University of Burgundy

Masters in Computer Vision and Robotics

Visual Perception Module

Report for Computer Vision Toolbox

in

PYTHON

By:

Vamshi Kodipaka

Supervisor:

Dr. Abd El Rahman SHABAYEK

CONTENTS

INTRODUCTION

a. What is Computer Vision?

Computer Vision is the science that aims to give a similar, if not better, capability to a machine or computer by understanding the digital images or videos. Computer Vision include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information. This image understanding can be obtained from image data using models built by using geometry, physics, statistics, and learning theory. From engineering perspective, it seeks to automated tasks that the human visual system can do.

Computer Vision uses artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or for example multi-dimensional data from a medical scanner. Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

b. What is a Toolbox? What is its necessity?

A Software Toolbox is a group of Programmed tools designed and put together into an application enabling easy access to the user through a GUI. And for building Computer Vision Toolbox, in pre-processing stage using image processing tools are preferred. So, we aim at building a toolbox where we can do all sort of image processing and computer vision processes in one Graphical Use Interface. This will make the Computer Vision Researcher's life easier!

PYTHON is an effective tool which I choose here to build the CV Toolbox. So, I have explored OpenCV online source, Qt Designer, Python, PyQt5, Ubuntu to build my Computer Vision Toolbox. I made use of the Qt Designer's Graphical User Interface (GUI) facility to build a good user interface for our toolbox.

ABOUT OPENCV

OpenCV (*Open Source Computer Vision*)[1] is a library of programming functions mainly aimed at real-time computer vision. It is an open source computer vision and machine learning software library. OpenCV was built to provide a common platform for computer vision applications and to accelerate the use of machine perception in the commercial use.

The library has optimized algorithms those include comprehensive set of both classic and state-of-the-art computer vision and machine learning techniques. These algorithms can be used to detect

and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, recognize scenery and step up with markers to attain Augmented Reality.

It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

SOFTWARE USED & INSTALLATIONS

Firstly, install all required software and libraries.

Software you should install in your PC:

1. OpenCV 3.4.2.17
2. Qt Designer for GUI
3. Python (Ubuntu Terminal)
4. Atom IDE (For Python Script Editing)

Secondly, go to the Ubuntu terminal and create a Virtual Environment

Thirdly, activate Virtual Environment and install all the libraries.

Here are the list of commands:

➔ Create Virtual Environment: sudo apt install –y python3 –venv
➔ Activate V-Environment: source toolbox/bin/activate
➔ To enter Python Environment: python3.6 –m venv toolbox
➔ To quit Python Environment: quit( )

➔ Go to working location: use cd <folder name/folder name/…./ folder name>
➔ Here in this step: Create GUI add all the components of your requirement

➔ Use this to convert your .ui to .py file:
Pyuic5 –x <your filename.ui> -o <your filename.py>
For Example: Pyuic5 –x ipcv111.ui -o ipcv222.py

➔ Edit the script: Import required Libraries and write GUI component functionalities
➔ To Compile and Run: python ipcv222.py
➔ To deactivate Virtual Environment use command: deactivate

Finally, all the outputs for the Toolbox can be displayed on GUI as per user action choice.

BUILDING TOOLBOX

Creating an efficient GUI is most primary task for designing a Toolbbox. As Qt Designer is a free and easy to build the GUI of one's own, I have chosen to build my project using Qt GUI established connection with Python Platform (on Ubuntu).

I realized that this is very effective way to work on designing a Toolbox related to OpenCV. This the simpler way, I figured out for basic level programming learners who don't have expertise in C++/Python. I am also a basic level programming learner, this proved me all the source to build the CV Toolbox Project.

PyQt is one of the two most popular Python bindings for the Qt cross-platform GUI/SQL/XML C++ framework (the other one is PySide). Qt-devel is the development package of Qt framework which contains all the header files required to access libraries. Qt Designer is a tool for creating and binding GUIswith QtWidgets.

To install(Ubuntu):

```
apt-get install python-qt5 pyqt-dev-tools qt5-designer
```

Now, according to logical classification of tasks, I used tabs in Qt GUI to group the tasks.

As Opencv has 'mat' data type, it's easy to think in terms of processing which is very similar data type in Matlab, where we are most used to!

2.1 General Architecture of toolbox

In the GUI, we have tabs to select the particular group of logical tasks and in each tab we have different features to select. The input and output data is shown in two panels. There are push buttons to load, process and save data as shown in Fig.1

For any task to perform, the steps are pretty simple.

1. Load the image using 'Load image' push button.

2. Select the tab and Press the Pushbutton to select the action to be performed on the input image.

3. Processed data will be displayed under output window.

6. Click 'save image' button to save the current output.

In Qt Designer, we have to create a GUI >> Go and click Blank GUI >> New workspace opens.

Now, one's task is to drag and drop all the GUI Components required and should be arranged in proper logical order to help in execution according to selection.

Arrange them in proper tabs and Group the icons using Group Items. All the background code for class and instances are structured into a .ui extension file once this GUI is saved. For a better understanding the Modular diagram of the toolbox is given in Fig. 6.

I have implemented modular programming approach, where in all tasks are divided according to the modules and solved.

Since I felt that in learning Python GUI is time taking to learn besides OpenCV, I mostly used Pushbutton components in my GUI and connected modules/functionalities to the Pushbuttons. *(I am a beginner in Python!)*

STEPS FOR BUILDING GUI:

Step 1: Open Qt Designer and Choose the type of GUI you want to build.

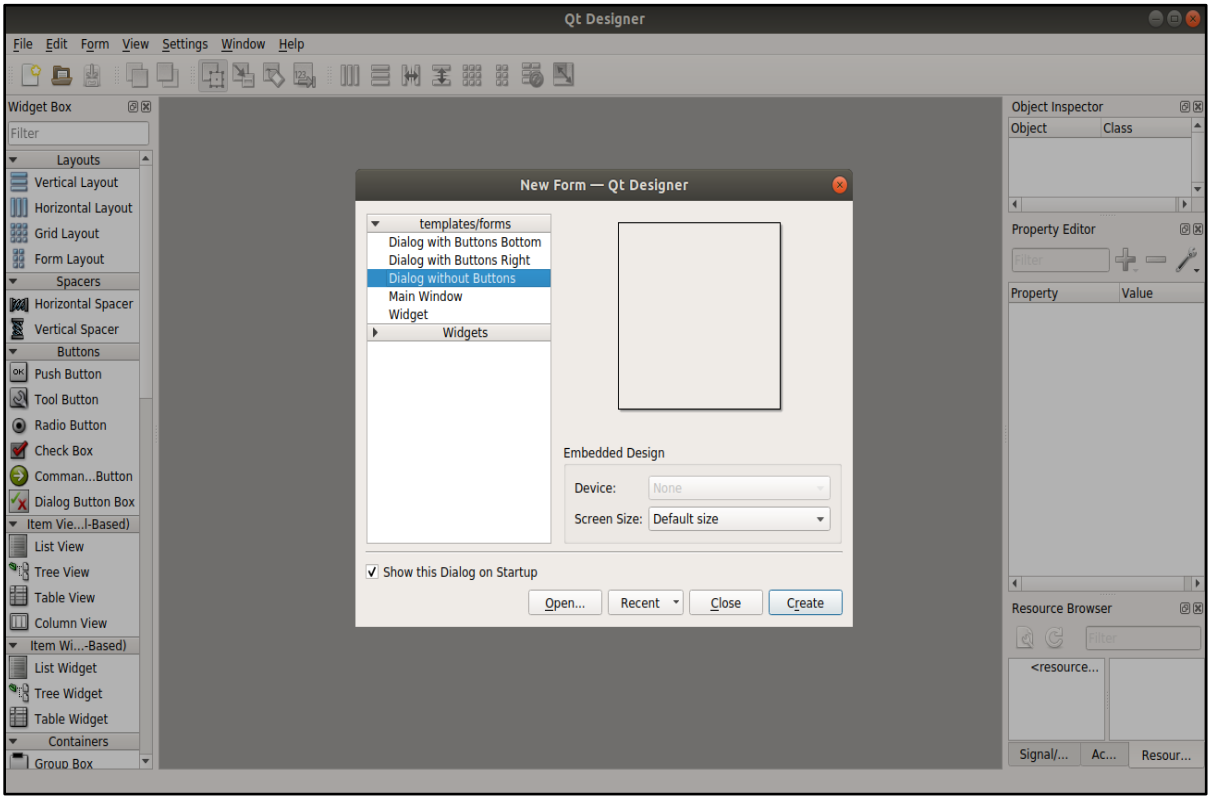


Fig.1: Pre GUI Selection Mode in Qt Designer

Step 2: Create your own GUI here in the blank workspace. Easy! You can now drag and drop the widgets from the side panel onto the window that you are designing.

These below screenshots represents my GUI design:

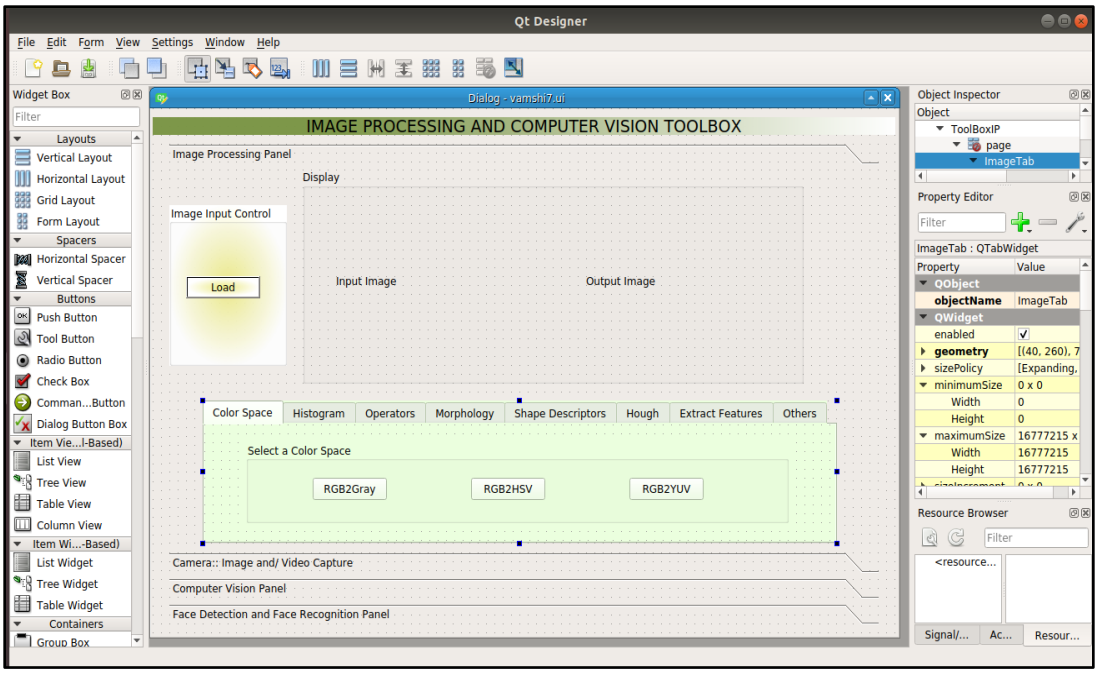


Fig.2: My GUI Layout for the Tool box (Image Processing Tools Section; Sec-I)

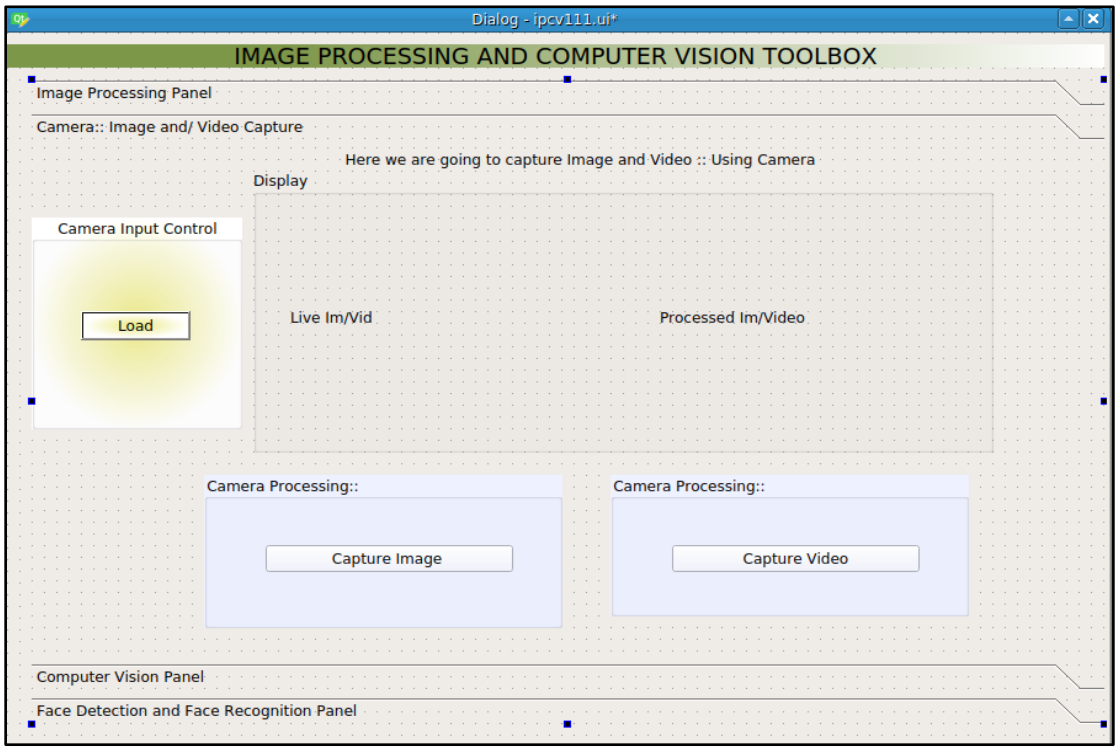


Fig.3: Camera/Video Capturing Tools Section (Sec-II)

