Visual Perception

Computer Vision Tool Box

May 8, 2018

Submitted by : Meldrick F. REIMMER
Supervised by : Abd El Rahman SHABAYEK

Introduction

This is a report for my course module in Visual Perception. In this report, you will get the manual documentation of two implementations of computer vision toolbox. The toolbox was implemented with Matlab and in Jupyter using OpenCV. In addition to this report, you will find my source codes for both of my implementations on my GitHub repository and also a video in which I gave a brief explanation on some tasks I performed and the results of my implementations.

Objectives

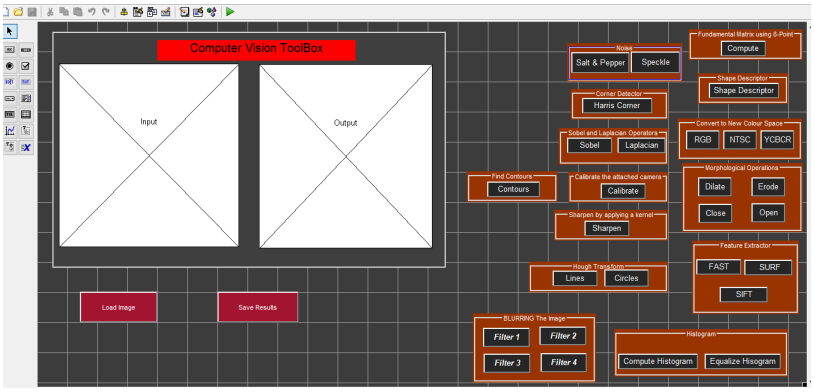
The main goal of the project was to get used to Matlab being a powerful tool for computer vision by testing and implementing its computer vision algorithm and also learn on how to use OpenCV, which also is one powerful tool in computer vision. Having to familiarized with the following platforms, I then continued to accomplish the tasks given by my supervisor. These are :

1. To develop a Computer Vision ToolBox using Matlab to perform same tasks just as libraries do.
2. To develop another Computer Vision ToolBox using OpenCV with any programming language other than Matlab.
3. Give a user-friendly environment with our GUI for easy use and easy access to task aske

Developement Using Matlab.

Graphical User Interface (GUI)

My GUI consists of 30 push buttons and two axes in which I used to display the input given and output. I managed to give it a user-friendly environment and highlighted the buttons with texts to show what each does. Each button as a callback function which executes after being pressed on. The functions are tasks which were given. They are general image processing tools life loading an image, saving an image, converting the image to grayscale, converting the image to colorspaces being HSV, NTSC, YCBCR, sharpening of the image, performing morphological operations like dilate, erode, close and Open. There are many other functions given. You can see them in my GUI image below.



Tasks Given

In accordance with each button, it performs specific tasks given to the callback function. Once you click on the button you will execute the task given. To start you will first need to load an image or video in which you can choose any of the functions in which you wish to perform. Will demonstrate some procedures below.

Add nosie to your image.

To do that you will need to follow this steps:

1. You first need to give an input (Video or Image)and this is done by clicking on the load image.

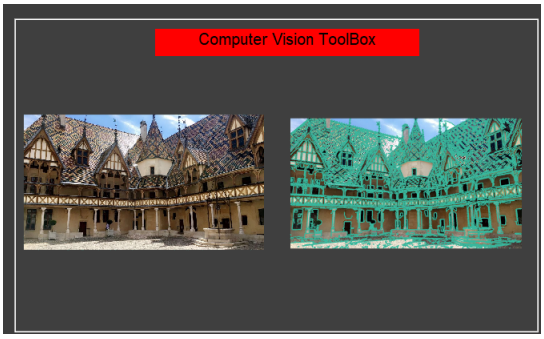
2. After clicking on the type of noise in which you want to perform on the image. There are two options. One being Salt&Pepper and the other Speckle noise.

3. Then in getting your results, you can click on the save button to save the image gotten at any location on your computer.

Below are the results for performing Salt&Pepper and Speckle noise on my image.

Finding the Contours.

Another task this program does is finding the contour of an image. This is by performing the steps as described above and clicking on the contour button to perform the task by extracting their boundary. Below is the results.



