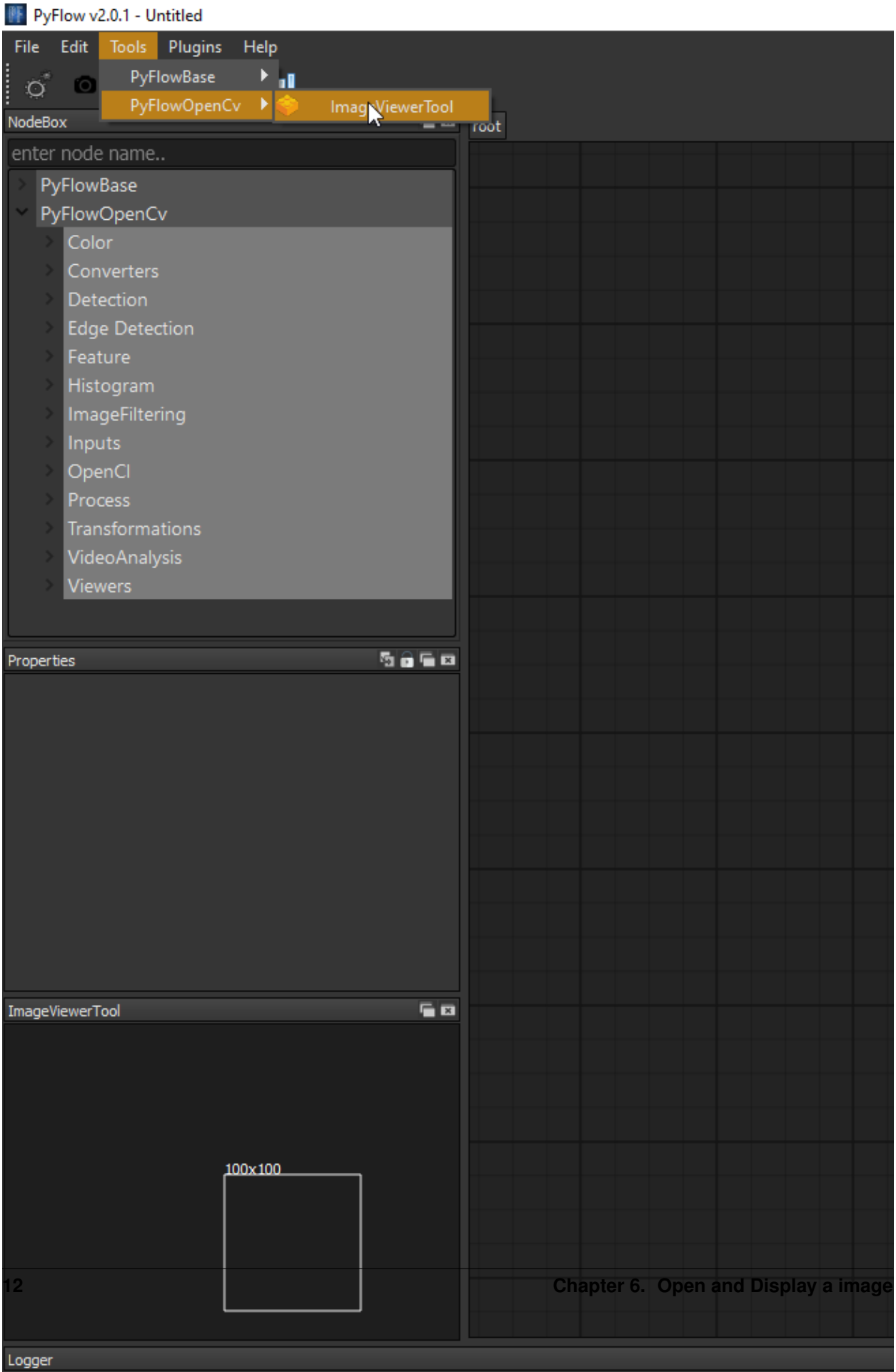
Join us to our discord channel and ask anything related to project! Please also let us know if you want more OpenCV features in PyFlowOpenCv.

CHAPTER 6

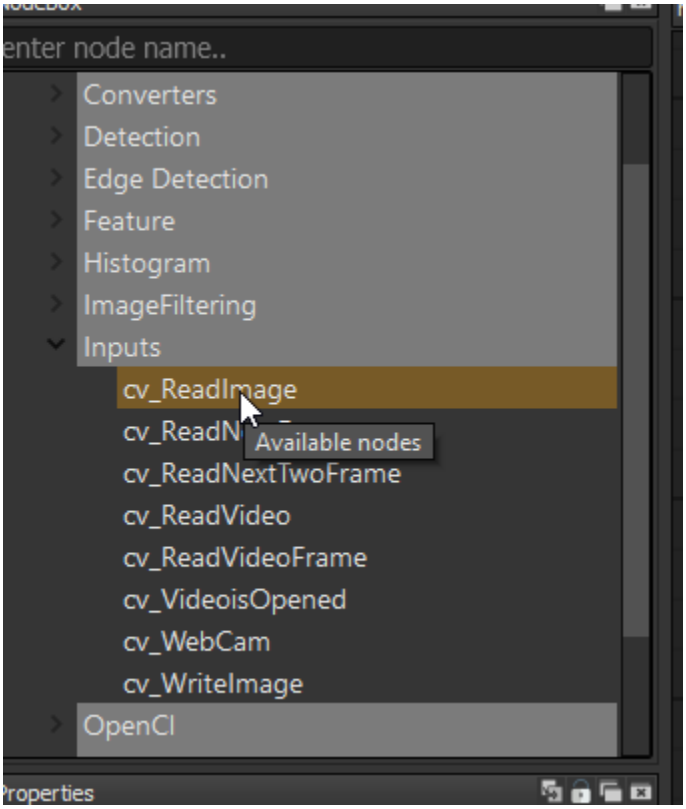
Open and Display a image

Now let's begin your first OpenCv diagram; loading a image and display it.