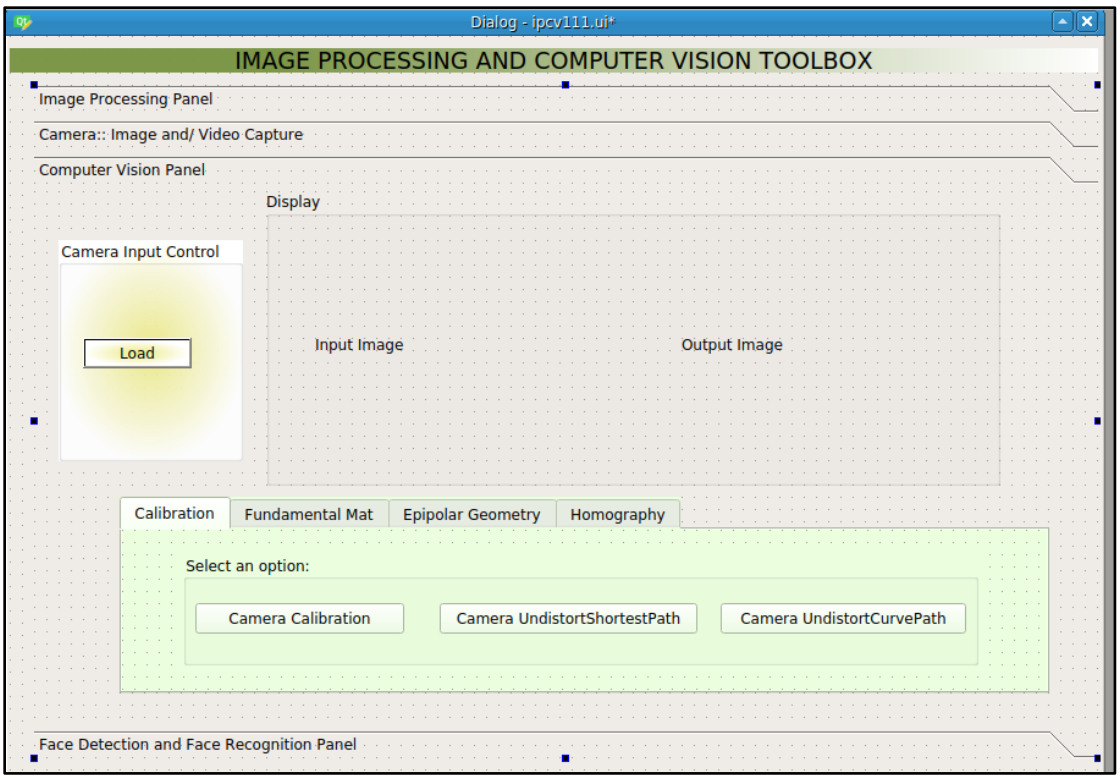


Fig.4: Camera Calibration Tools Section (Sec-III)

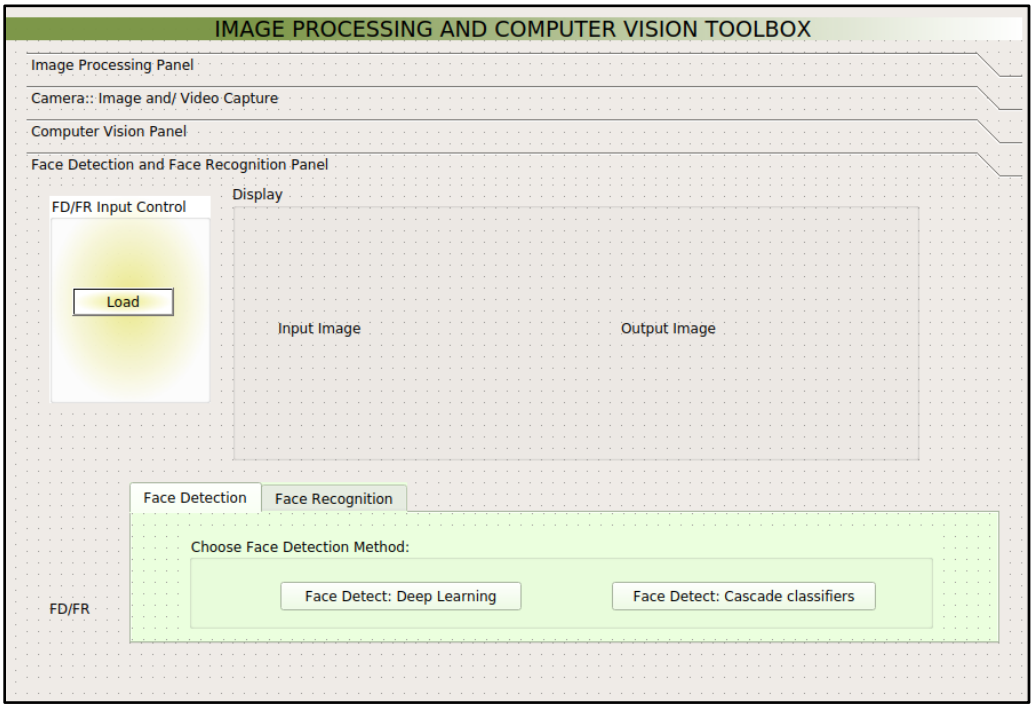


Fig.5: Face Detection and Recognition Tools Section (Sec-IV)

PROGRAMMING IN PYTHON

After GUI Layout >> Save the file with .ui extension and follow the Steps.

Step1: Now, Convert the .ui extension file to .py extension file (as explained in installation section)

Step2: Then, Open that .py extension file (ipcv222.py in Atom Python Script Editor)

Step3: As Python is also Object Oriented Programming Language, you will observe that structure for class & object instances are already created in our program. Especially these 3 sections:

```
class Ui_Dialog(object):
```

```
    def setupUi(self, Dialog):
```

```
    def retranslateUi(self, Dialog):
```

Note: Here in **class "Ui_Dialog"** two methods are created, **setupUi and retranslateUi**

setupUi creates the widget objects in the proper containers and assigns the proper object names to them. **retranslateUi** sets the text and titles of the widgets.

Step4: Importing Libraries (use the below imports). Program starts with **_main_** function and initially QWidgets objects is creates and reused again and again.

Caution: imports should be written on the top of the programming lines)

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtWidgets import (QApplication, QCheckBox, QComboBox, QDateTimeEdit,
        QDial, QDialog, QGridLayout, QGroupBox, QHBoxLayout, QLabel, QLineEdit,
        QProgressBar, QPushButton, QRadioButton, QScrollBar, QSizePolicy,
        QSlider, QSpinBox, QStyleFactory, QTableWidget, QTabWidget, QTextEdit,
        QVBoxLayout, QWidget, QFileDialog,QMessageBox)

import numpy as np
import matplotlib.pyplot as plt     # Python-2D plotting library
import cv2 as cv                    # a library of Python bindings of Computer Vision tools
import glob                         # finds all the pathnames matching a specified pattern
from IPython.display import Image
import sys

import os
```

Step5: Now comes the linking part! I will explain you with an example.

For suppose, you want to Load and Image. You need these:

1. appropriate library import
2. button class constructor, object, instances and method calling (as I explained before, all these are structured after .ui file to .py file conversion)
3. proper written definitions for the buttons: You should have exact idea of what that particular button should do on its action click.
4. Most importantly, choose the objectNames as the functionalities they perform *(It's a programming sign convention. This makes programmer to easily identify the button and its functionality)*
5. Use OpenCV Source codes available on https://opencv.org/. Modify and definitions for the described buttons and GUI componets as per your requirements (based on the inputs, functionalities and processed outputs).
6. Also I attach tooltips for each and every section. You can observe them when you run the program.

Here, I need to write this below line in linking section:

```
        self.LoadButton.clicked.connect(self.on_LoadButton_clicked)
```

Also, I need to define the functionality as:

```
    def on_LoadButton_clicked(self):
```

Step6: Making use of the python opencv documentation and python tutorials were explained in https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html. That is what I exactly did.

Step7: All the Sections with subsections were detailed in the OpenCV Sources. So, in the next section, I will explain how my GUI works on button clicks, overall functionalities of each module and how it produces the Output.

NOTE:

1. I tried my maximum to keep the GUI inputs dynamic; especially for Image Processing Section, Camera and Video Processing Modules to load Image/Camera/Video via pushbutton of user choice. I made them dynamic.
2. For Camera Calibration, I designed module where user has to click on the Camera Calibration button in-order to generate the calibration of the chess board pattern. Similarly for Face Detection and Recognition Section, on pushbutton click, program automatically absorbs the static images from folder without user's choice. *(Since they have multiple inputs during execution, I cannot find time to do it. So, I arrange them as a static input)*

MODULARITY DIAGRAM FOR TOOLBOX FUNCTIONALITIES

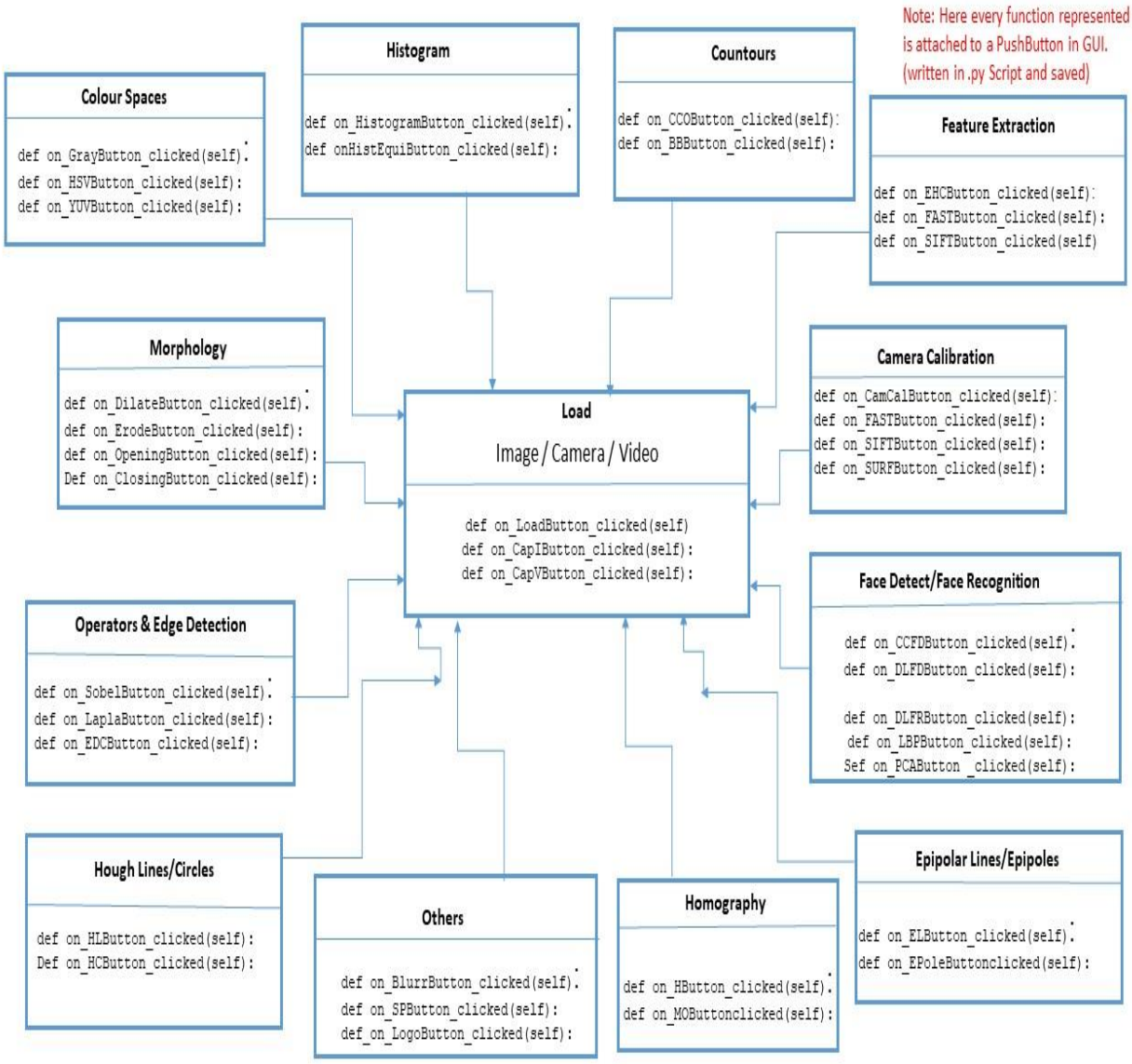
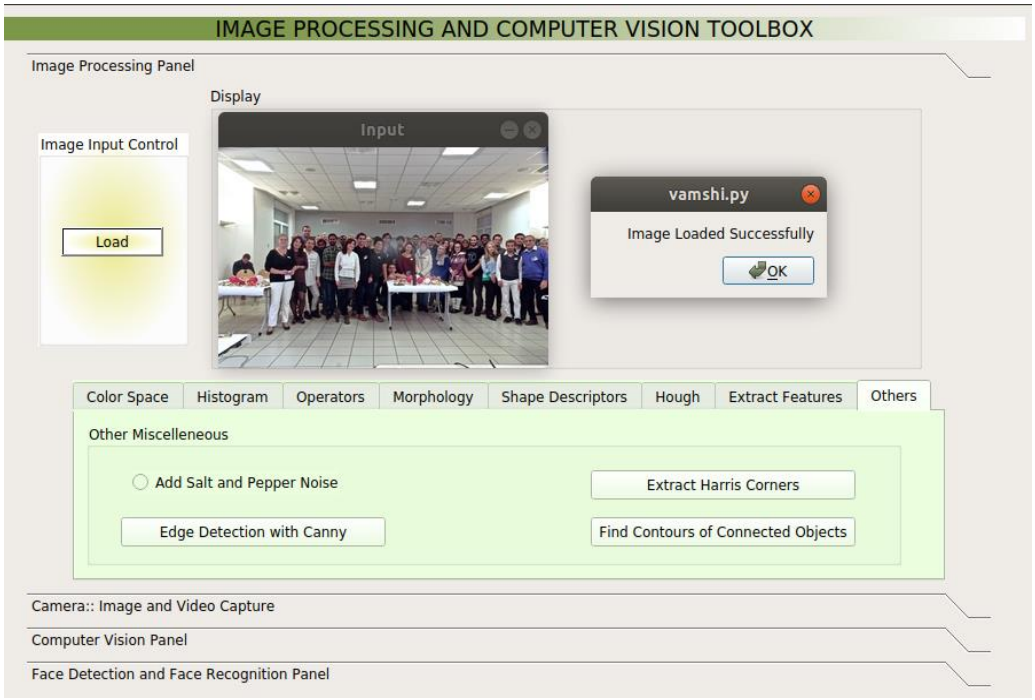


Fig.6 Functional Definitions to represent the modules of CV Toolbox

