VIRTUAL PERCEPTION
Project Work

Computer Vision ToolBox

Sadiq MACAULAY

May 21, 2019

INTRODUCTION

Computer Vision is the science that aims to give a similar capability to a machine or computers. Computer vision, helps to obtain relevant information from images and make decisions based on these information. In other words, computer vision is making the computer see as humans do. The Computer vision Science is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos[3].
OpenCV is an open source C++ and Python library for image processing and computer vision, It is a library mainly aimed at real time processing and has several hundreds of inbuilt functions to implement image processing and computer vision algorithms for developing advanced computer vision applications and is also highly suitable for research purposes[2].
This Project work is focused on creating a Computer Vision toolbox with Opencv library using Qt-Creator widget application. A GUI has been created where some computer vision and image processing operations are implemented and the respective output of the operation is displayed in real time. A description of all possible operations as instructed in on the project description are discussed in the consequent sections

OVERVIEW

The implementation of the toolbox was intended to be a user friendly and interactive display of input image, video or camera and output the respective possible operation on the input in real time. The implementation has been carried out using OpenCv library in C++. However, for the execution of the program, an extra module `https://github.com/opencv/opencv_contrib` would have to built and linked to the project for the functionality of some operations. A description of the interface is shown on figure: 0.1.

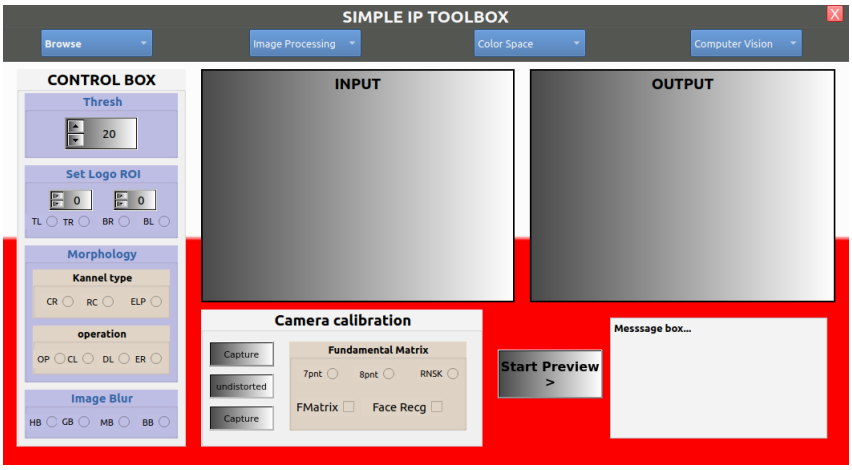

Figure 0.1: GUI-panel

A description and list of the required task for the project can be refereed to in [1]. Basically for this implementation, the task is divided into two parts which are the basic image processing operations and the Computer Vision operations through a video, camera and image inputs, as indicated on the figure. Each operation are independent of any other one except for a few (displaying a logo, histogram of image). It would be required to stop previous operation before preceding to next operations in order to obtain the required output of the operation at hand. A brief structure of the pipeline is described in the figure: 0.2.