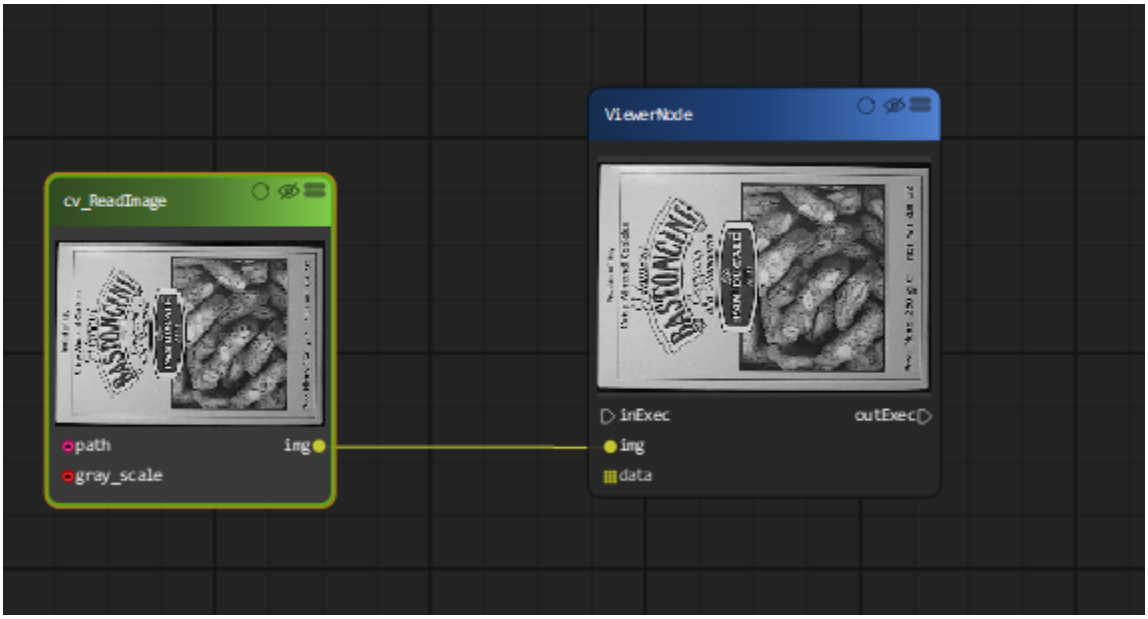
Before we start we start the first example, please go to Menu : Tools > PyFlowOpencv > imageveiwtool and click. This step will open a image viewer windows, so that you can inspect your image process result.


PyFlow v2.0.1 - Untitled
File
Edit
Tools
Plugins
Help
PyFlowBase
PyFlowOpenCv
ImageViewerTool
NodeBox
enter node name..
PyFlowBase
PyFlowOpenCv
Color
Converters
Detection
Edge Detection
Feature
Histogram
ImageFiltering
Inputs
OpenCl
Process
Transformations
VideoAnalysis
Viewers
Properties
ImageViewerTool
100x100
Logger

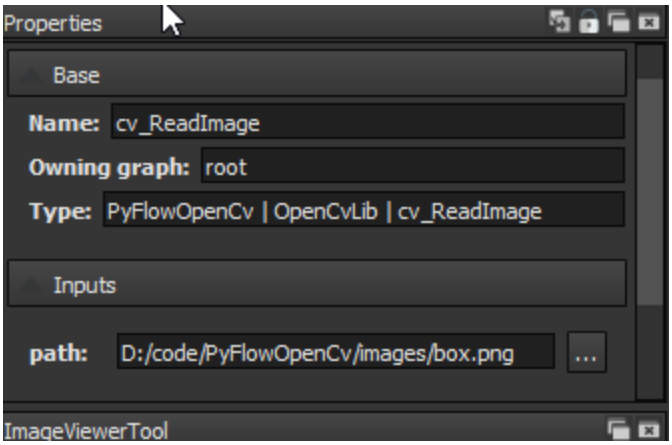
Now, go to NodeBox windows on GUI, expand PyFlowOpencv>Inputs , you will find Cv_ReadImage node, drag it to the right side diagram area.



Next, Go to PyFlowOpencv>viewers>ViewerNode and drag. Connect the endpoint 'image' of Cv_ReadImage to the endpoint 'image' of ViewerNode.



Then go to the property dialog of the gui, choose the image file you want to open.



Now, your first diagram is finished. To view the image, you may need to click the 'refresh' button on the ViewerNode.

The whole process should be worked as follows.

After you created your first diagram, you may want to save your diagram by click Menu 'File' > 'Save'

CHAPTER 7

Open and play a video

To open a video you need create a Cv_ReadVideo node similar to our first example. Then connect it to Cv_ReadNextFrame node to get a frame from the video. At last, you need the ViewNode to display you video.

However, this diagram can only display the first frame for you, we still need to create a loop to play every frame by adding a 'tick' node.

Tips: You can search the node by its name using the search box on the top of the NodeBox dialog.

CHAPTER 8

Image Filter

Continue our video example, we will add a image filter block to blur the video image.

You can even adjust the smoothing factor on realtime.

CHAPTER 9

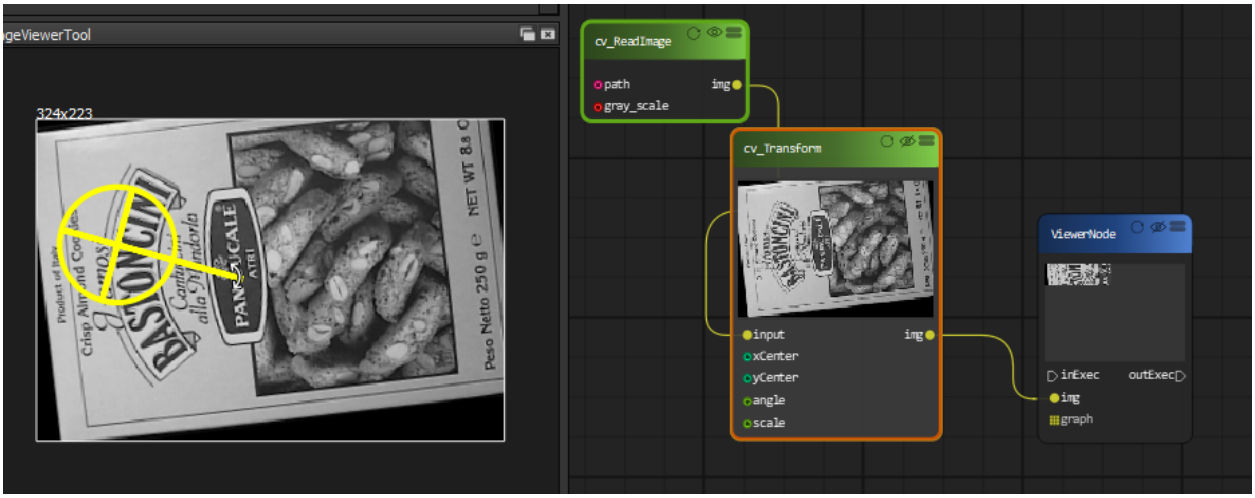
Image Threshold

See how image threshold works.

CHAPTER 10

Image Transform(Rotation and Translation)

Image rotation can be done with the CV_Transform block, which provide a joystick like tool to choose the rotation center and angle.

