Visual Perception - Toolbox Project

BOUDISSA Selma - MSCV 1 student

Deadline 07/05/2018

Contents

Acknowledgement

I wanted to say thank you for the people who help me and who I work with which are Meldrick.F REIMMER, Mohamed ALI, Danie SONIZARA, Kévin DESCHARRIERES and Karim BOSTROS.

Introduction

For this Visual Perception course module we had a project which consist to create a toolbox with differents function for image processing, in two different environnement in Matlab and one in Python using Opencv.

In this report some guidelines will be given for the explanaiton and the usage of both toolbox.

Part I
Matlab Toolbox

1 Graphical User Interface (GUI)

1.1 General

First step of this project I decide to create my GUI using the proposed matlab tool calling “guide” in the command window.

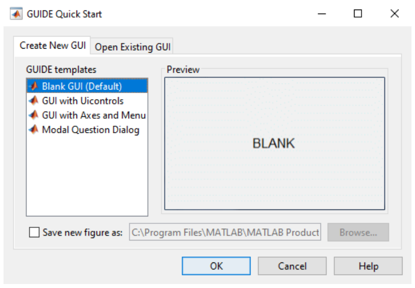


Figure 1: Guide tool from matlab

Explanation: So as you can see in the Figure 1, this appear when you call “guide” in matlab command window, you have the choice to create your own GUI or open an existing GUI.

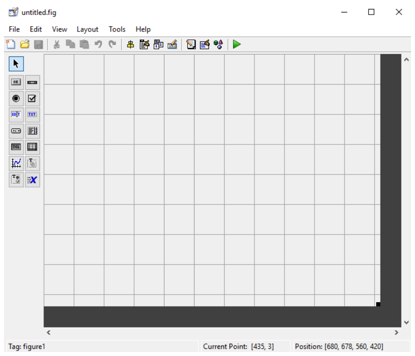

untitled.fig
File Edit View Layout Tools Help
Tag: figure1
Current Point: [435, 3]
Position: [680, 678, 560, 420]

Figure 2: Empty GUI

So on the left of the Figure 2 you have all the tools proposed: pushbutton, slider, checkbox, ... depending on what you need. To design your GUI is pretty easy you just need to choose the tools and drag them on the right. You are free to arrange and organize it the way you want and also choose the size of your window.

Once you are done you just save your gui and this will automatically generate a file '.fig' and your matlab file, you can now start coding.

1.2 My GUI

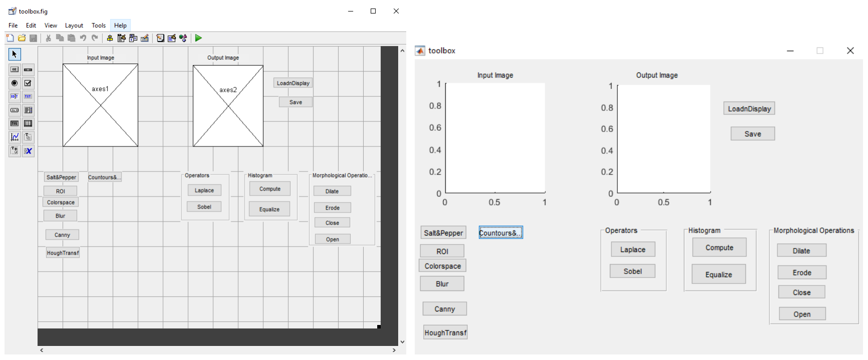


Figure 3: My gui

In the Figure 3 is the design of my own GUI for my toolbox,on the left the modifiable gui and the right the final result. As you can see I use pushbutton but also axes to display my image input and ouput. I organize some function by class using some panel to kind of organize the group of function that are related for example I create a panel named 'Histogram' where I put to pushbutton 'Equalize' and 'Compute'.

2 Explanation about the functions

I will now give you example of the result obtain for few functions.

2.1 Salt and pepper noise

For this task I use 'imnoise' proposed by the environnement which I specify the type of noise. In the Figure 4 you can see the result on the output image (on the right).