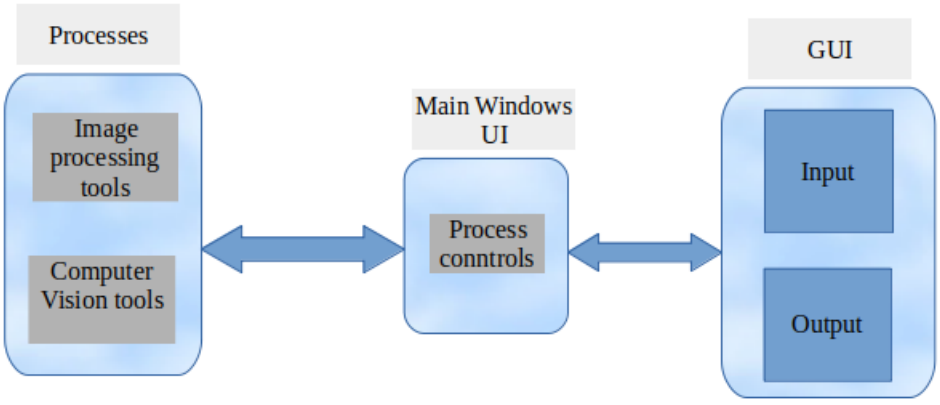

Figure 0.2: software pipeline

Loading Data to the Environment

Image data can be loaded into the environment by clicking on the Browse list-button (figure: 0.3) on the top left corners which provides drops down list of sources from which input image frame data can be fed into the environment. By selecting any of the option an output is displayed on the input axis as well as on the output axis on the left side of the GUI. The Preview push-button would need to be clicked on as well.

- **Video:** This provide the option to load video files from a directory. Video frames are read into the environment.
- **Image:** Image is also be read from a directory by selecting the Image on the source drop down list-button.
- **Cam:** This opens the default camera being the inbuilt camera to the computer and adds each frame to the environment.

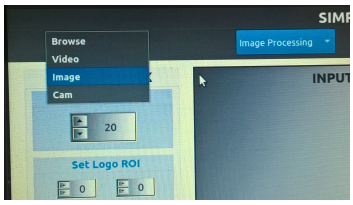

Figure 0.3: Browse list-button

When the Start Preview is toggled, by default in input image is directly displayed on the output axis until there is a required operation is selected.

Basic Image Processing Operations

A some image processing functions are implemented in this section whose output result are delivered in real time. each time an operation is called, the corresponding outputs are displayed on the output axis on the right with necessary data on the text box below. in order to undo this operation, the options has to be selected the second time and the output image frame is restored. The list of possible operations here includes:

- Adding Salt and pepper
- Showing Logo on output image in a desired region of interest
- changing the color space of an input frame
- Histogram of Image
- Histogram equalization of an image
- Morphological operations(Dilate,Erode,Open,Close)
- Image blurring

- Applying Sobel and Laplacian operators
- Canny edge detection
- Extracting lines and Circles using Hough transform
- Drawing contours of connected objects
- Applying component shape descriptor
- Extract corers using Harris corner

ADDING SALT AND PEPPER NOISE

This is implemented by generating random noise on a image. In order to do this, you have to select the **Tools** list-button, and from the list of the drop down you have, select **Salt Pepper Noise**.

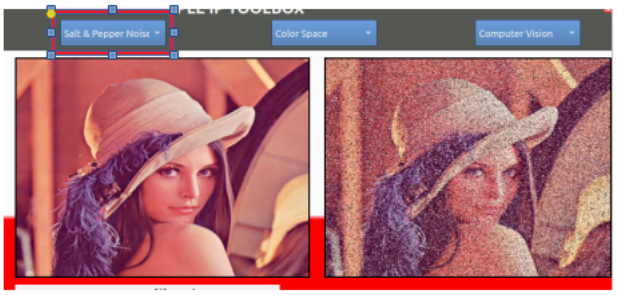

Figure 0.4: Salt and Pepper Noise

SHOW LOGO

This option calls a function that overlays a smaller image on the main image in the output axis by selection the region of interest on the **set Logo ROI panel** in (figure: 0.5). **TL, TR, BR, and BL** represent top left, top right, bottom left and bottom right respectively and also the values of the X and Y axis of the logo position can be written manually from the Spin box on the panel as well.