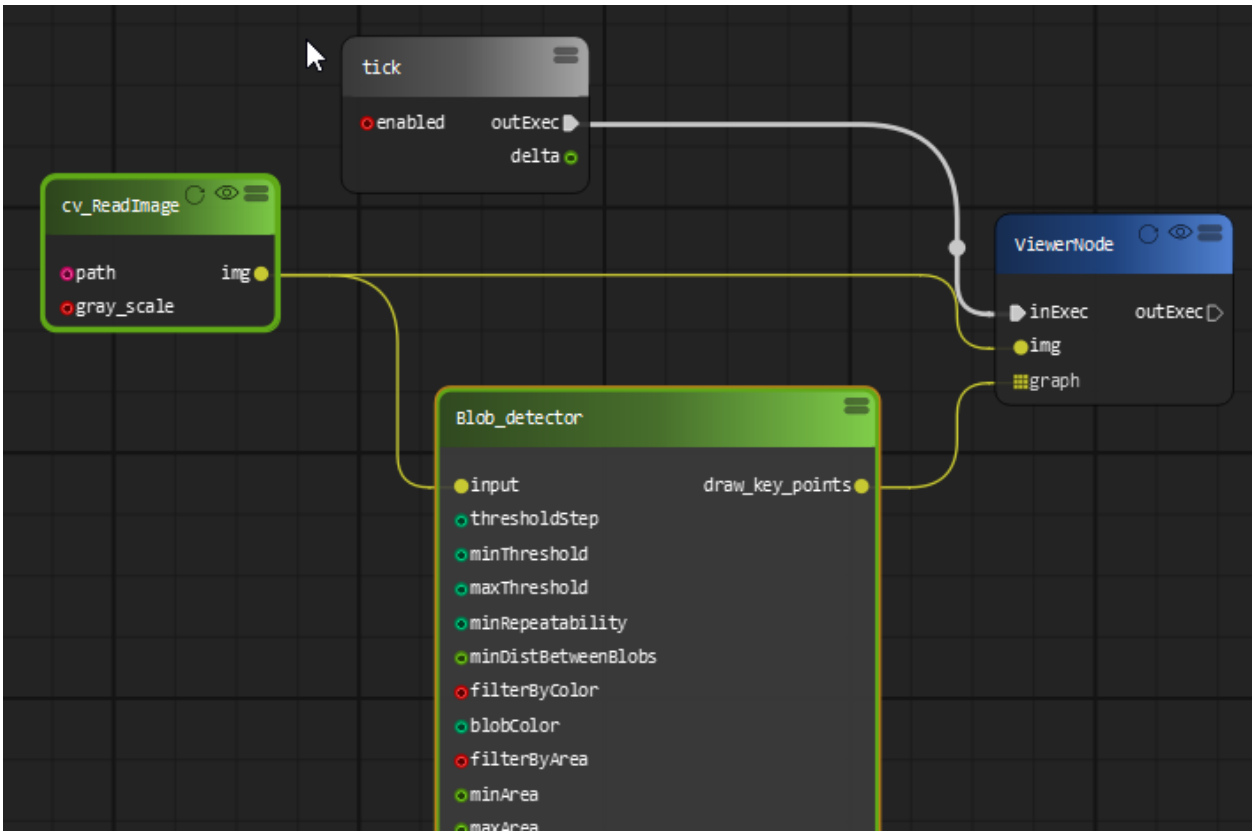


CHAPTER 11

Blob detection

In this example, we are going to create a project can detect blobs using OpenCV. A blob is a group of connected pixels in an image that share some common property.

We are going to use blob_detection node to detect the blob in the image.



The algorithm for extracting blobs from an image works in the folllowing steps:

- Convert the source image to binary images by applying thresholding with several thresholds from minThreshold (inclusive) to maxThreshold (exclusive) with distance thresholdStep between neighboring thresholds.
- Extract connected components from every binary image by findContours and calculate their centers.
- Group centers from several binary images by their coordinates. Close centers form one group that corresponds to one blob, which is controlled by the minDistBetweenBlobs parameter.
- From the groups, estimate final centers of blobs and their radiuses and return as locations and sizes of key points.

This node can also performs several filtrations of returned blobs. You should set filterBy* to true/false to turn on/off corresponding filtration. Available filtrations:

- By color. This filter compares the intensity of a binary image at the center of a blob to blobColor. If they differ, the blob is filtered out. Use blobColor = 0 to extract dark blobs and blobColor = 255 to extract light blobs.
- By area. Extracted blobs have an area between minArea (inclusive) and maxArea (exclusive).
- By circularity. Extracted blobs have circularity ($4\pi \frac{Area}{perimeter \cdot perimeter}$) between minCircularity (inclusive) and maxCircularity (exclusive).
- By ratio of the minimum inertia to maximum inertia. Extracted blobs have this ratio between minInertiaRatio (inclusive) and maxInertiaRatio (exclusive).
- By convexity. Extracted blobs have convexity (area / area of blob convex hull) between minConvexity (inclusive) and maxConvexity (exclusive).

By changing the filteration parameters in property dialog. The result image will be changed in realtime.

CHAPTER 12

Feature extraction and matching

Welcome to a feature matching tutorial with PyFlowOpenCv. We start with the image that we're hoping to find, and then we can search for this image within another image. The beauty here is that the image does not need to be the same lighting, angle, rotation... etc. The features just need to match up.

To start, we need some sample images. Our "template," or image we're going to try to match:

Then our image to search for this template in:

12.1 Feature detection

OpenCV includes the following feature extraction algorithms:

- Haris corner detection
- Shi-Tomasi corner detection
- SIFT (Scale-Invariant Feature Transform)
- SURF (Speeded-Up Robust Features)
- FAST algorithm for corner detection
- ORB (Oriented FAST and Rotated Brief)

SIFT, SURF are patented and are not available free for commercial use. It requires opencv-contrib to be installed in order to use them

12.2 Feature matching

Feature matching between images in OpenCV can be done with Brute-Force matcher or FLANN based matcher.