6. Processing Tasks for Images

Load Image : Click on load image button and Choose an image (I choose 'fam.png').

See the below figure.



6. Image Processing Section (After you load image). Start Processing

6.1 Add noise

I have implemented function to add salt and pepper noise. the result is as shown in Fig.3

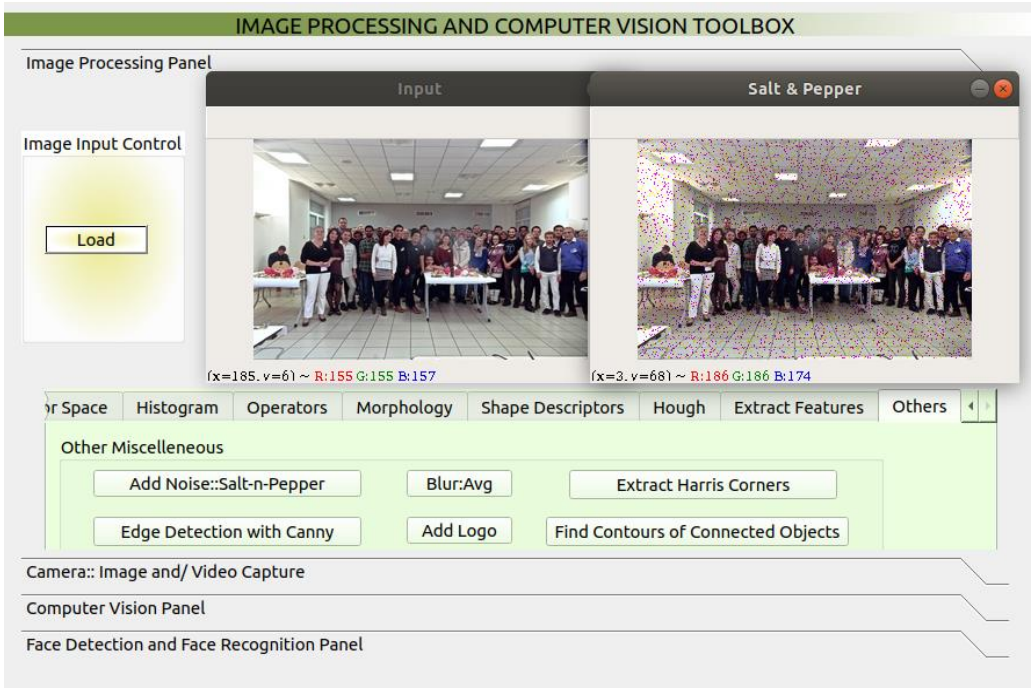


Fig.7 Applied Salt and Pepper noise on input image

6.2 Add Logo

Now, I click on add logo button and choose other image ‘face.jpg’. Choosing ROI press enter. It provides logo in the same image. The result is as shwon in Fig. 4.

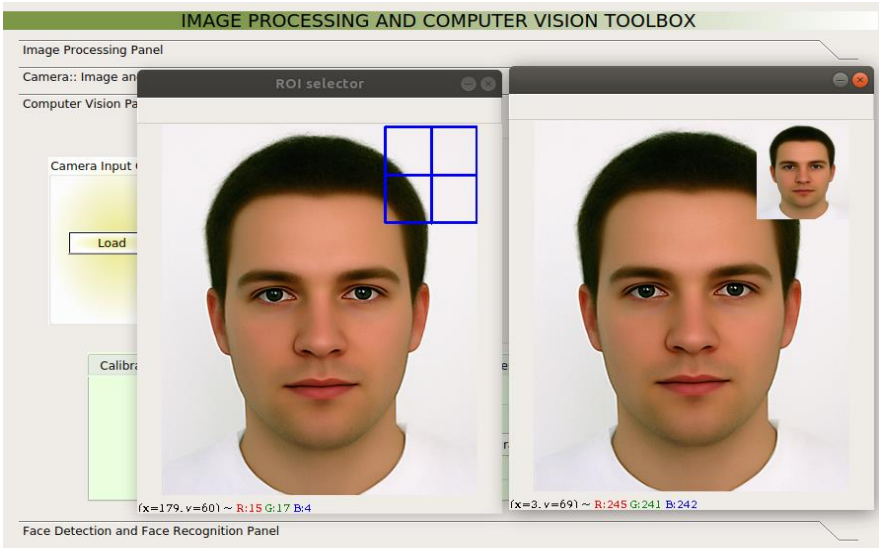


Fig. 8 Add logo to image

6.3 Change Color Space

We can change color spaces in opencv easily using 'cvtColor'. Four colorspace conversions are implemented as seen below. The result is as shown in Figures.

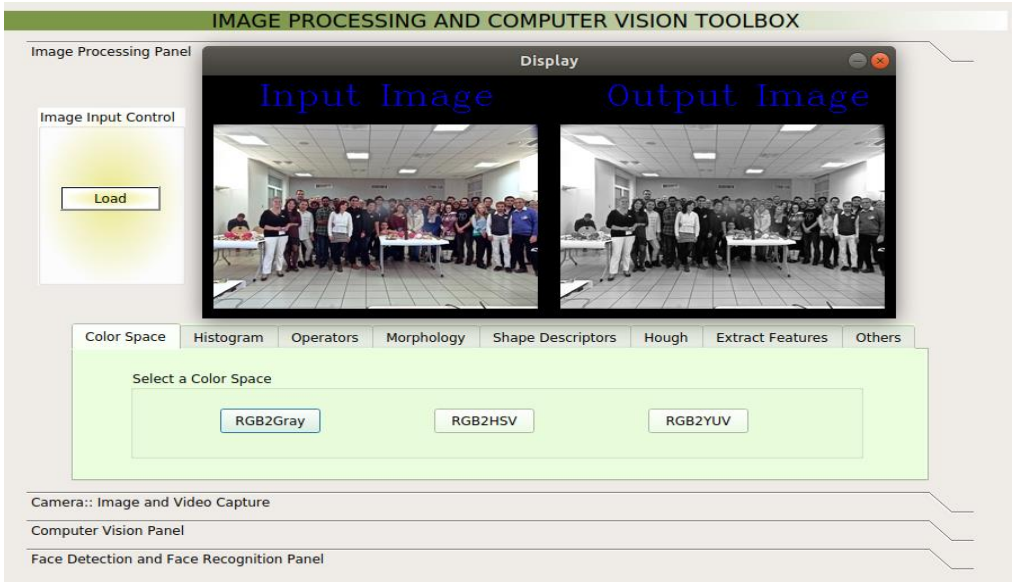


Fig. 9 RGB to Gray

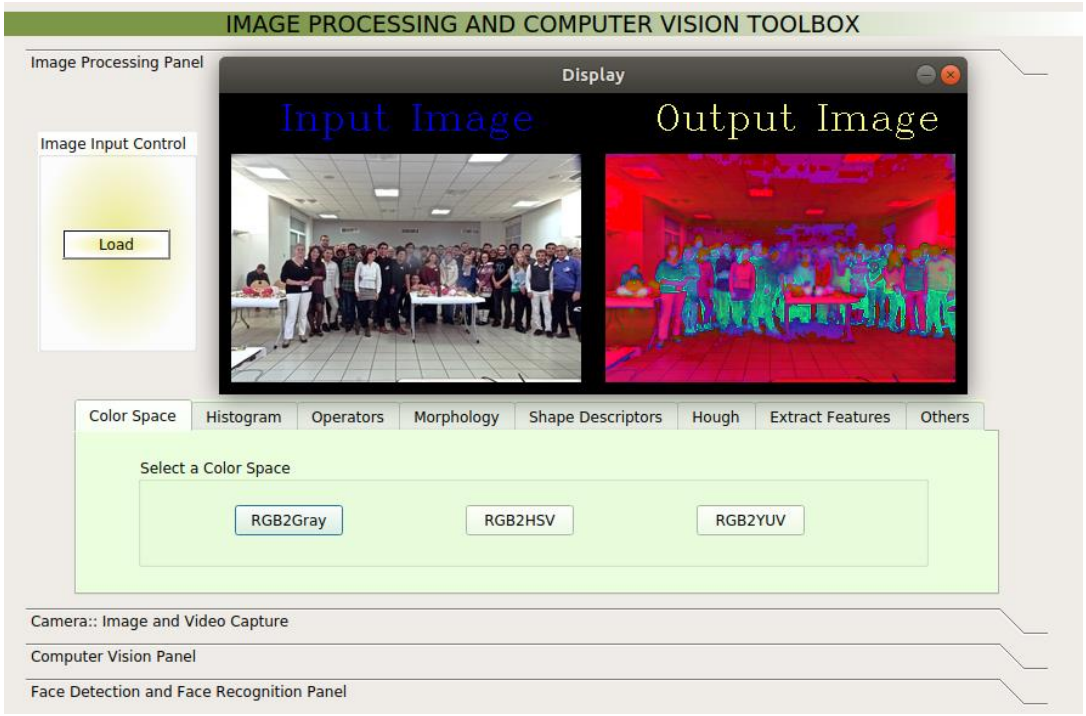


Fig. 10 RGB to HSV

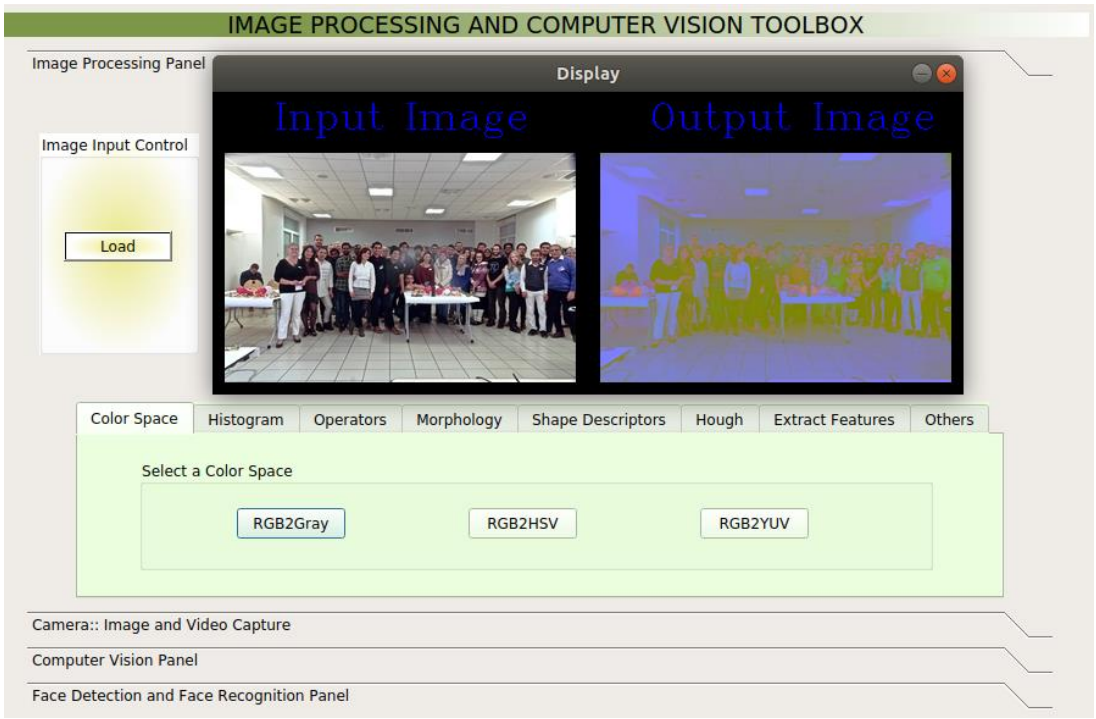


Fig. 10 RGB to HSV

6.4 Histogram

Load an image and then Choose Histogram Button. Similarly, Histogram Equalization

The results for plotting histogram of image are as shown below.

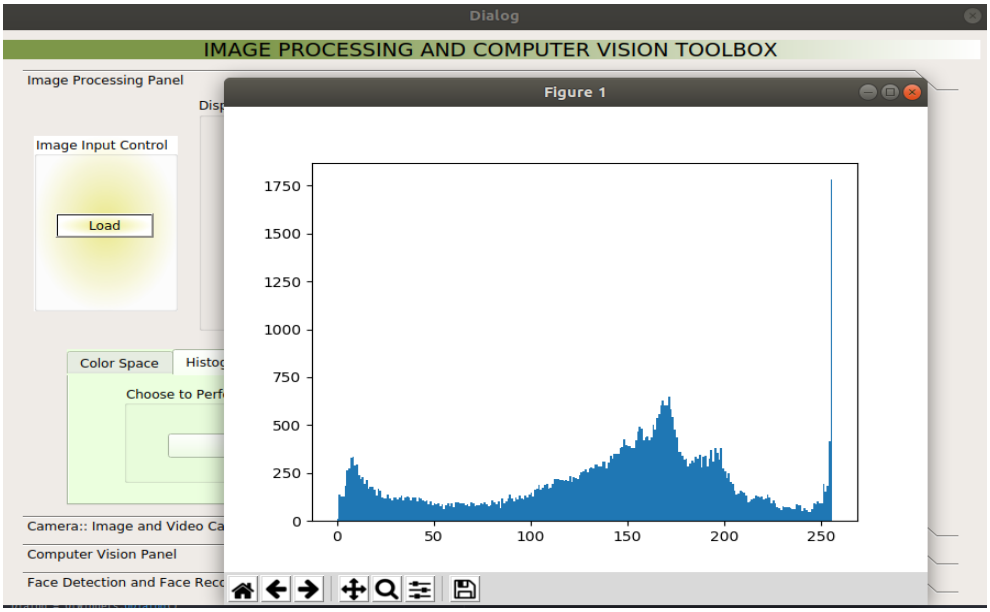


Fig.11 histogram of an image

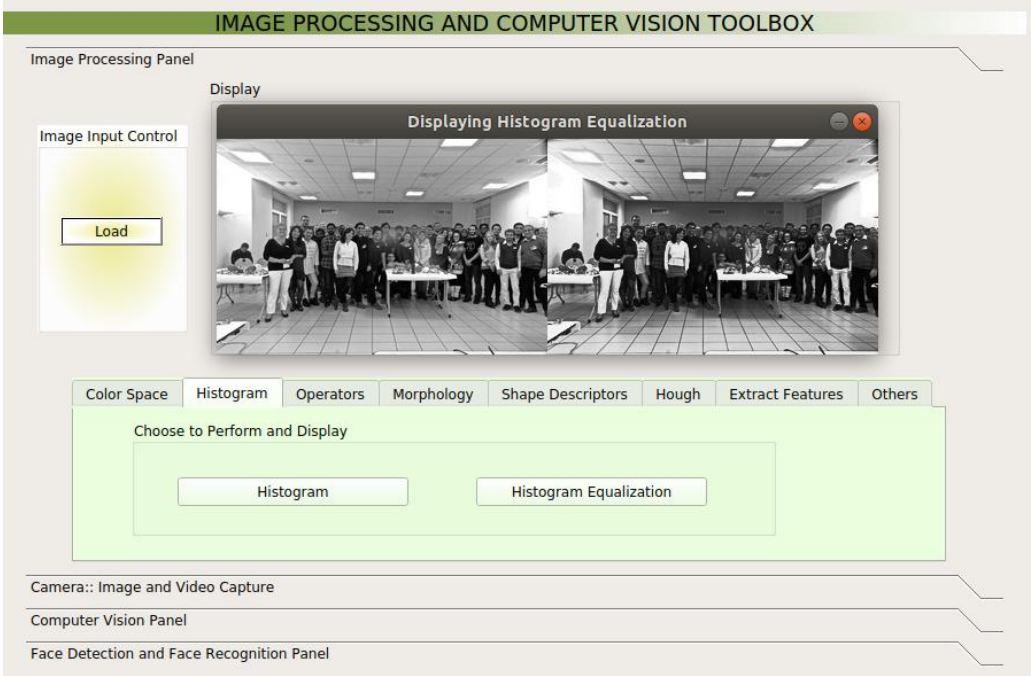


Fig. 12 Equalized histogram