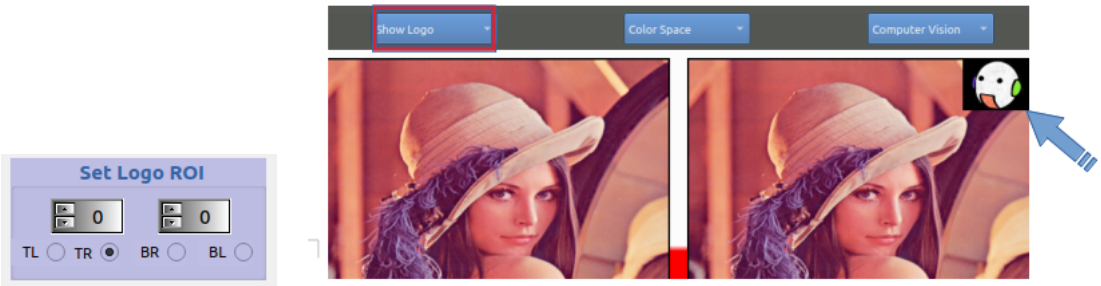

Figure 0.5: Setting Logo Position

COLOR SPACE

The next list-box after the tools list-box is the the list-box for color space options for the input image, by default, in OpenCv color images are read in BRG and with the use of this options the color space can be modified to RGB, LAB, YCrCb, HSV as well as Gray and Binary image.

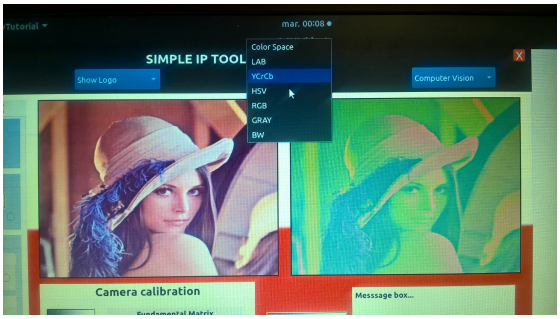

Figure 0.6: Colour Space

HISTOGRAM OF IMAGE

On the **Image Processing list-button** is an option that compute the Histogram of an Image from 0 to 256 pixel intensity value of the image. This option calls a function that compute histogram using Opencv functions. For a colored image, the intensity level are computed through the process of first separating the color channels, converting them to gray scale and then computing the histogram of their respective pixel intensity.

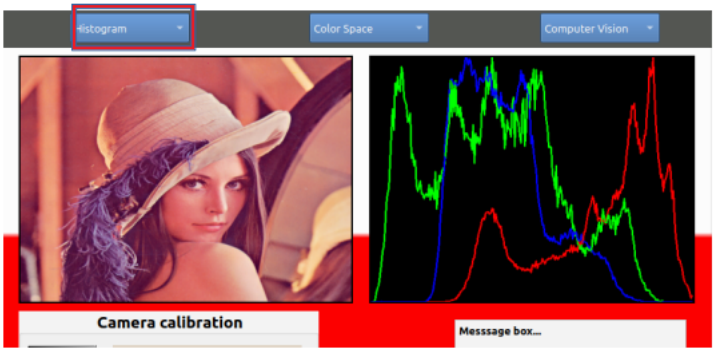

Figure 0.7: Histogram of colored image Image

HIST-EQUALIZATION

This option is computed to perform an enhancement on the input frame by stretching the histogram of the image pixel values such that their intensity are evenly distributed over the range of 0 to 255 pixel intensity, thereby making the dark region of the image brighter.

MORPHOLOGICAL

This consist of a set of operation including **Dilate, Erode, Open, Close** that has been computed to operate over a range of thresh hold values and kannel size as well as the type of kernel required. These set of options can be selected from the **Morphology** panel on the right of the input axis.
The **Kernel type** includes **CR, RC, BR and BL** that represents Cross, rectangle and ellipse shaped kernel respectively while on the operation panel are the **OP, CL, DL, and ER** which also represents the Opening, Closing, Dilatate and Erode morphological operation. The size of the Kernel can as well be modified by using the set thresh spin-box at the top of the **CONTROL BOX**.