12.2.1 Brute-Force (BF) Matcher

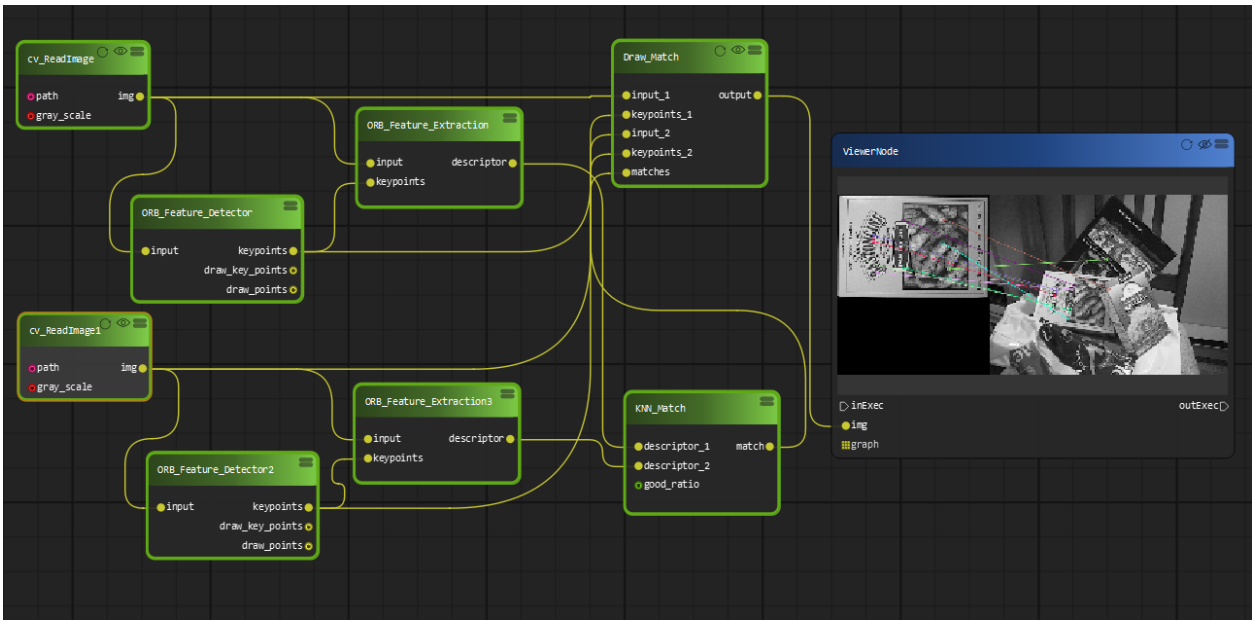
BF Matcher matches the descriptor of a feature from one image with all other features of another image and returns the match based on the distance. It is slow since it checks match with all the features

12.2.2 FLANN based matcher

Fast Library for Approximate Nearest Neighbors (FLANN) is optimised to find the matches with search even with large datasets hence its fast when compared to Brute-Force matcher. With ORB and FLANN matcher let us extract the tesla book cover from the second image and correct the rotation with respect to the first image

We are going to build the diagram in the following way:

- open the template image and target image
- Initialize the ORB detector and detect the keypoints in query image and scene.
- Compute the descriptors belonging to both the images.
- Match the keypoints using KNNMatcher.
- Show the matched images.




ViewerNode
inExec
img
graph
outExec

CHAPTER 13

Image Histogram

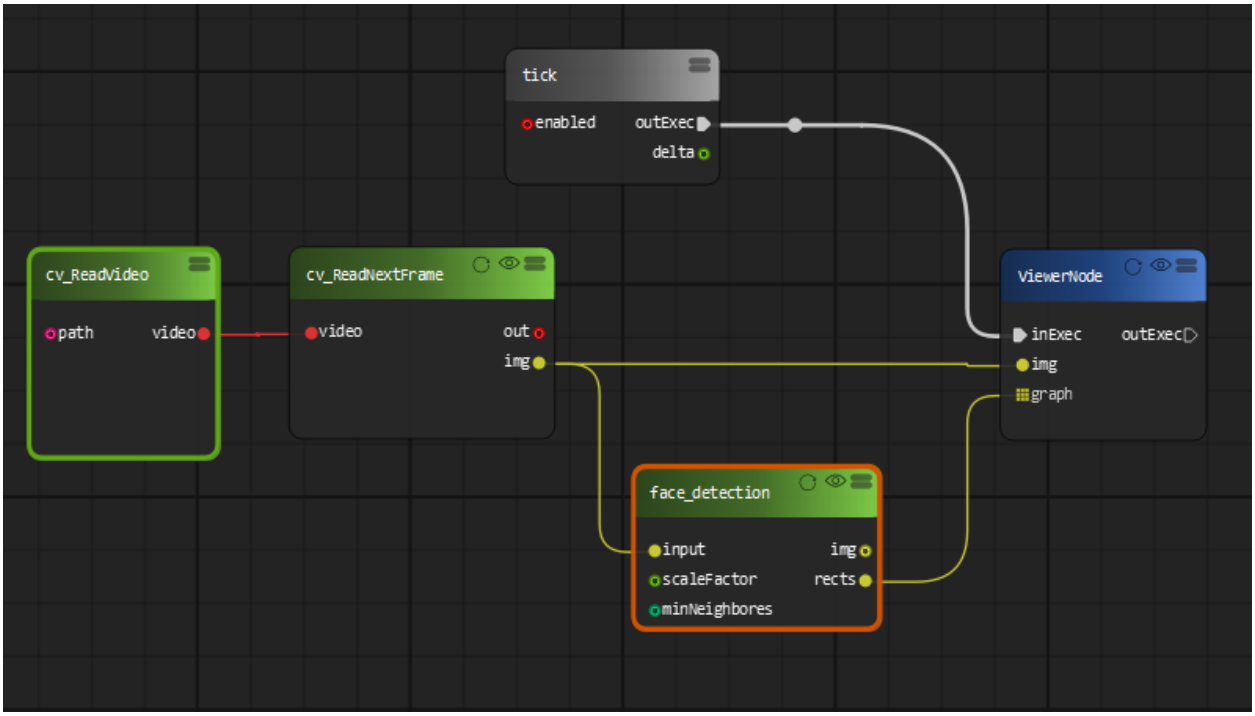
A histogram is a very important tool in Image processing. It is a graphical representation of the distribution of data. An image histogram gives a graphical representation of the distribution of pixel intensities in a digital image.

CHAPTER 14

Face detection by Haar Classifier

Face detection is a technique that identifies or locates human faces in digital images. Face detection in OpenCV is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive(face) or negative(not a face). A classifier needs to be trained on thousands of images with and without faces. OpenCV come with a pre-trained Haar feature based face detection classifiers, which can readily be used in a PyFlowOpenCv.

Now let's create a OpenCV data flow diagram can detection the face in a video or a WebCam.

