# Image Processing and Acquisition using Python.

Aniruddha R. Kulkarni G-22 11910240 , Anurag Kulkarni G-23 11910287, Darshan Kulkarni G-25 11910168, Aniruddha J. Kulkarni G-21 11910038, Advait Kondra G-12 11911419.
Department of Engineering, Sciences and Humanities (DESH)

***Abstract*— Image processing is the method to process a certain image using an algorithm to get the desired results. The demand for image processing technologies is increasing day by day. Tedious work can easily be processed and be acted upon by using image processing algorithms. Image scanning, targeting objects based on their features such as color, shape, area of the screen covered can be easily done. Softwares such as image scanners use image processing at the core. Image processing algorithms allow the computer to actually what it is seeing and with precise instructions also help the computer to execute certain actions. How humans see or perceive objects in the same way image processing algorithms can help the computer to perceive what exactly it is seeing.**

***Keywords*—— *Image processing, Image scanners, algorithms.***

## I. INTRODUCTION

In today's times seeing is believing and the same applies to computers too. This is where computer vision comes into picture. When we look at a picture and if we do not understand what's in it we simply take the picture and upload it to Google Lens. That's the easy part however at the backend of it there are several other processing technologies involved and image processing plays a major role in it. Image processing algorithms are the ones that help the Google Lens the results that it gets.

Image processing is an umbrella term for many functions that analyze images or convert one representation of an image into another. In imaging science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image.

This paper is based on the features of image processing and how it can be used to enhance images and extract required information from it.

## II. IMPLEMENTATION DETAILS

The project is based on Computer Vision that has been implemented in Python. The project basically contains different inbuilt functions in Computer Vision which can be further used for many tasks. Basically the project is based on how Computer Vision technologies enhance image processing softwares to get the best results. The project also states the different areas in which Computer Vision Algorithms are used and are being used. Object detection, Object tracking, Facial detection, Facial recognition, shape detection [1], colour thresholding [2] techniques which can be used to find objects from a cluster of objects. Further the project also states how Computer Vision Algorithms can be used in the field of Deep Learning [3] wherein custom classes that have been trained into a neural network [4] can be identified.

Softwares/Languages used:

1. Language for the code: Python
   - Python: It is the most widely used programming language.
   - The reason Python is used completely in this project is because of the easy syntax it provides which makes the code concise and readable to the user.
   - It also has a very huge database of modules that can be imported into the codes to make efficient working codes with proper logic implemented.
2. IDLE/Coding platform used: Any IDLE that supports Python and its libraries can be used for coding in Python.
3. Modules to be imported:
   - cv2/OpenCV: OpenCV is a library of programming functions mainly aimed at real-time computer vision. It is a computer vision library that can be programmed in both Python and C++. Using OpenCV with Python the code is easy to be understood thanks to Python's easy to understand syntax.
   - Numpy/Numerical Python: Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

4. PIL/Pillow: Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.
5. PyQt5: It is another GUI building module in Python however it can also be programmed to work with C++. A GUI in Python using PyQt5 is an easy process. The frontend part of the GUI can be designed into the QT Designer which can be converted into a Python file so that the basic code for the layout of the GUI is already prepared. The only thing that remains is how we code to get the required functionality.

   Inbuilt functions in OpenCV which can be used to obtain the best results are used.

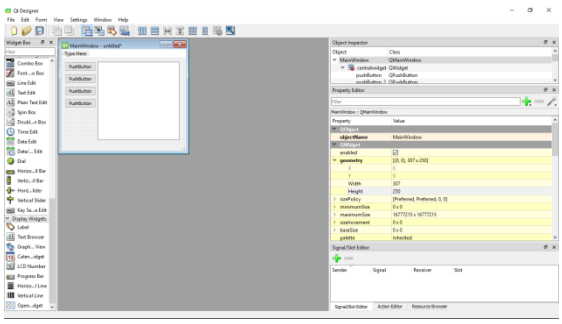

Fig 1: Frontend GUI designing in QT Designer.