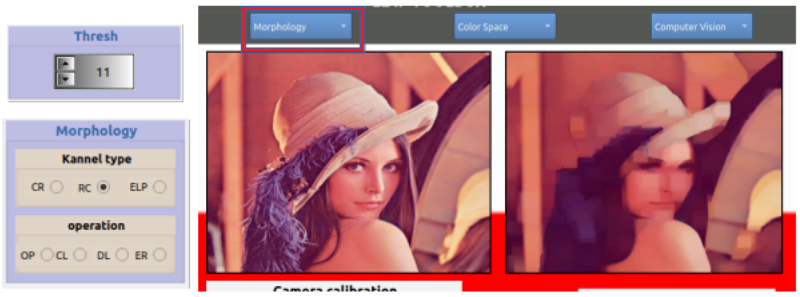

Figure 0.8: Morphological operation on input frames

BLUR

This implements an Opencv function that applies blur on the input image with the required thresh value that can be selected from the Thresh spin-box. The OpenCv library provide function that implement the different blur algorithm including Median, Gaussian, Bilateral and Homogeneous blur to an in put image. By selecting this option, a set of these operation is implemented by checking the different operation on the **Image Blur panel**. Also, the thresh can be controlled by changing the value of the **Thresh** spin-box.

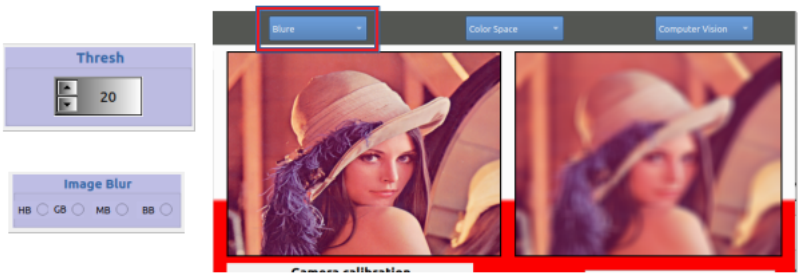

Figure 0.9: Image blur

OTEHER OPERATIONS

Every other operation under the image processing options follows the same regular pattern of selecting the option and the corresponding output is displayed on the output axis and with the thresh spin-box, adjustment can be made on output image.

Figure 0.10: Thresh spin-box

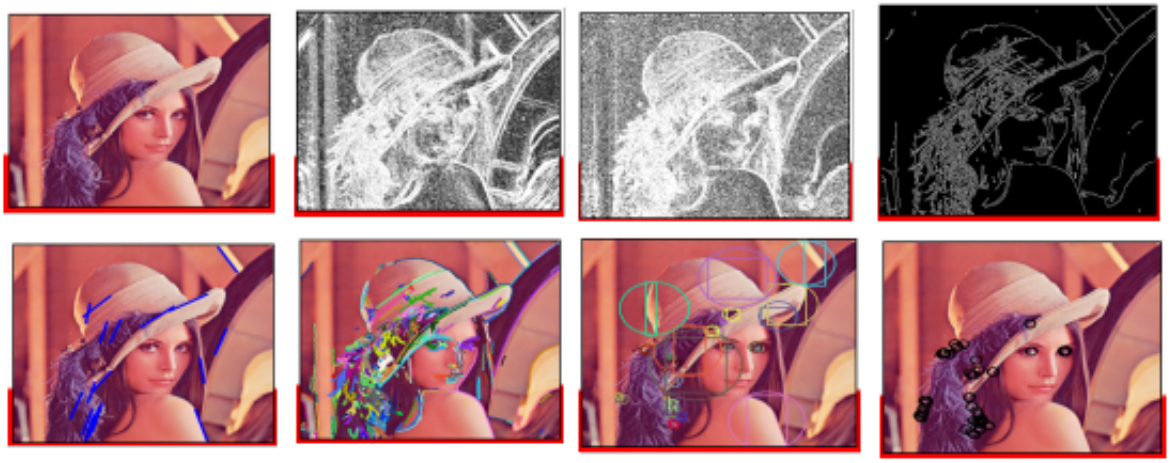

Figure 0.11: from the top right: original image, Sobel, Laplacian, Canny, Hough Lines, Contours, Shape Descriptor, and Harris Cornner detector

COMPUTER VISION OPERATIONS

Operations carried out in this part mostly relates to vision system such as features extraction, calibration, computing the parameters of a camera, point matching, image stitching and so on.

most of this operation will involve computing properties between images for their implementations. Some of the processes are explained in the following subsection.