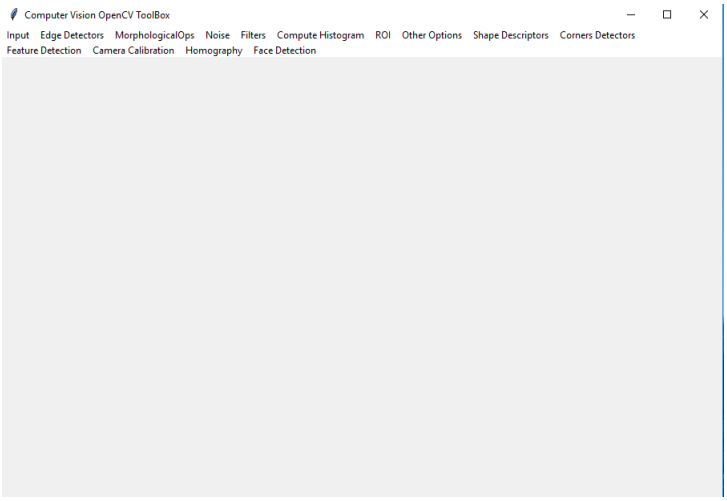
Other Tasks:

There are many other tasks this program can perform. You can try your hands on them by just following the link in the references. It is fun to use and gives a nice result. You can try them on any formats of images and get great results.

Development using OpenCV with Python.

Graphical User Interface (GUI)

My GUI was developed in Jupyter notebook using tkinker. It had 14 menu-bars under each menu-bar as a function in which it performs certain tasks given to it. It is user-friendly and gives very broad results. It display is very large since I intended not to give a label to show the output instead I used OpenCV to show results. I gave the link to my video in which I explained how each function is executed and the steps needed to follow. This compared to Matlab GUI does not restrict the image in displaying and shows the image as wide as it is. Below is an image of my interface.



Tasks Given

For this part of the project, every task was done using OpenCV. Using Jupyter notebook I imported the OpenCV library and used it in performing the task given. You will see in the source code given my implementations and each task and calling of the functions needed to achieve the task given. Examples of the tasks given were:

Performing Edge Detection.

To do this you will need to perform the following steps:

1. Load an image to be your input after.

2. Click on the function you want to perform. Here is edge detection so we click on the Edge-detectors menubar and we get three functions under it we can choose any and once we click on it we get the results.

3. The results in automatically saved in a folder in which we assigned to called "Saved Photos"

Above is the results gotten from performing the steps as described above. The first image is the original image followed by Laplacian and Sobel.

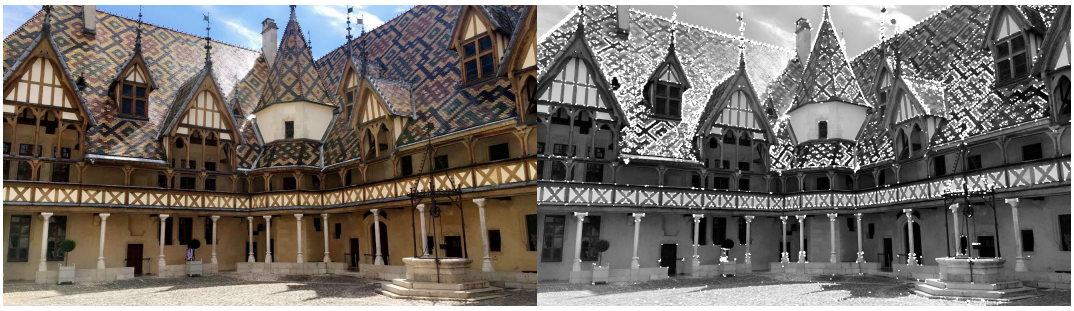
Performing Morphological Operations

To do this you just need to follow the same steps above and click on the morphological operations menubar and get the functions under it and click on the one you want.

Above is the results of performing morphological operations. The first image is the original followed by the open, close and erosion.

Extracting Corners

To extract corners I used the Harris Corner detectors. I loaded an image of a building just as the one I used to detect the contour using Matlab. You just need to follow the same steps and after click on Corner Detectors menubar and under it click on the Harris Function.



Above is the results of the image detecting corners using Harris Corner Detection.

Other Tasks.

Other tasks can be performed using this program. These tasks I performed are just to show you on how the program works. You can run the program on the link in the reference or on my GitHub page and also watch the videos to see how others tasks are done

Conclusion

I will say given much time I could have explored many tasks using OpenCV. It was very easy to use and had many documentations online to aid you when you get errors. I learned much and gain new knowledge. I was also able to advance my knowledge in Matlab and discovered new things. In general, I will continue to work more on both languages and add C++ also with OpenCV, since its also a good way of applying OpenCV.

References

- GitHub link to get source codes : `https://github.com/brown4eva?tab=repositories`
- My Youtube Video to show more tasks being done for the OpenCV with Jupyter : `https://youtu.be/GtZa9z9C3EM`
- My Youtube Video to show more tasks being done for the Matlab Development : `https://youtu.be/WtQY34Ftfgk`