6.5 Morphology:

It constitutes Dilation, Erosion, Open, Closing. The result are as shown below.

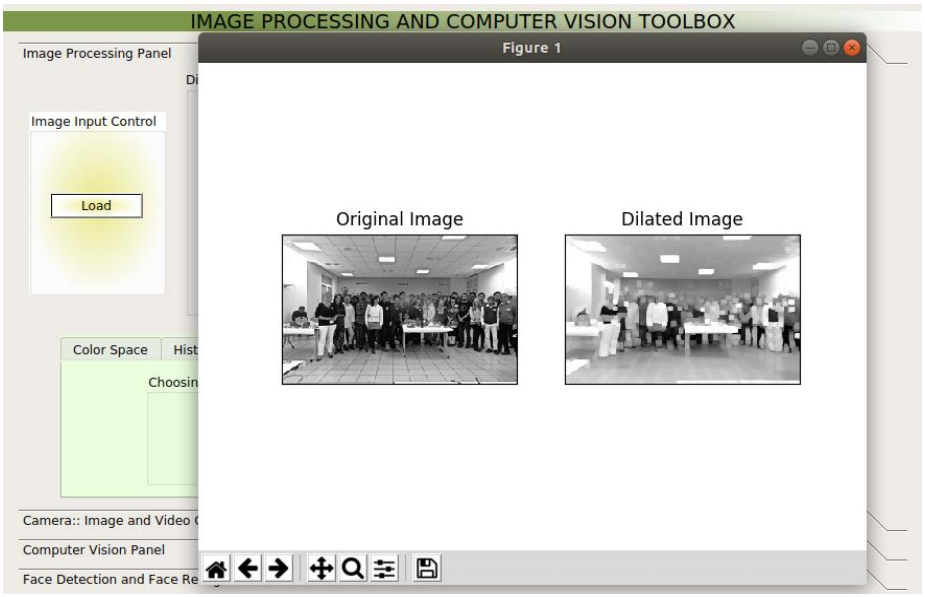


Fig.13 Dilated Image

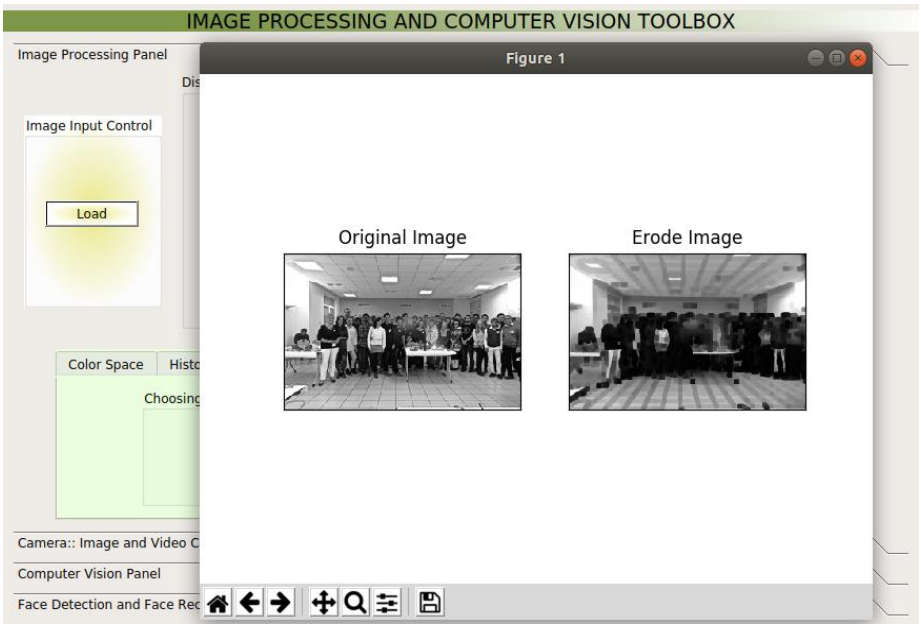


Fig.13 Erode Image

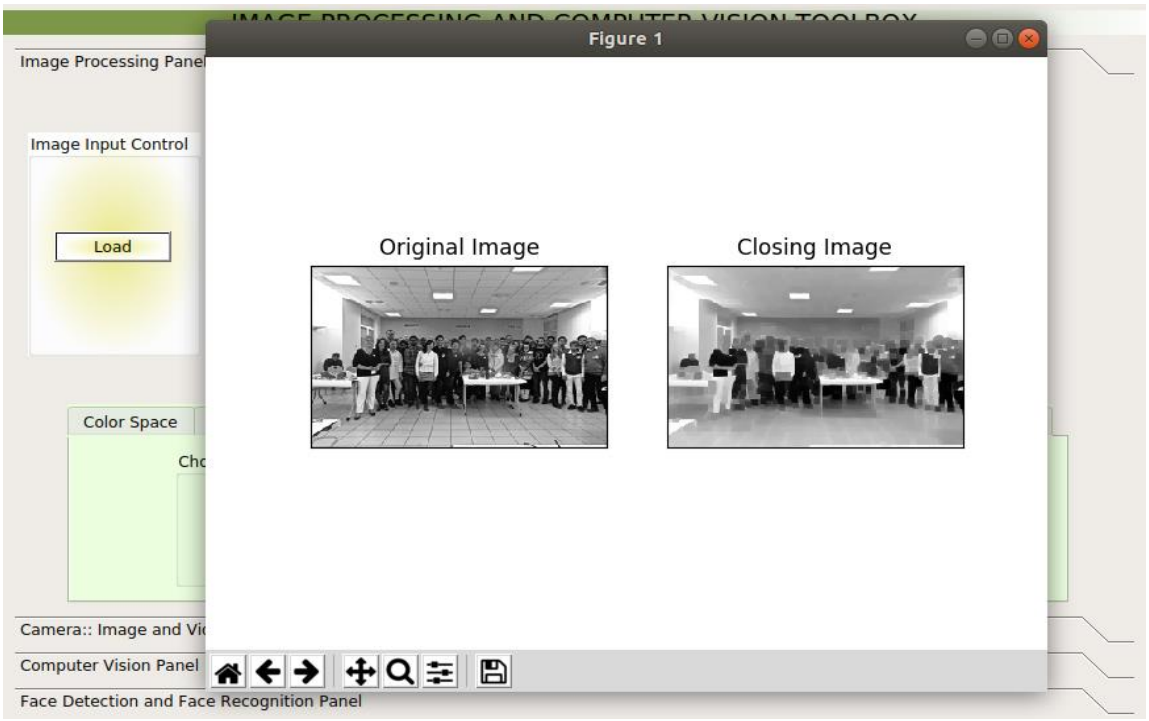


Fig.14 Closing Image

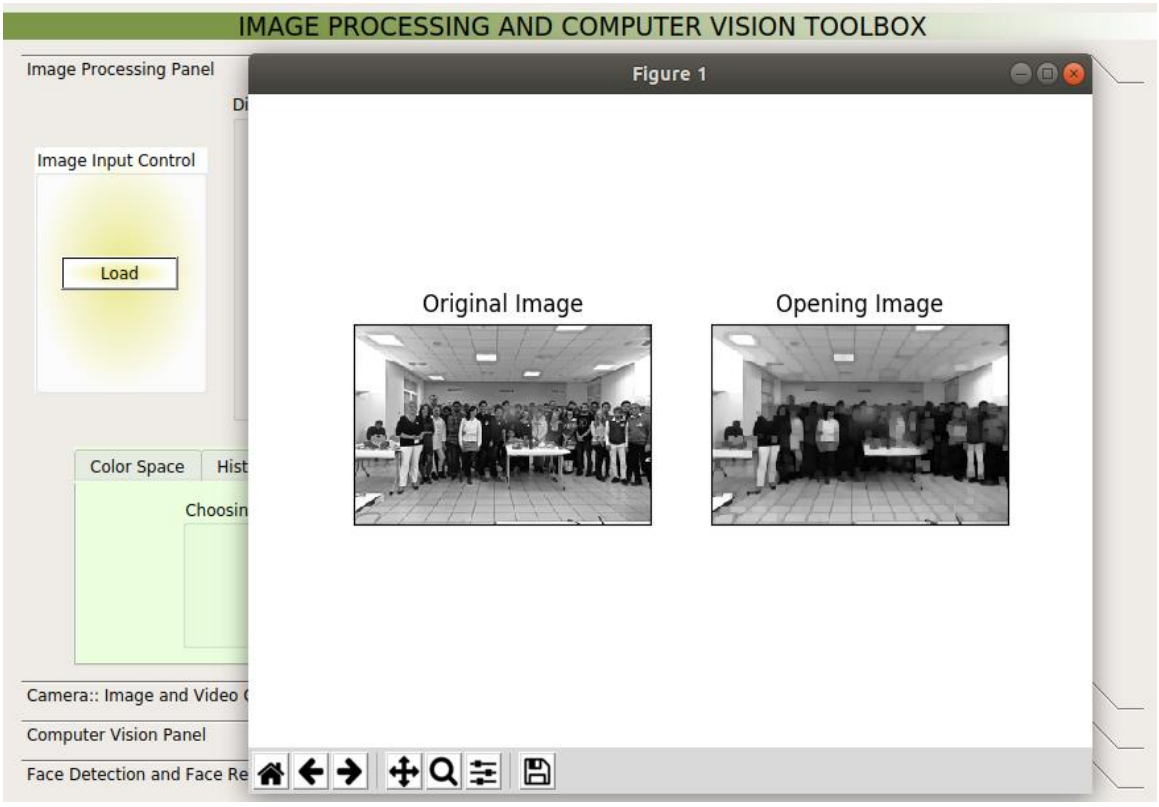


Fig.15 Opening Image

6.6 Blur: Apply Average Blur on Input Image

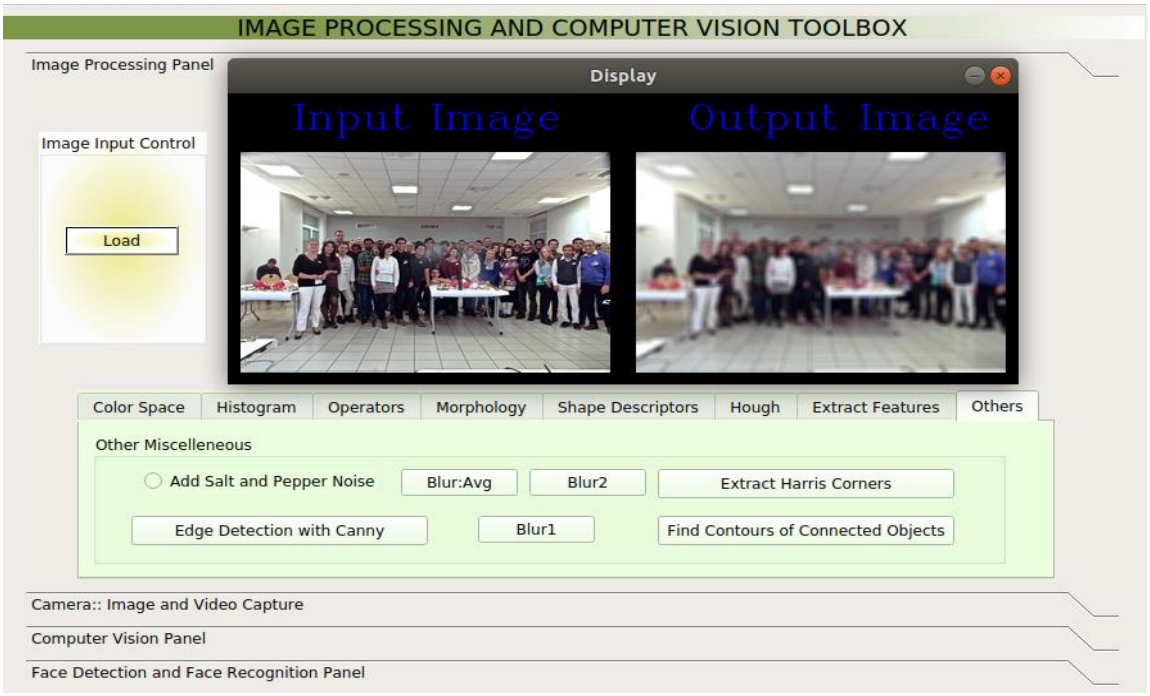


Fig.16 Blurring an image

6.7 Apply Operators: Apply Laplacian and Sobel Operators on the selected input image

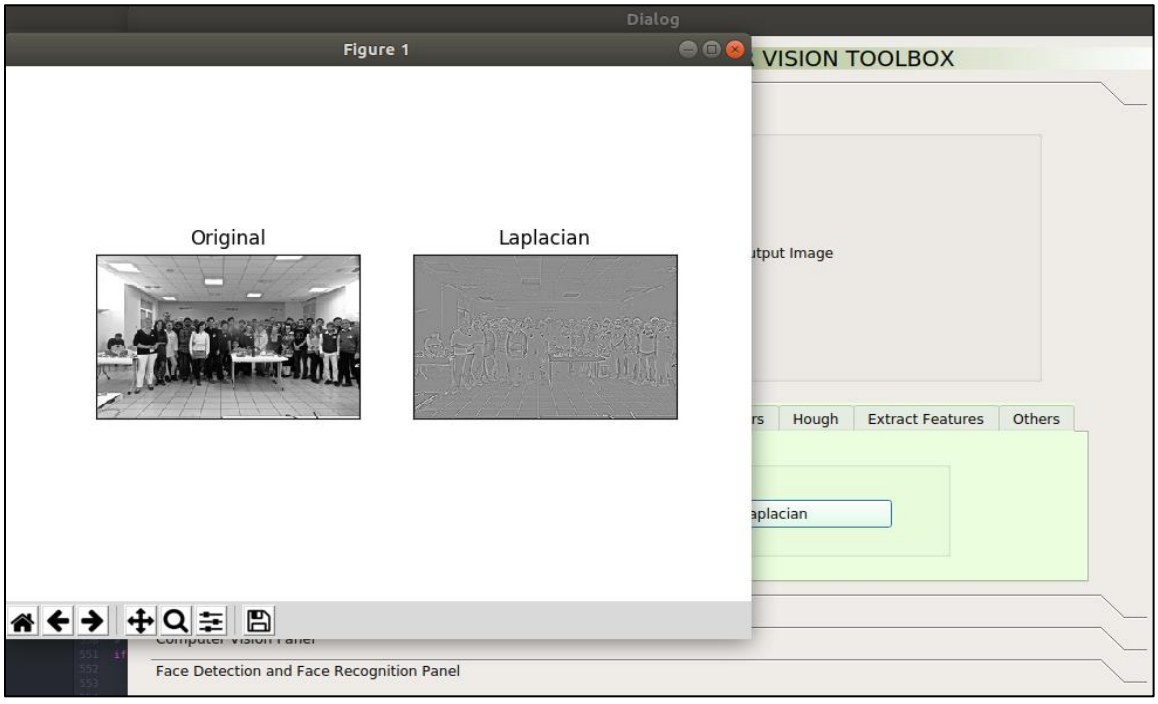


Fig. 17 Laplacian Operator applied

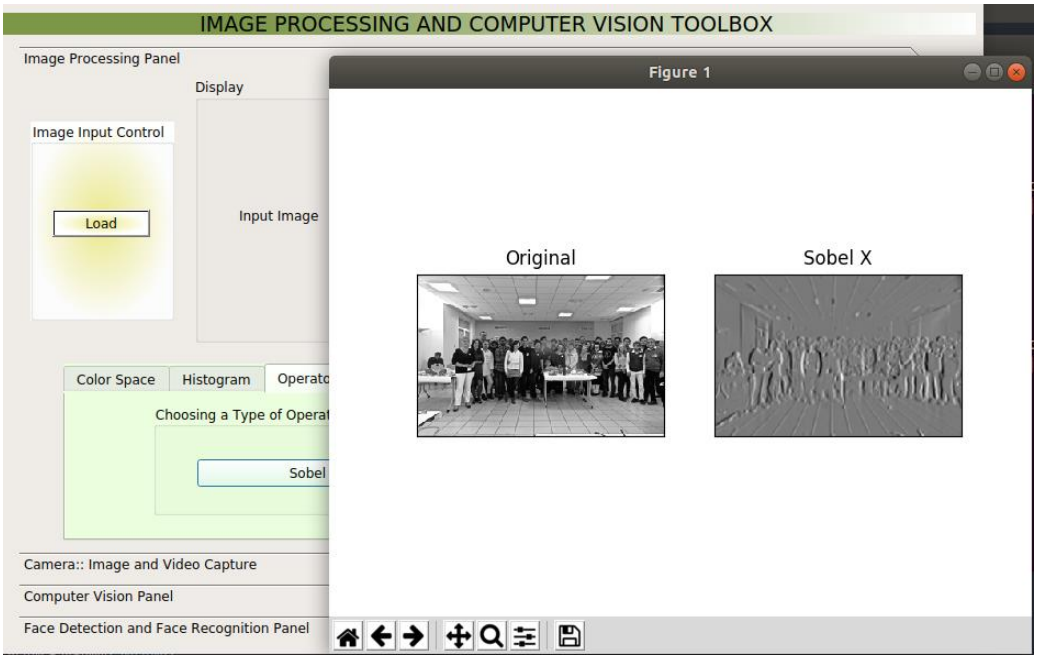


Fig.18 Sobel-X Operator applied

6.8 Canny Edge Detector

The results are as shown in fig. 19.

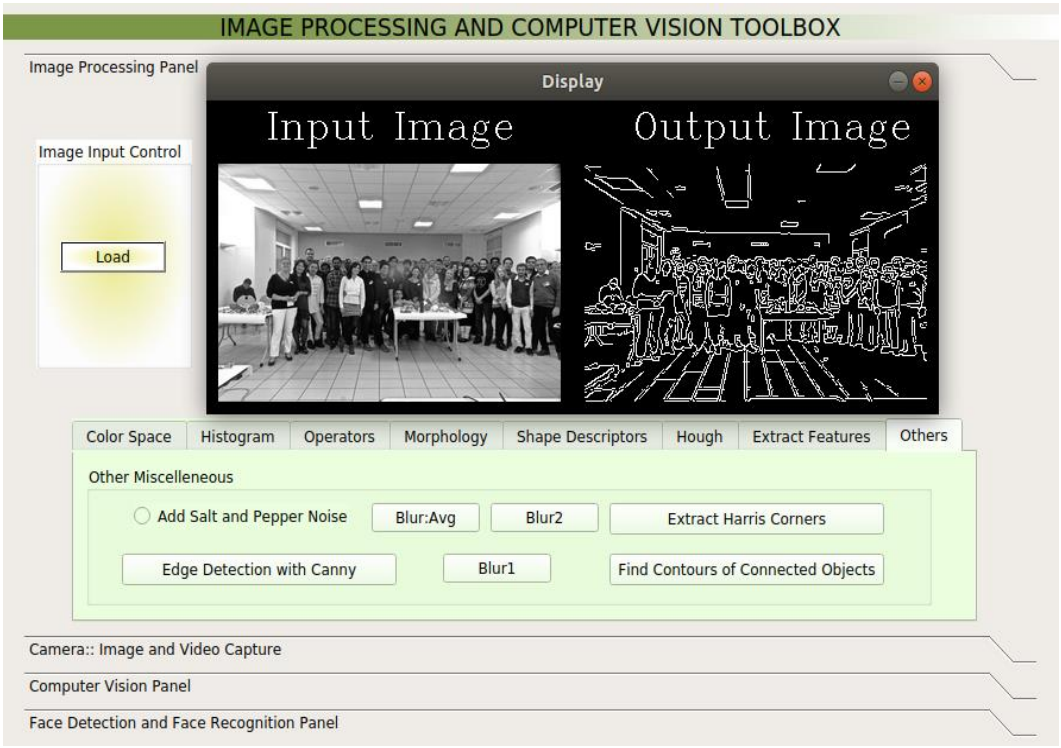


Fig.19 Canny Edge Detection

6.9 Lines and Contours

6.9.1 Hough transform

Hough transform is used to detect lines and circles in an image.

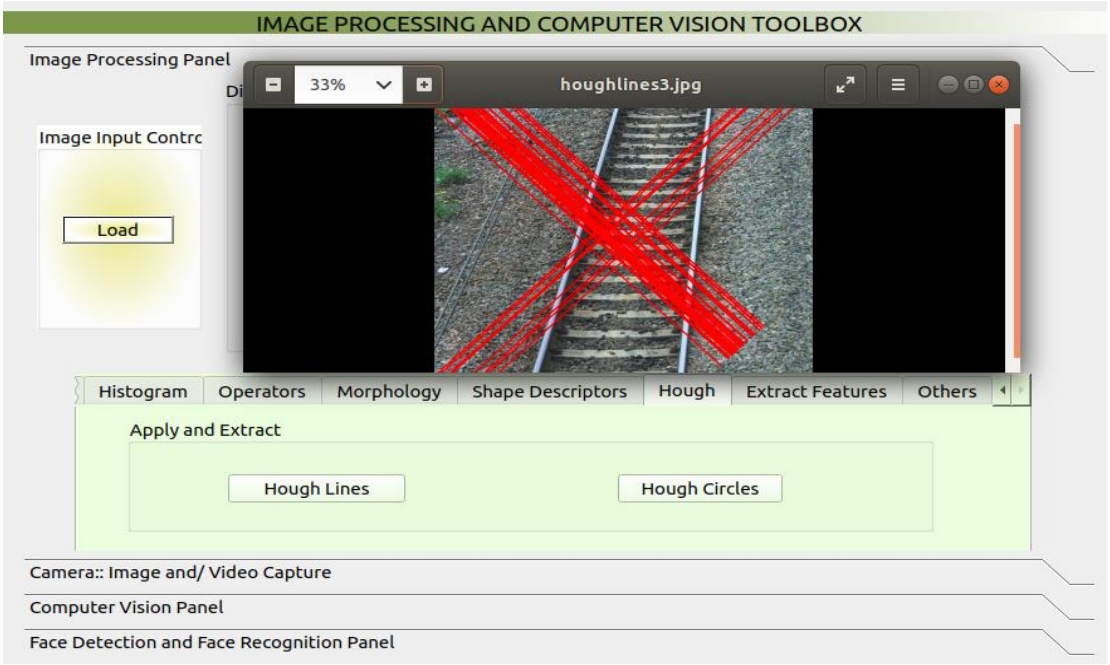


Fig. 20 Hough lines

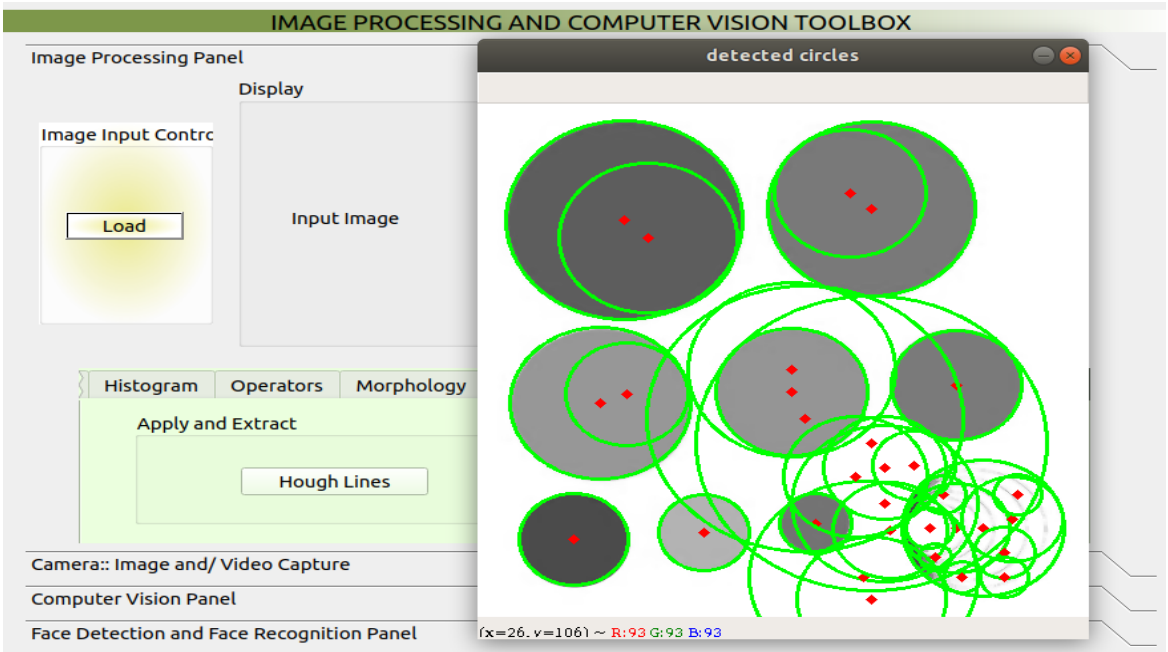


Fig.21 Hough Circles

6.9.2 Contours

Contours and drawn on binary images with minimum and maximum contour length as parameters and converted back to ColorSpaces.

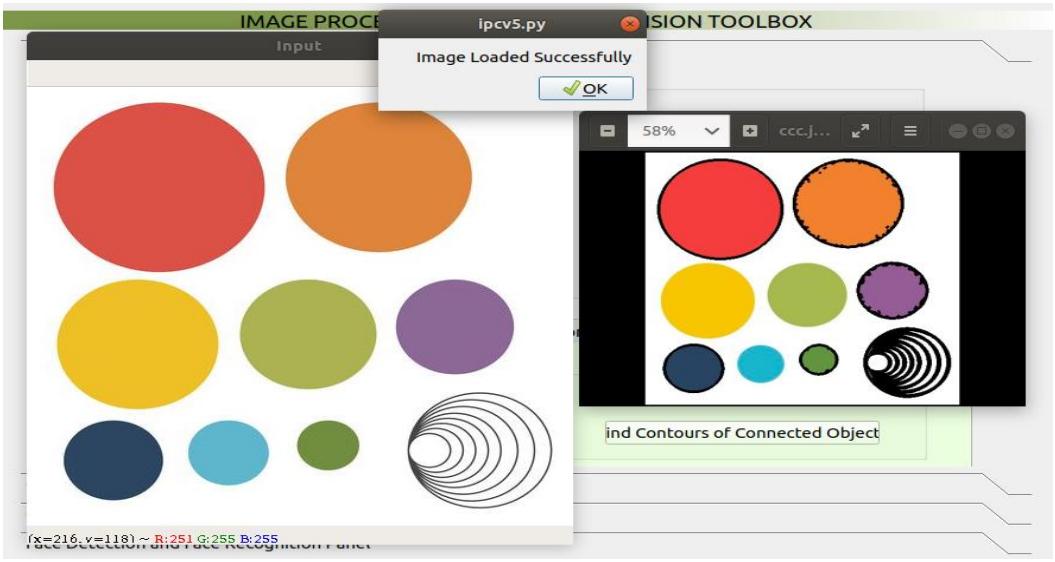


Fig.22 Contours

6.9.3 Shape descriptors