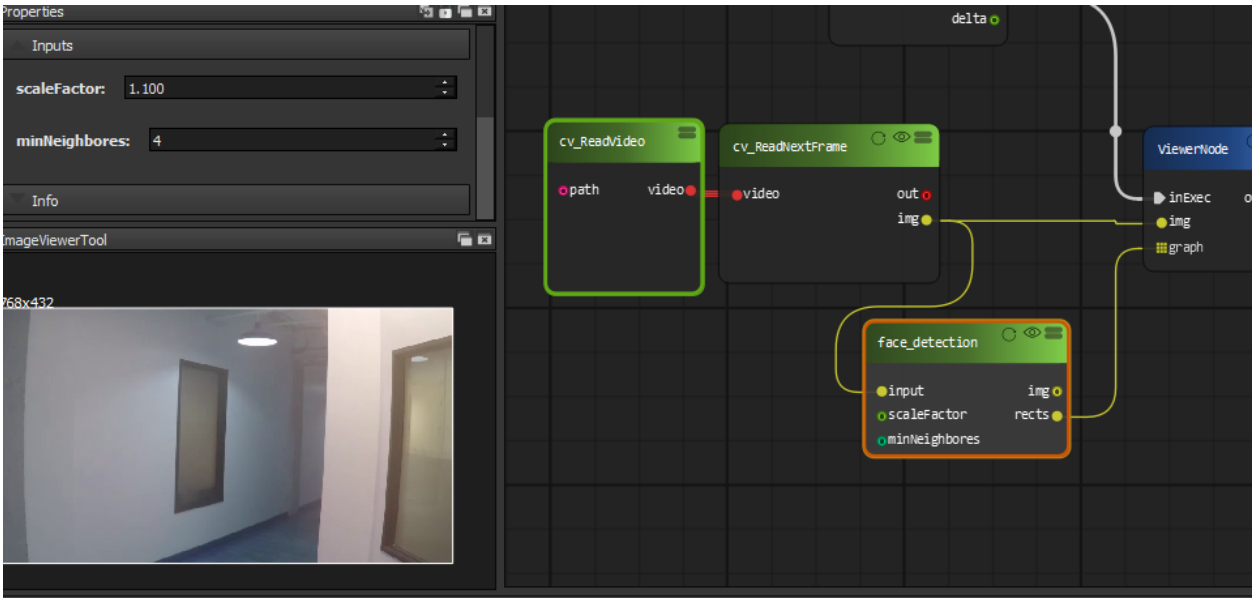


The result video on videoviewer node will be something like this:

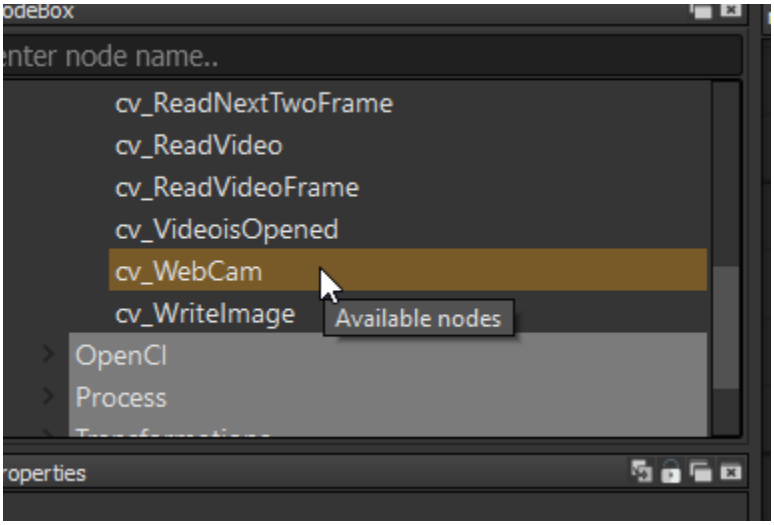
Under the hood, we are using the detectMultiscale module of the OpenCV haar feature classifier. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. This function has two important parameters which have to be tuned according to the data.

- scalefactor: In a group photo, there may be some faces which are near the camera than others. Naturally, such faces would appear more prominent than the ones behind. This factor compensates for that.
- minNeighbors: This parameter specifies the number of neighbors a rectangle should have to be called a face. You can read more about it here.

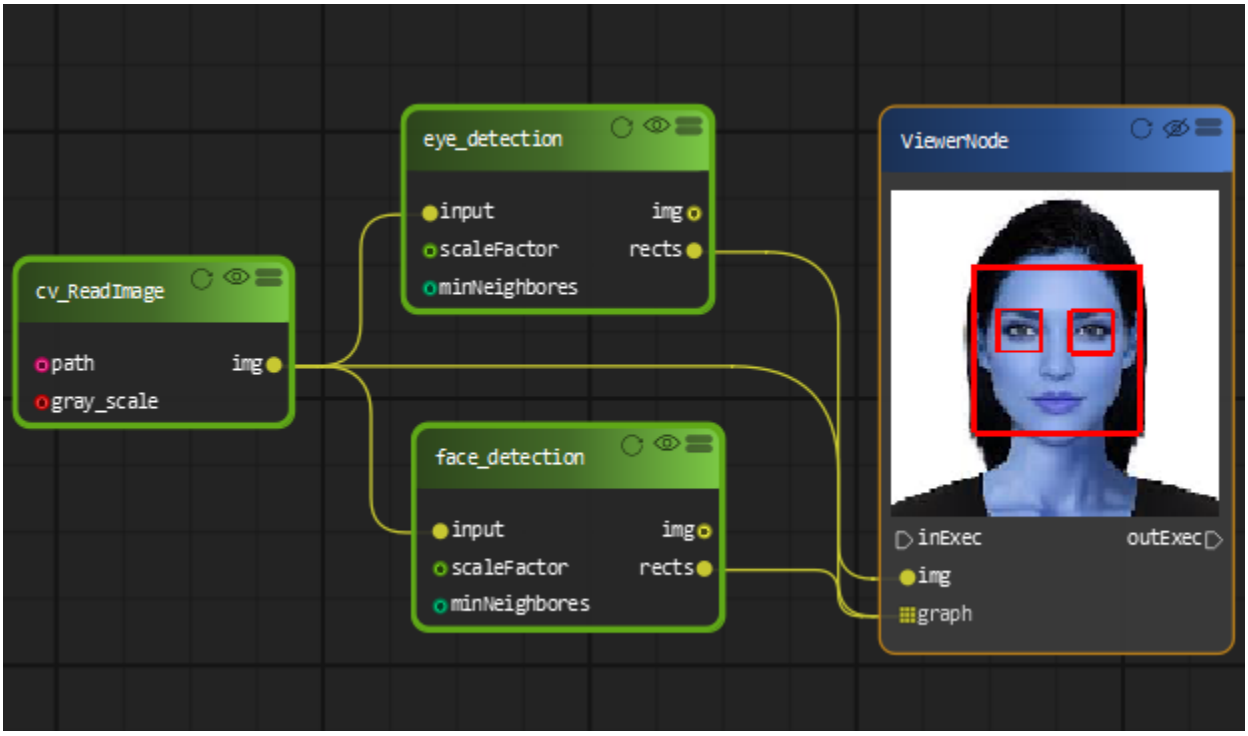
You can change the two parameters on the fly and see how the parameter changed the detection result.