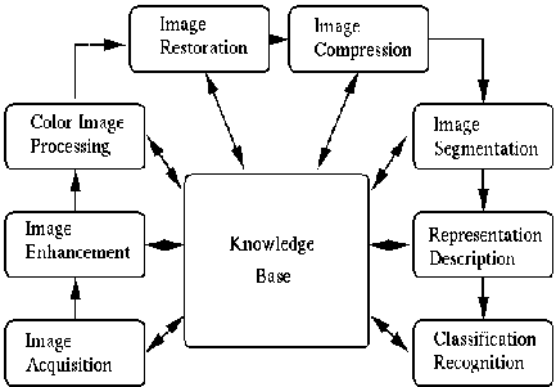


Fig. 2: Frontend GUI designing in Qt Designer.

```python
def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(418, 230)
    MainWindow.move(0,0)
    MainWindow.setStyleSheet(" background-color:rgb(0, 85, 127) ;")
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.pushButton = QtWidgets.QPushButton(self.centralwidget)
    self.pushButton.setGeometry(QtCore.QRect(0, 0, 150, 50))
    self.pushButton.setObjectName("pushButton")
    self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
    self.pushButton_2.setGeometry(QtCore.QRect(0, 50, 150, 50))
    self.pushButton_2.setObjectName("pushButton_2")
    self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
    self.pushButton_3.setGeometry(QtCore.QRect(0, 100, 150, 50))
    self.pushButton_3.setObjectName("pushButton_3")
    self.pushButton_4 = QtWidgets.QPushButton(self.centralwidget)
    self.pushButton_4.setGeometry(QtCore.QRect(0, 150, 150, 50))
```

Fig 3: Code for the basic Outline of GUI using PyQt5.

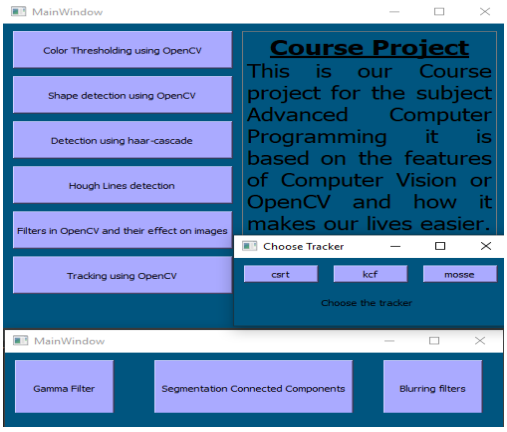


Fig. 4: Basic Outline of GUI created to execute the codes.

```python
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtWidgets import QPlainTextEdit
import tkinter as tk
import cv2
from PIL import Image, ImageTk
import numpy as np
import imutils
import settings
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout
from PyQt5 import QtCore, QtGui, QtWidgets
from OtherWindow import Ui_OtherWindow
from shape import Ui_shape
```

Fig. 5: Different modules imported.

```
def on_click(self):
    print("Starting Code for Color Thresholding")
    exec(open("ct.py").read())
def on_click2(self):
    print("Starting Shape Detection for the given Input image")
    exec(open("detect_shapes.py").read())
def on_click3(self):
    print("Starting Face Detection using Haar-Cascade")
    exec(open("face.py").read())
def on_click4(self):
    print("Starting Striaght line Detection using Hough-Lines")
    import cv2 as cv
    import numpy as np
    import matplotlib.pyplot as plt

    def do_canny(frame):
        gray = cv.cvtColor(frame, cv.COLOR_RGB2GRAY)
        blur = cv.GaussianBlur(gray, (5, 5), 0)
        canny = cv.Canny(blur, 50, 150)
        return canny
```

Fig. 6: This image shows some of the code that is used in our project codes.

```
class Ui_MainWindow(object):
    def openWin(self):
        self.window=QtWidgets.QMainWindow()
        self.ui=Ui_shape()
        self.ui.setupUi(self.window)
        self.window.show()
    def openWindow(self):
        self.window=QtWidgets.QMainWindow()
        self.ui= Ui_OtherWindow()
        self.ui.setupUi(self.window)
        self.window.show()
```

Fig 7: Code written to link different GUI windows.

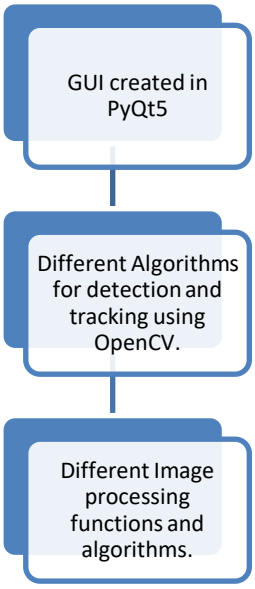


Fig 8: Flowchart of the Project.

- Steps to create the GUI that be used access all of the codes written.
  1. The GUI is written in Python.
  2. It is based on Modules installed in Python namely PyQt5.
  3. PyQt5 is used to program the backend part of the GUI.
  4. Qt Designer is used to create the frontend part of the GUI.
  5. The ui file from the Qt designer is converted into a python file.
  6. This python file can then be programmed with classes and events to make it user friendly.
  7. One GUI window can be connected to a different GUI window by using each other as modules and one file can be accessed from the other.
  8. Different python files each containing a part of the GUI are created and are connected using multithreading to keep the codes precise and easy to read.