IMAGE MATCHING

This involves finding salient features between different images, it is computed by finding the relevant features on the individual images and matching the similar point, the method of finding features are dependent on the application, the different algorithm includes,SURF, FAST and SIFT which are also implemented in the toolbox. Selecting this option will require two different images. the dialogue box helps to select a second image to the image already available in the toolbox environment. An output image of with the salient points and matching lines of the two image is shown on the output axis.

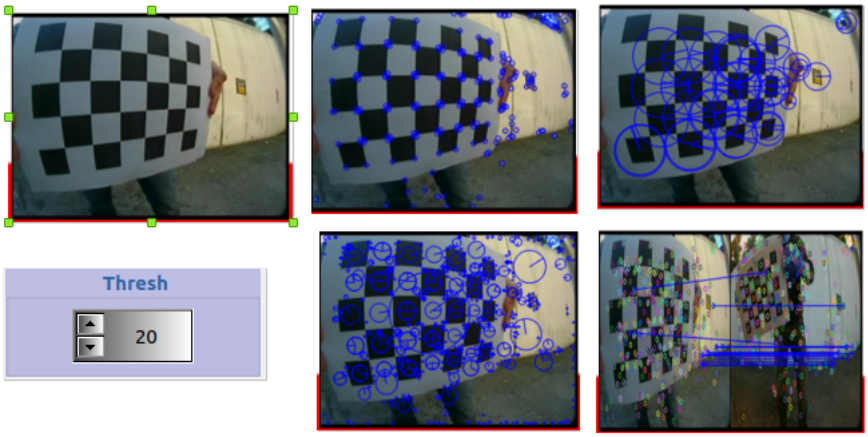

Figure 0.12: Point detection and matching different algorithm; starting from the top left is the input image, FAST, SURF, control spin-box, SIFT and Matched Image

CAMERA CALIBRATION

This follow the following processes

- Select from the **Source** list-button, **Cam**
- Capture images of a check-board with known size of the box with different forms of distortion. The capture button is on the camera calibration panel this save images on the current frame into a folder
- Select the **Camera Calibration** option on the **Compute Vision** list-button.
- from the opened directory, select a distorted image to be corrected

Calibration is then completed while the points of the squares are highlighted on each images that were previously saved to the calibration folder. The Distorted image that was set is also corrected using the data of the calibration and displayed in sequence and the.

Note: calibration is done more efficiently at the beginning of the launch of the program and not in a stream of operations.

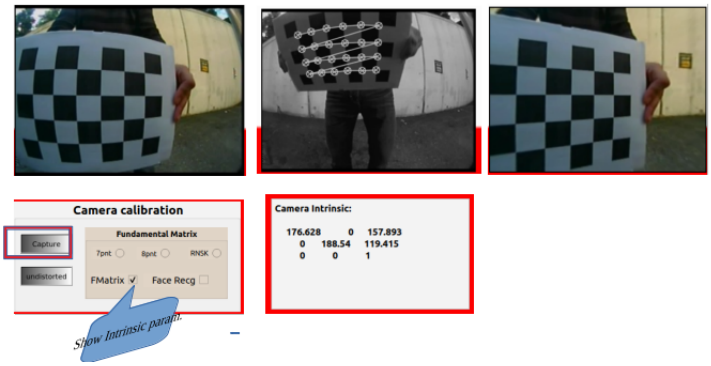

Figure 0.13: from top left: distorted image, points on calibration pattern, corrected distortion, panel with capture button for calibration and check-box to display calibration param, and camera intrinsic displayed on msg box

FUNDAMENTAL MATRIX AND EPIPOLAR LINES

This also require tow different images from the camera the first been the initial image loaded on the in put while the second will be required when the option is selected from the computer vision list-button. the epipolar lines are drawn on the output image and the fundamental matrix is as well computed simultaneously, however, the FM check box will need to me selected to view the data on the message box.