Different types of shape descriptors are impended such as bouding box, minimum enclosing circle, polygon enclosing, convex hull and centroids. White Bounding Box is seen in the output.

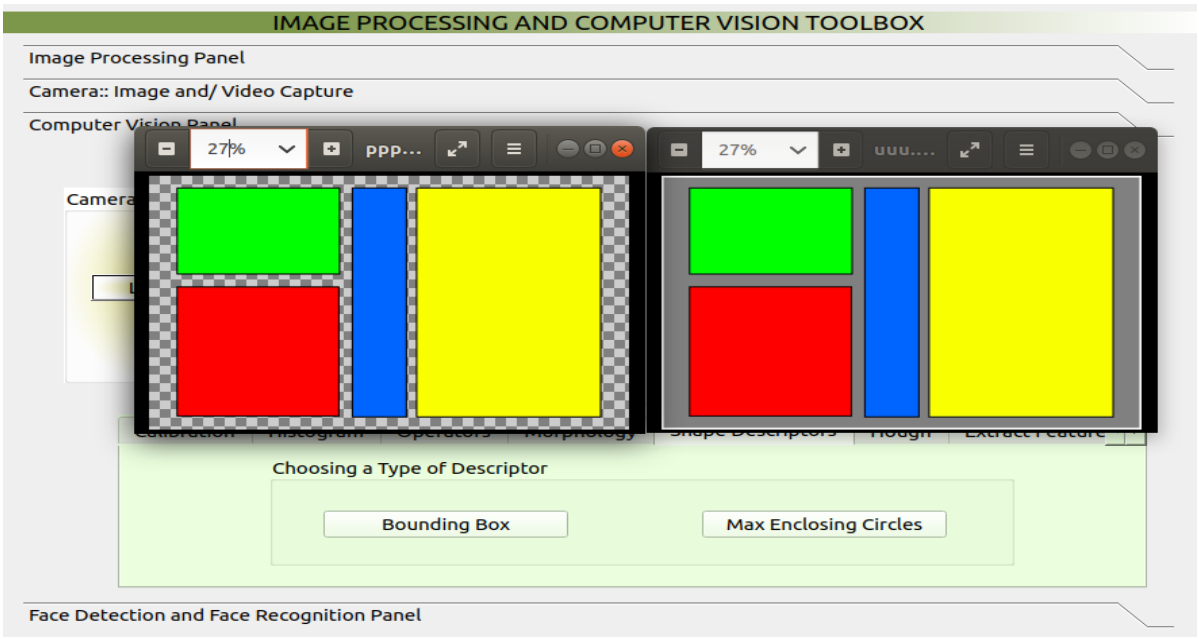


Fig.23 Bounding Box as a Shape Descriptor

6.10 Features points

6.10.1 Corner points(Harris Corner Detector)

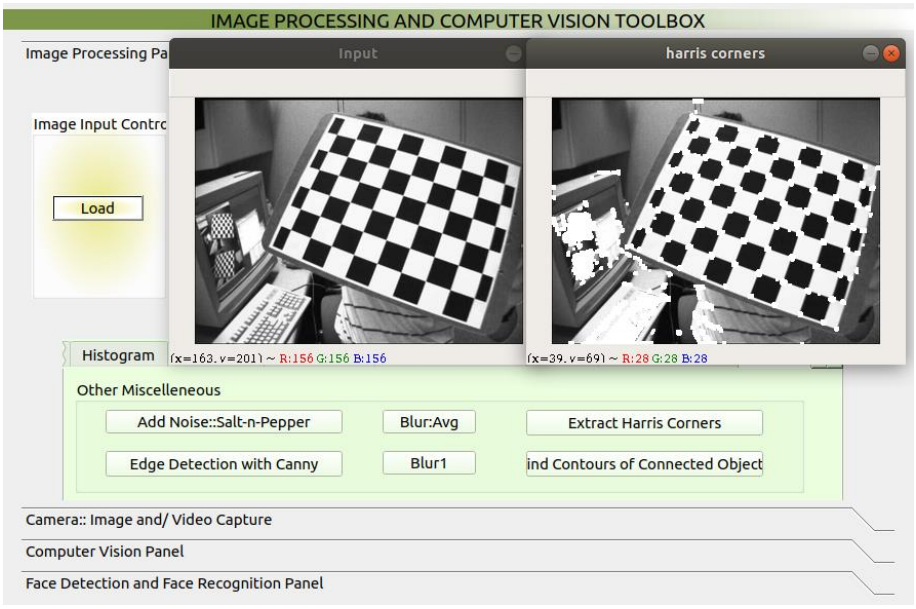


Fig.24 Corners using Harris corner detector: Observe White Corners

6.10.2 Other features (FAST,SURF, SIFT)

The feature detection and description plays an important role in finding matches or corresponding points which plays significant role in computer vision. .

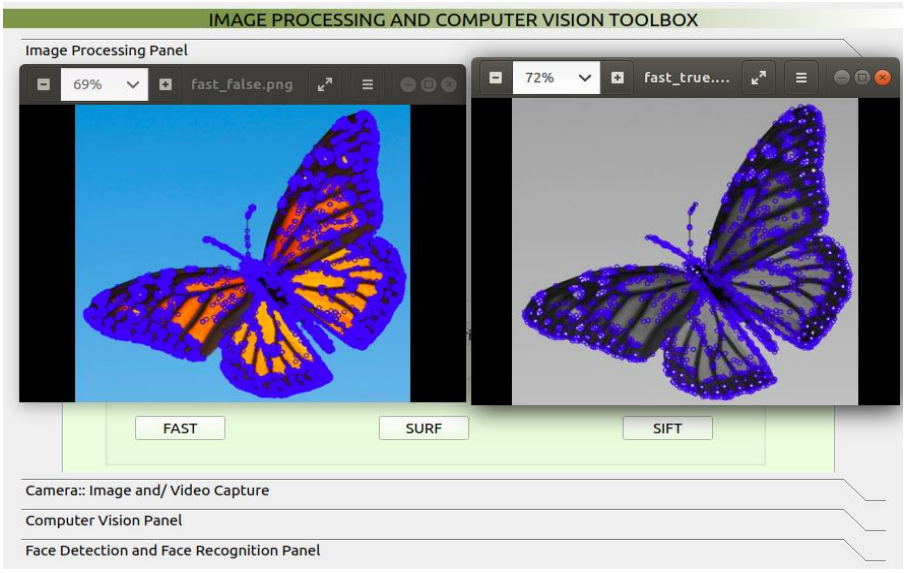


Fig.25 FAST Features

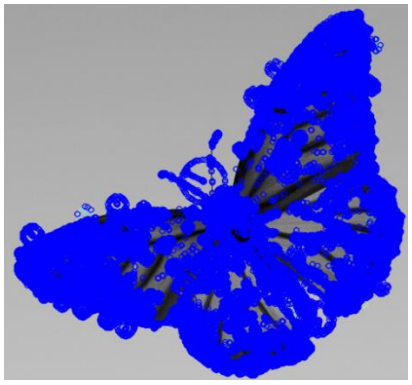


Fig. 26 False FAST

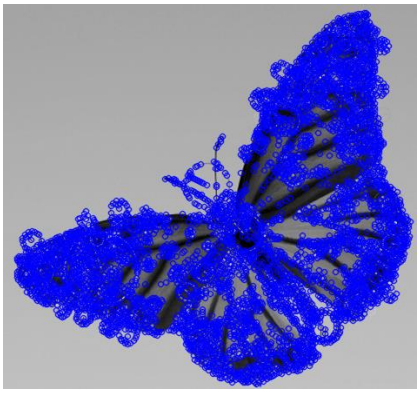


Fig.27 True FAST

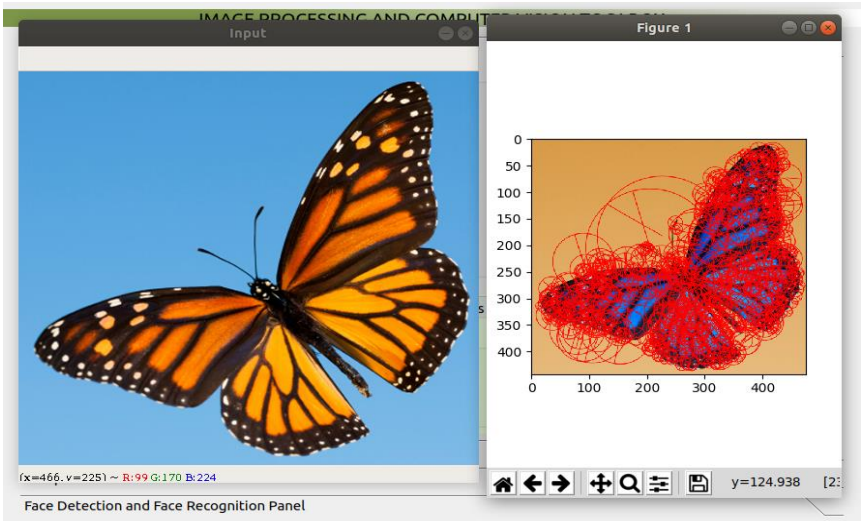


Fig.28 SURF Features

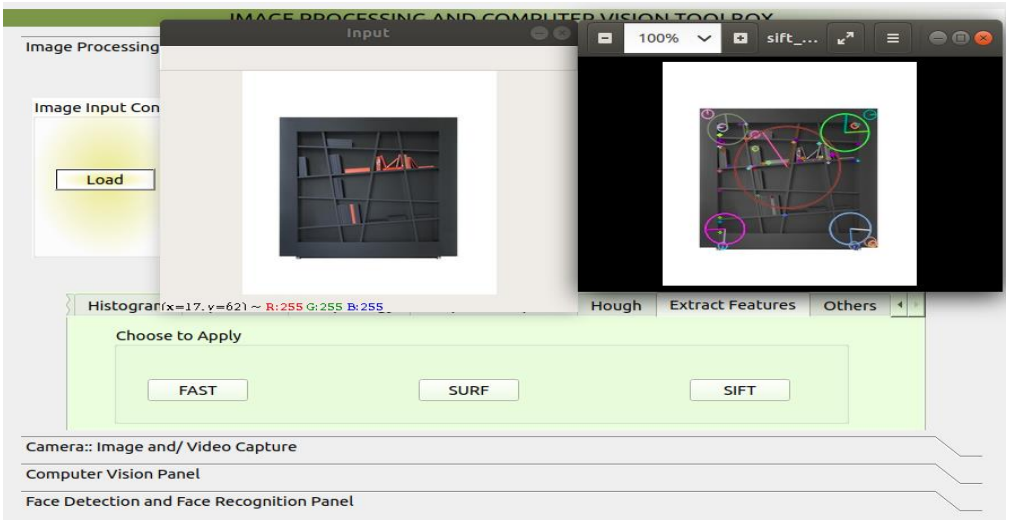


Fig.29 SIFT Features

7. Camera Input for Image Capture and Video Capture

7.1 Image Capturing

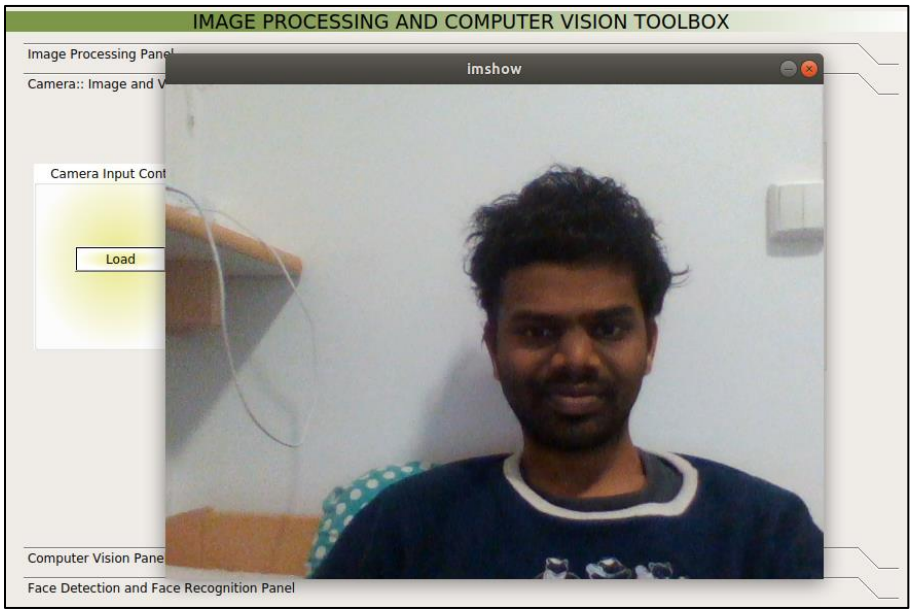


Fig.30 Web Camera Access and Image Capturing

7.2 Video Capturing

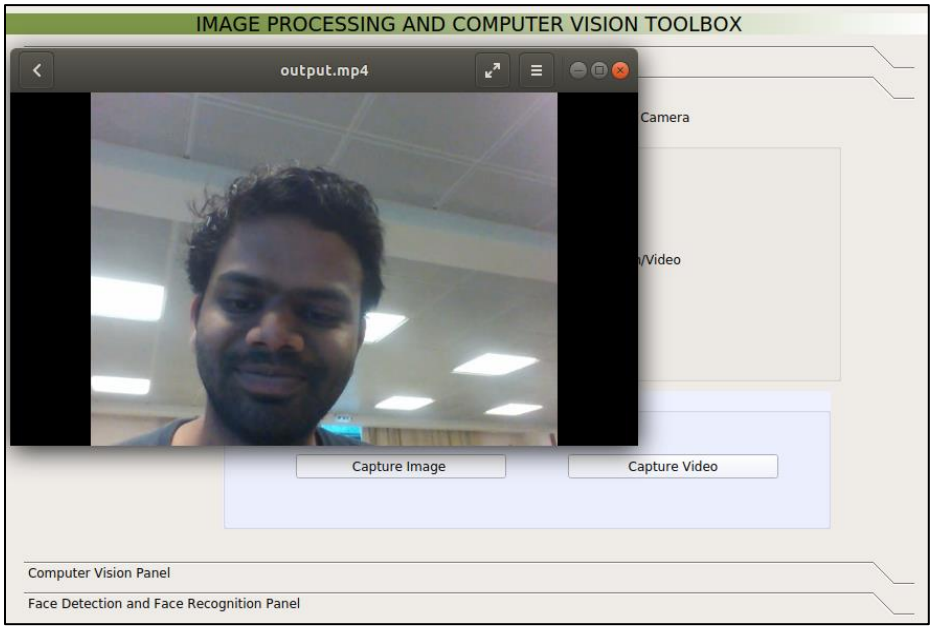


Fig.31 Web Camera Access and Video Capturing

Observation: Video is the sequence of images. If we can process on images, we can adjust the frame rate and run a video after processing