If you want to detect face from the WebCam, just drag a WebCam node to the diagram instead of the ReadVideo node:



It is also possible to detect face and eye at the same time.

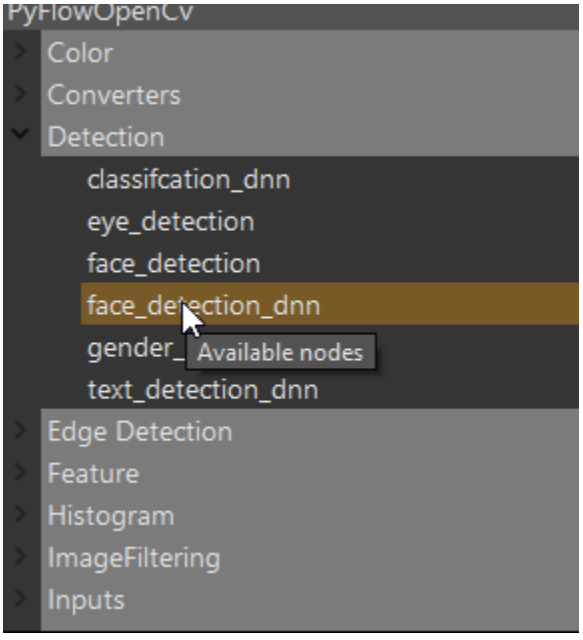

eye_detection
input
scaleFactor
minNeighbores
img
rects
ViewerNode
cv_ReadImage
path
gray_scale
img
face_detection
input
scaleFactor
minNeighbores
img
rects
inExec
outExec
img
graph

CHAPTER 15

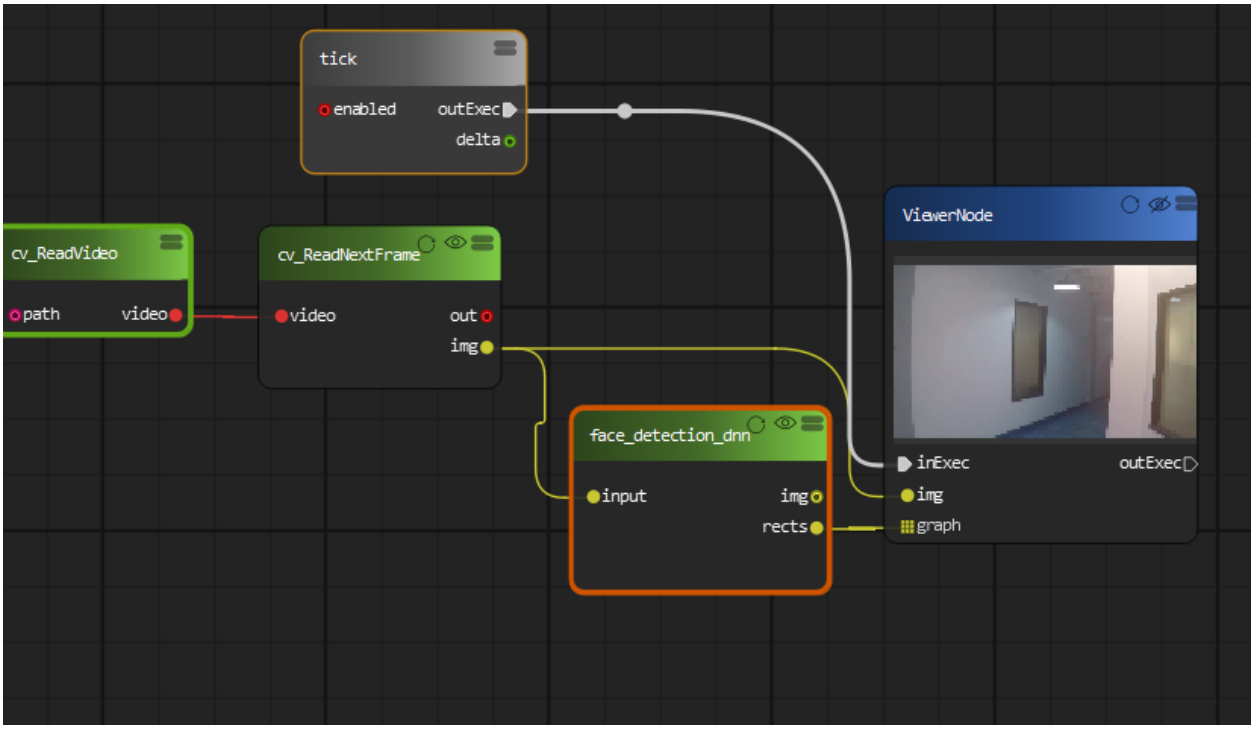
Face Detection by Convolutional Neural Network (CNN) detectors

Starting from version 3.3, OpenCV supports the Caffe, TensorFlow, and Torch/PyTorch frameworks. OpenCV can load pre-trained CNN model directly.

In this example, we are going to build a computer vision diagram can detect face using Deep Learning Neural Network. We are going to use a new node named 'face_detection_dnn'.



Similar to our previous face Detection example, we are going to create the following diagram:



The Detection result will be much stable than our previous example using Haar feature.

CHAPTER 16

Text Detection by deep learning and Recognition

16.1 EAST Detector for Text Detection

OpenCV's EAST(Efficient and Accurate Scene Text Detection) text detector is a deep learning model, based on a novel architecture and training pattern. It is capable of running at near real-time at 13 FPS on 720p images and obtains state-of-the-art text detection accuracy.

Link to paper

OpenCV's text detector implementation of EAST is quite robust, capable of localizing text even when it's blurred, reflective, or partially obscured.

This is an example of text detection on a webcam.

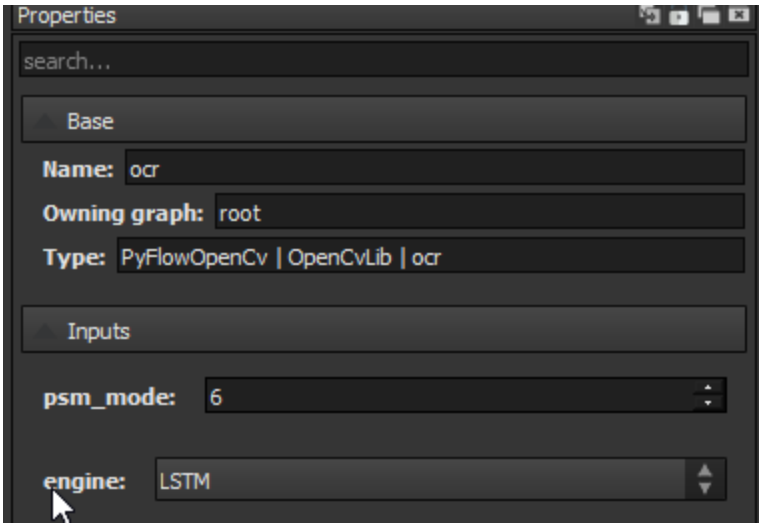
16.2 OCR by Tesseract

Tesseract is an open source text recognition (OCR) Engine, available under the Apache 2.0 license. Installing tesseract on Windows is easy with the precompiled binaries found here. Do not forget to edit "path" environment variable and add tesseract path. For Linux or Mac installation it is installed with few commands.