Note: If FMatrix check-box is already checked you will have to uncheck and check back to display the fundamental matrix on the message box.

Figure 0.14: Showing the three different algorithm to compute the epipoles and fundamental matrix, starting from the left is the 7point, 8point, and RANSAC with their corresponding computed fundamental matrix right below them

COMPUTING HOMOGRAPHY

Homography is computed from two similar images with different rotation and translation. This is achieved with feature point matching in the two images to produce the that can be used to render the correct perspective of the image, This can be determined by selecting the initial image and its corresponding image pair when the option is selected from the computer vision list-button. An output of the images are displayed with the correlated point for computing homograph and the blue box represent the estimated rotation and translation on the image. When the Fmatrix check box is check an output of the homography matrix is displayed on the message box below the output axis.

Note: If FMatrix check-box is already checked you will have to uncheck and check back to display the Homography matrix on the message box.

Figure 0.15: Homography between two images of different orientation. green box shows the orientation of the second image with respect to the first

IMAGE STITCHING

There are no display on the output axis when images provided cannot be stitched.

Figure 0.16: firs image on the fight, following is the second image and the last on the right is the output of the stitching of both images.

FACE DETECTION

The face detection includes both deep learning and the use of cascade classifiers which uses pre-trained data from on OpenCV git repository to detect the human face and eyes as well with image and video. A rectangle or box is drawn on the detected face in the image or video.

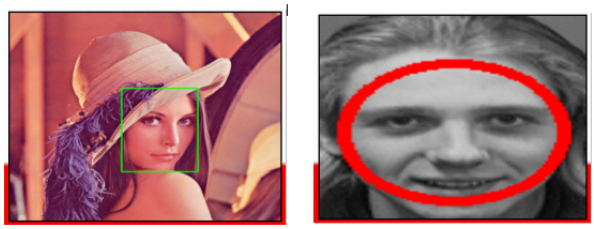

Figure 0.17: Face detection using DNN and Cascade classifier

FACE RECOGNITION

The face recognition options which include computation with Principal Component Analysis and Local Binary Pattern with data set from the ATT face database. An index label file is created in the project directory of .csv file extension. The computation is only limited to the pre-trained data in the project, and not so robust to recognize any face, A face label is displayed on the message box to show the recognized face in the input video file or image.

Figure 0.18: Face recognition using PCA (on the left) and with LBP (on the right) with their respective predicted label below each image.

CONCLUSION

Following the implementation of the required task for the project work, about 98% of the task has been implemented efficiently in C++ despite the serious challenges in developing the project structure, and the troubles linking additional libraries to the project, yet a remarkable experience have been achieved over the period of the implementation and a great deal of skills developed with Opencv libraries for image processing, Computer vision.
I would like continue with the development of this software toward actualizing its functionality for a real life application. Also, I am considering to implement this in python which is presumed to be a much easier programming language for this implementation.

ACKNOWLEDGEMENT

My sincere appreciations to the Professor "Rahman SHABAYEK" for his advice and referral to the "OpenCv 2 Computer Vision application programming Cook book", Stack Overflow, OpenCv official documentation cite also, a former Mscv student Gopi Krisha for sharing his materials and experiences for references. All this has greatly contributed to my experience and progress in this project thus far.

REFERENCES

[1] Abd El Rahman SHABAYEK. *Visual Perception Course work*. SnT, SIGCOM, Computer Vision Group, University of Luxembourg, April 12, 2019.

[2] OpenCv Documentations»Tutorials. *https://docs.opencv.org*. .

[3] Gopi Krisha: Project work on Comuperoolbox, `https://github.com/gopi231091/`