8. Camera Geometry and other

8.1 Camera calibration

Camera calibration is a major task in computer vision. A calibration pattern like a chess board pattern is taken and images are captured using a camera to calibrate it. The images are processed accordingly to find the camera calibration parameters. These parameters can be used to undistort any distorted images taken from the camera.

The results is as shown in fig.32 & 33

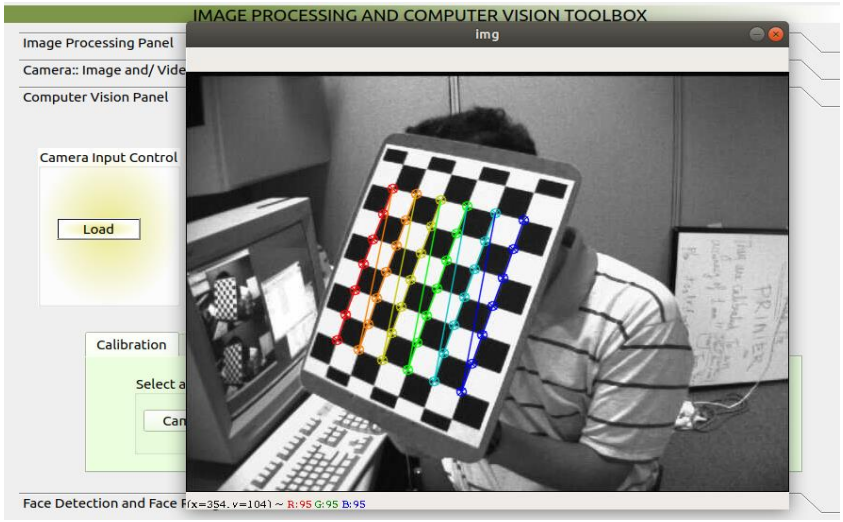


Fig.32 Camera calibration

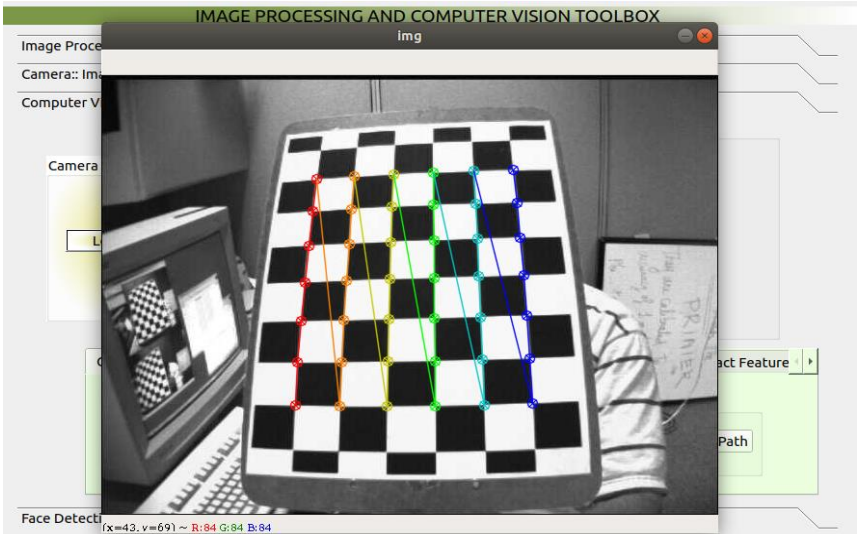


Fig.33 undistorted image using calibration

8.2 Fundamental matrix

By finding the matching points between two images using Surf or Sift, we can make use of the matching points to get the fundamental matrix between two views.

I have implemented three types : 7-point, 8-point and Ransac.

8.2.1 Image Matching:

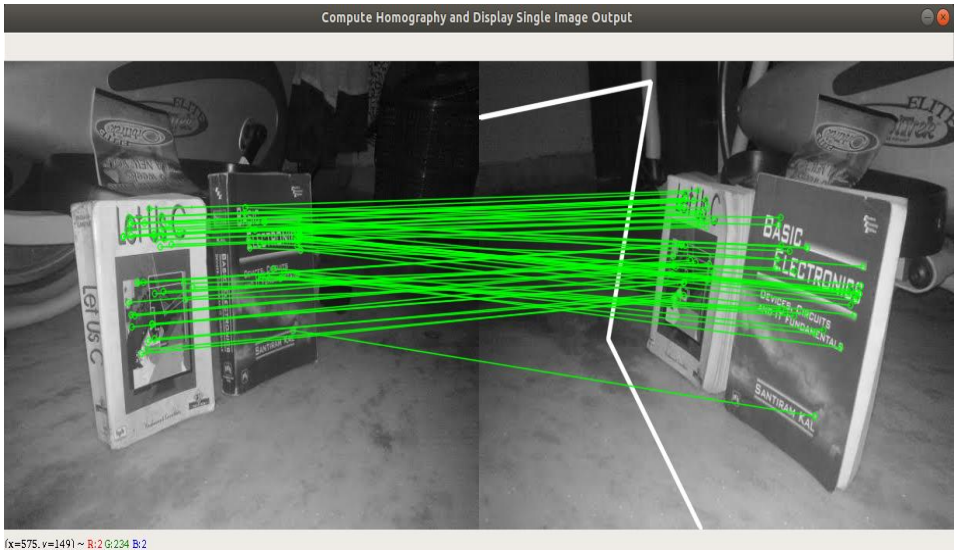


Fig.34 Image Matching in first and second images

Epipoles and Epilines:

By computing the fundamental amtrix, we can check the accuracy by drawing epipolar lines. From epipolar geometry, the matching point of first image in second image should lie on the epipolar line of the point in first image.

Due to missing functionalities or missing function SIFT I cannot find Epipoles and Epilines