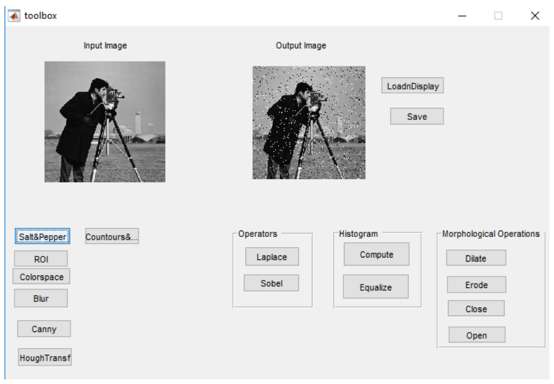


Figure 4: Salt and Pepper noise result

2.2 Histogram

The goal was to compute and equalize the histogram, I implemented this two function in two different pushbutton using also matlab function which are 'imhist' and histeq'. In the Figure 5 you can see the result.

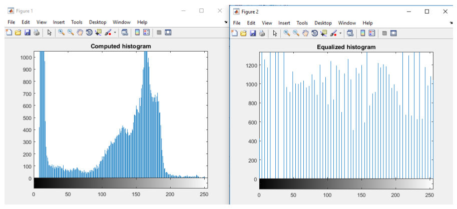


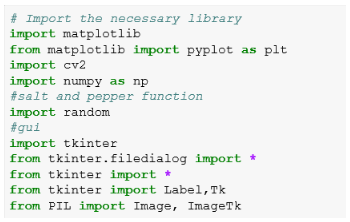
Figure 5: Histogram computed and equalize results

Part II
Python using opencv toolbox

For the second part of the project I have used python with opencv, contrary to the matlab toolbox this time, I fist right my code on jupyter-notebook and then using 'Tkinter' create my GUI.

3 Library

In the first time I imported the necessary library to be able to use all the opencv function.



```
# Import the necessary library
import matplotlib
from matplotlib import pyplot as plt
import cv2
import numpy as np
#salt and pepper function
import random
#gui
import tkinter
from tkinter.filedialog import *
from tkinter import *
from tkinter import Label,Tk
from PIL import Image, ImageTk
```

Figure 6: Library from python using opencv

- import cv2: able the usage of all the available function proposed by opencv
- import tkinter: this useful for the GUI
- import matplolib: useful to plot something with in the jupyter-notebook file

4 Code

I define all the function I will need for my toolbox, using the help provide by opencv.org. I commented my code and put the reference of each website giving explanation on how to do it.

Example:

1. Convertion to the colorspace: I implemented different typer of colorspace like YCrCb by using the command : " ycrcb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)"
2. Morphological operation like Erosion: img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

5 GUI

As jupyter-notebook compare to matlab as not is own gui I had to import 'Tkinter', I have created this one on the last part of my code.

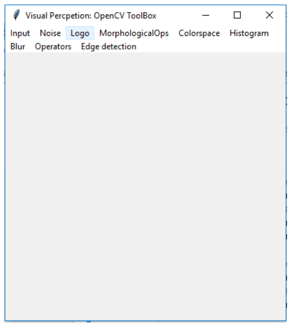


Figure 7: GUI on jupyter-notebook

Result: In the Figue 8-9 is the resulted images from the GUI using opencv function, the difference with matlab is that the image are diplayed outside the GUI, you can just call the function and the image will be shown in a window.

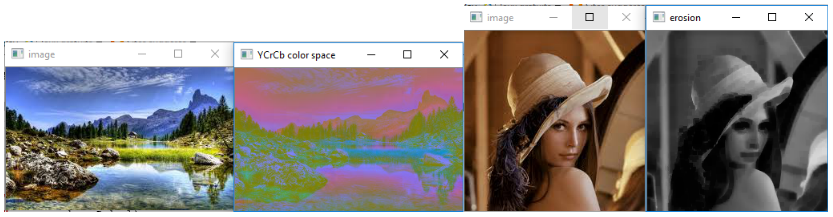


Figure 8: Color space using YCrCb result (Landscape)

Figure 9: Morphological operation using erosion (Lena)

Conclusion

To illustrate this few guidelines I invite you to check the link below of the short youtube video recording my screen while showing both GUI.

- opencv: https://www.youtube.com/watch?v=MomaFsaiyg8
- matlab: https://www.youtube.com/watch?v=sCbmk_mTqt0