There is also one more important argument, OCR engine mode (oem). Tesseract 4 has two OCR engines — Legacy Tesseract engine and LSTM engine. There are four modes of operation chosen using the –oem option:

- Legacy engine only.
- Neural nets LSTM engine only.
- Legacy + LSTM engines.
- Default, based on what is available.

With PyFlowOpenCv, you can choose the the engine in the property dialog:

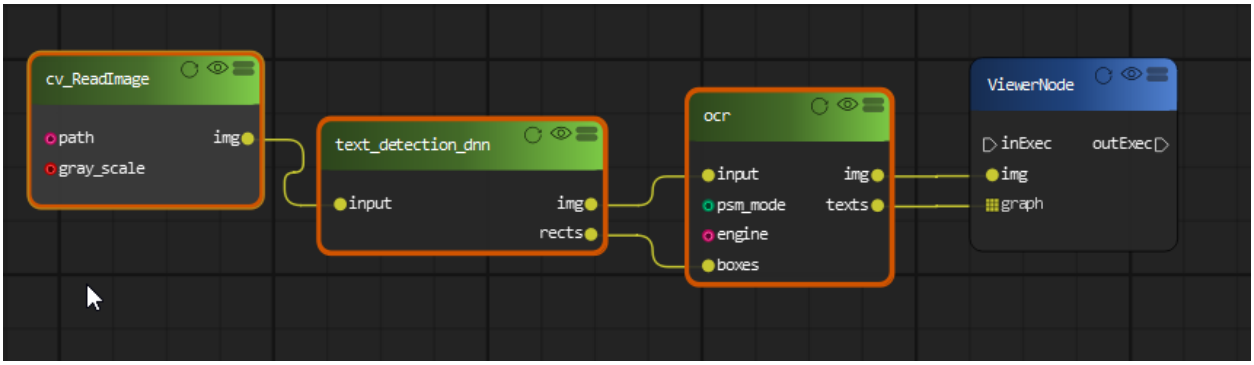


Page Segmentation Mode (–psm). That affects how Tesseract splits image in lines of text and words. Pick the one which works best for you:

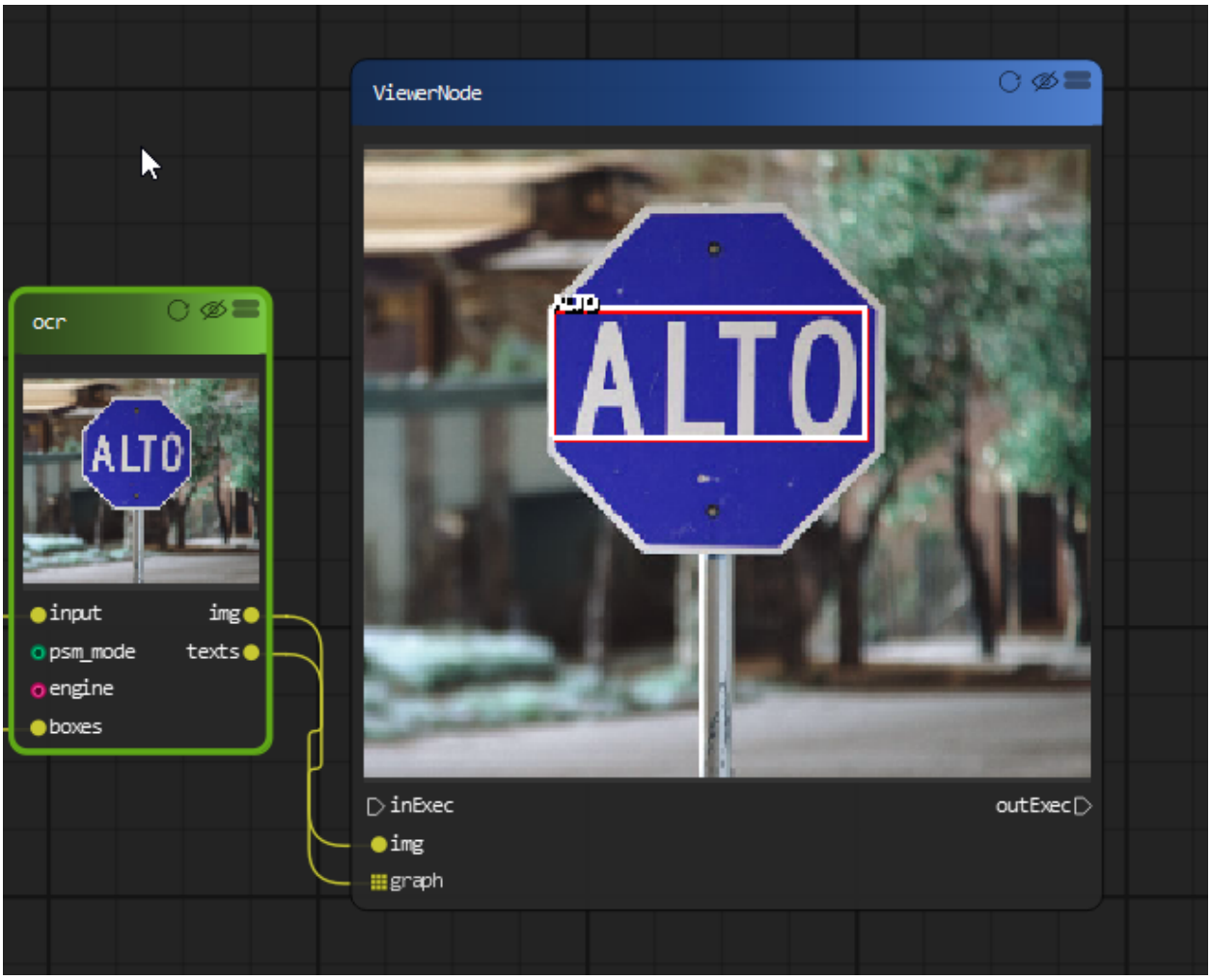
0. Orientation and script detection (OSD) only.
1. Automatic page segmentation with OSD.
2. Automatic page segmentation, but no OSD, or OCR.
3. Fully automatic page segmentation, but no OSD. (Default)
4. Assume a single column of text of variable sizes.
5. Assume a single uniform block of vertically aligned text.
6. Assume a single uniform block of text.
7. Treat the image as a single text line.
8. Treat the image as a single word.
9. Treat the image as a single word in a circle.
10. Treat the image as a single character.
11. Sparse text. Find as much text as possible in no particular order.
12. Sparse text with OSD.
13. Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific.