```
(toolbox) vamshi@vamshi-Inspiron-3542:~/Downloads/Mar2019$ python ipcv5.py
Traceback (most recent call last):
  File "ipcv5.py", line 941, in on_MECButton_clicked
    sift = cv.SIFT()
AttributeError: module 'cv2.cv2' has no attribute 'SIFT'
Aborted (core dumped)
(toolbox) vamshi@vamshi-Inspiron-3542:~/Downloads/Mar2019$
```

Fig. 26 Epipolar lines in first and second images

8.3 Homography

From the corresponding points we can also calculate the homography matrix and use the homogrpahy to mosaic two images.

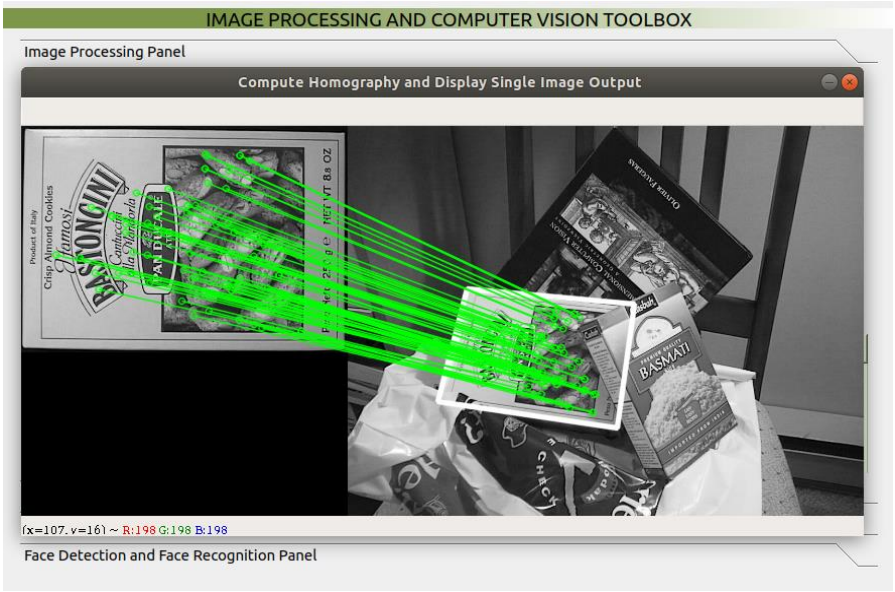


Fig.35 Homography Computation and Feature Matching

9. Face Detection:

a) Using Cascade Classifier:

Given, input images and features likes mouth, nose, eyes, ears, we can classify whether the given images contains face or not. Also we can categorize the pars of face as shown below. This method uses “Cascade Classifier” with .xml file extensions. Fig.35 &36 demonstrate the outputs of Face Detection

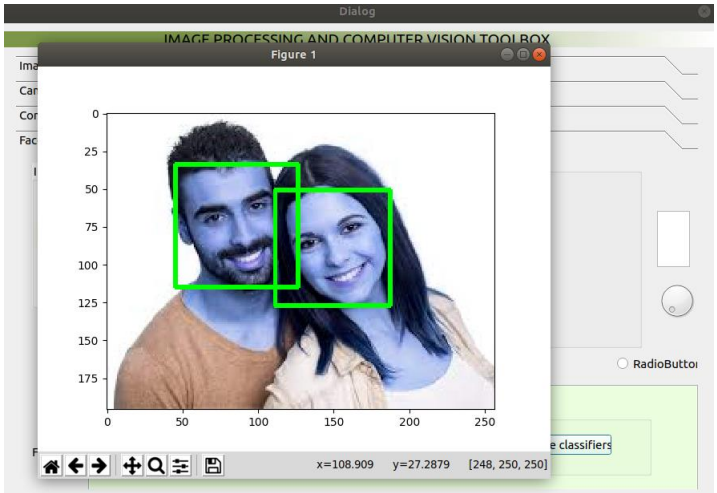


Fig.35 Face Detection using Cascade Classifier

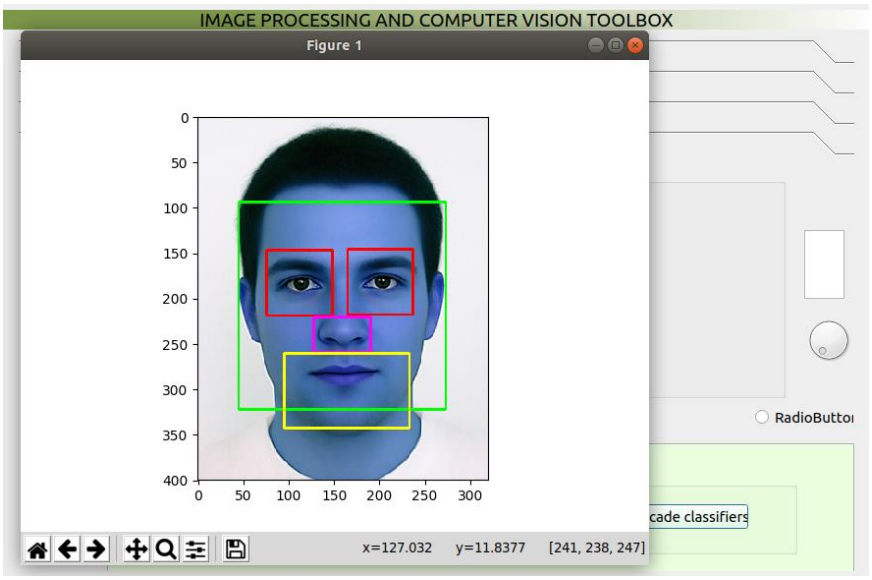


Fig.36 Face Features classification Output

10. Face Recognition:

a) Using Local Binary Patterns:

Given, input images and set of labels for the images. This algorithm on button click derives the Name of the person in the images as per the prior training data.

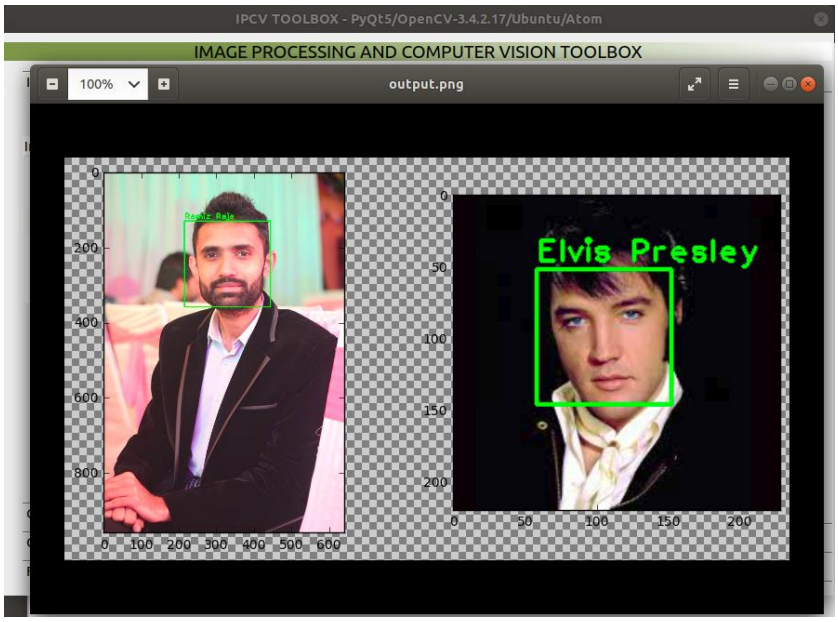


Fig.37 Face Recognition by Local Binary Patterns- Output

Note*: Though out all the project I used different image inputs and processed Outputs upon them

PROS AND CONS

Limitations:

1. I mostly used pushbuttons in the project, which limits user choice of entering inputs. It limits the dynamic usage of the project when tested with different input images especially in Camera Calibration part.
2. Indentation errors are common and often required to solve.
3. Single Lengthy GUI .py file to execute every time. Any change in GUI .ui file effects the .py file functions and cause errors. So, every change has to be considered while using .py file.
4. Due to lack of time, I could not complete Face Recognition Tool, Epipolar Lines and Epipoles and 7/8/RANSAC.

Pros:

1. Efficient, easy and quickest way to learn OpenCV project functionalities through PyQt5.
2. Unlimited extensions for the functionalities is possible
3. Easy for Python/OpenCV beginners.

Difficulties:

1. At first I started my project in C++, I see that C++ handling is more difficult than python. If you are a beginner in python like me, I suggest my approach to you to build CV Toolbox project which is more handy.
2. Handling Libraries and Installing and checking for all available modules has killed half of my work time on this project.

TASK ACHIEVEMENT: Due to lack of time, I could not include these tools (from task sheet): Epilines-Epipoles, Deep Learning for Face Detection, PCA and Deep Learning Technique for Face Recognition

CONCLUSION

This project has given me a great exposure of the OpenCV as a powerful computer vision and image processing tool. This has also given me a great experience to learn many new things regarding GUI, installing libs, using Ubuntu, Qt Designer.

FUTURE WORK

I was not able to use all the GUI Components, I took time for me to learn PyQt5, OpenCV and work on GUI. I would like to work on this toolbox further to develop it in more sophisticated way and including more features like a geometrical tools, motion capture, object detection tools, water

shed algorithm (used for Segmentation in Image Processing), digit recognition (for MNIST Data set) and deep learning tools as we do in commercial toolboxes.

ACKNOWLEDGEMENT

I sincerely thank my supervisor, Dr. Abd El Rahman Shabayek for motivating us to work on the listed tasks or tools. The cookbook for OpenCV "OpenCV Computer Vision with Python has helped me a lot to complete this toolbox.

This project enormously built a Research interest in me in Computer Vision.

References

[1] Mr. Gopi Krishna's GitHub repositories: https://github.com/gopi231091/Development-of-Image-Processing-and-Computer-Vision-Toolbox-using-C-and-OpenCV (for Code explanations)

[2] OpenCV , official website:: https://docs.opencv.org/3.4.2/

[3] OpenCV Computer Vision with Python Cookbook::

http://chamilo1.grenet.fr/ujf/courses/FAMILIARISATIONAVECPYTHONSUITEANACON/document/OpenCV_Computer_Vision_with_Python_-eBook-.pdf

[4] OpenCV forums :: for Definition in Code modules

[5] Stackoverflow forum :: for solving installations

[6] Jie Jann's Youtube explanation:: https://www.youtube.com/watch?v=tZNnXL7cW6o

[7] Mr.Meldrick REIMMER Toolbox explanation: https://github.com/brown4eva/Computer-Vision-ToolBox-OpenCV-with-python/blob/master/ComputerVisionToolBox_Python/MeldrickREIMMER_ToolBoxReport.pdf

[8] GUI with PyQt5 : https://www.youtube.com/watch?v=ksW59gYEl6Q

[9] Install OpenCV on Ubuntu:: https://youtu.be/DYTfwThePBw

[10] Qt Tutorials for Beginners:
https://youtu.be/5JVLO8yBMXA?list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA

[11] Connecting PyQt5 with Qt Designer: https://www.youtube.com/watch?v=pNr0-63ukaE

[10] OpenCV, Wikipedia