You can also choose the page segmentation mode in property dialog, but since we feed the text region to OCR engine, mode 6 should work best in most cases.

To recognize the text in the image, we will use the EAST text detection node first, and feed the detected region to OCR engine.



And you should able to get all the text in the image.

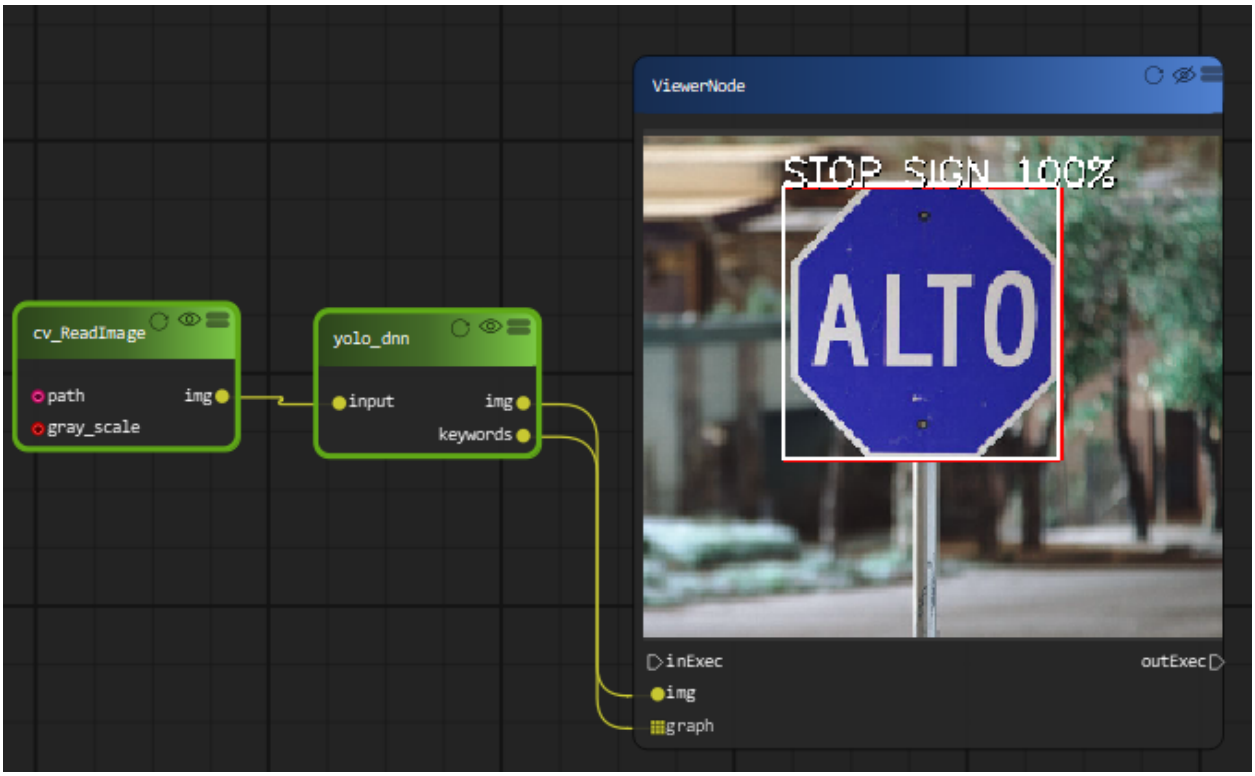


CHAPTER 17

YOLO object detection with OpenCV

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev.

Due to the large size of the yolo model file, we only includes a tiny yolo model with the package.



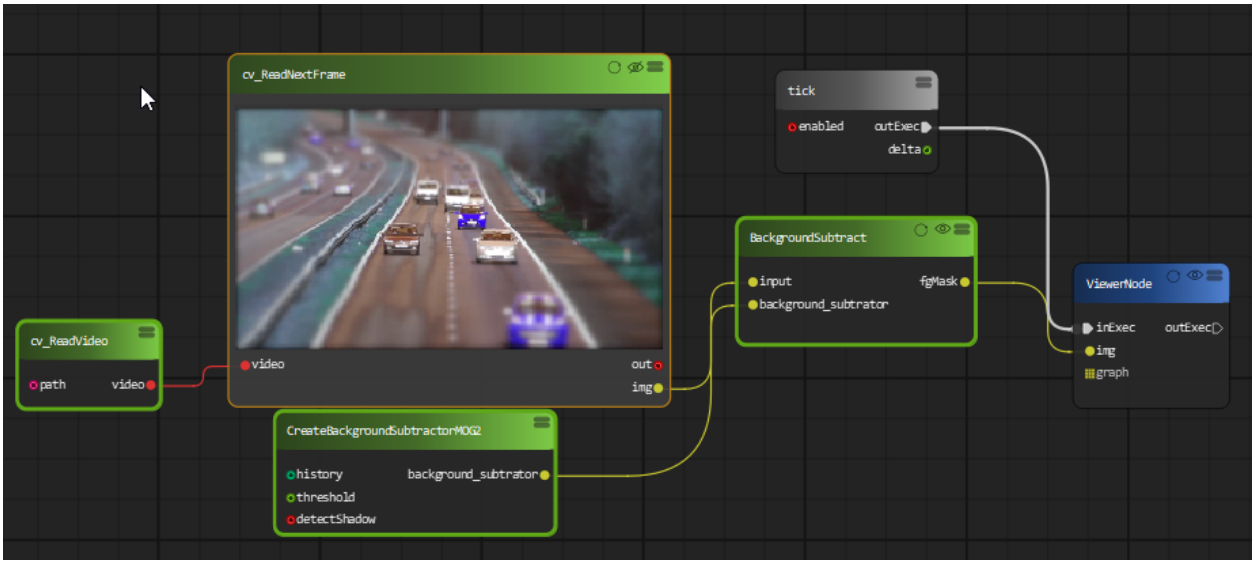
You can also use YOLO detector on realtime webcam.

CHAPTER 18

Background subtraction

Background subtraction (BS) is a common and widely used technique for generating a foreground mask In PyFlowOpenCv we implemented two BS model by using OpenCV MOG2 and KNN.

First we need to create a CreateBackgroundSubtractorMOG2 or CreateBackgroundSubtractorKNN block and connect to a BackgroundSubtract block.



You can also change the parameter of background subtractor model on realtime.

CHAPTER 19

Indices and tables

- genindex
- modindex
- search