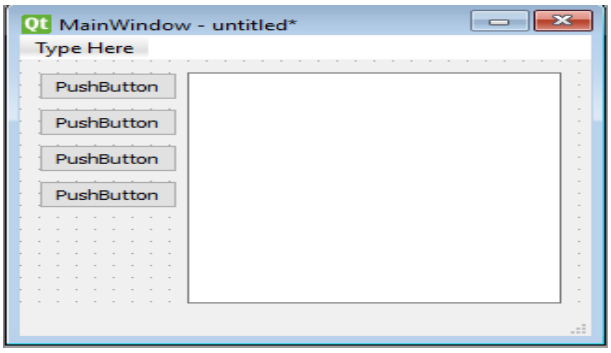


Fig 9: Frontend GUI created in QT designer.

```
def on_click(self):
    print("Executing csrt Tracking")
    exec(open("csrt.py").read())
def on_click1(self):
    print("Executing kcf Tracking")
    exec(open("kcf.py").read())

def on_click2(self):
    print("Executing mosse Tracking")
    exec(open("mosse.py").read())
```

Fig 10: Functions assigned to be executed on on_click button event.

## III. Results and Discussion

Thus using Computer Vision we can use image processing to perform different tasks like facial detection, facial recognition, object detection using haar-cascades, object detection using deep learning etc. we can also perform colour thresholding tasks which further help in shape detection [5] and filtering of object in a frame.

Some of the results obtained are as follows:

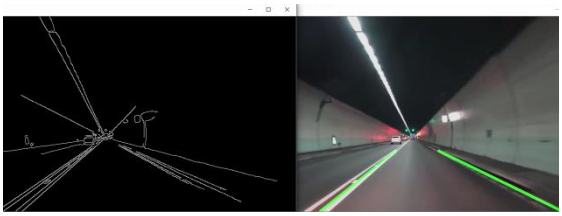

Fig. 11: Lane detector using Hough lines[6] which is an inbuilt function in OpenCV.

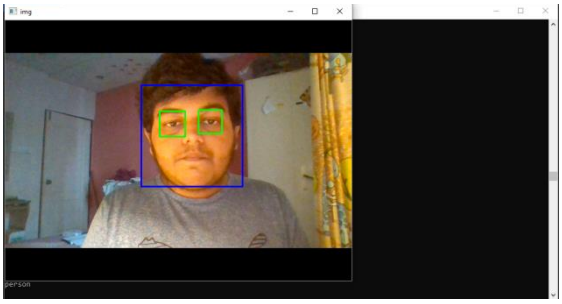

Fig. 12: Face detection [7] which can be further implemented to perform facial recognition. This code uses different haar-cascades to perform the detection.

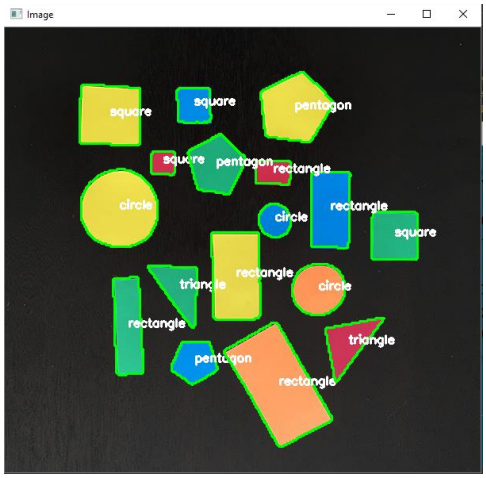


Fig 13: Shape Detection using Contour detection and corner detection.

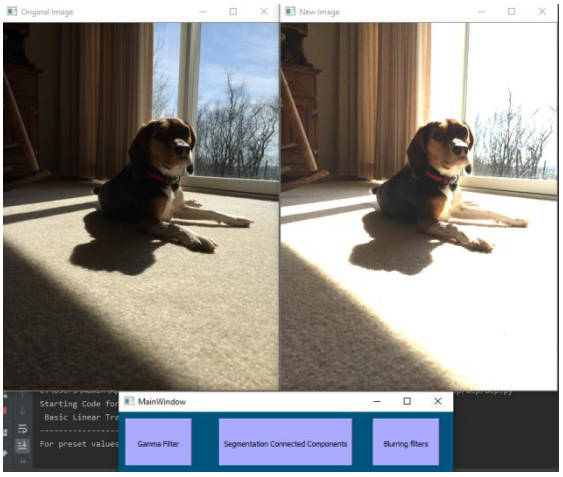


Fig 14: Using Gamma Filters on a dark image to illuminate it.

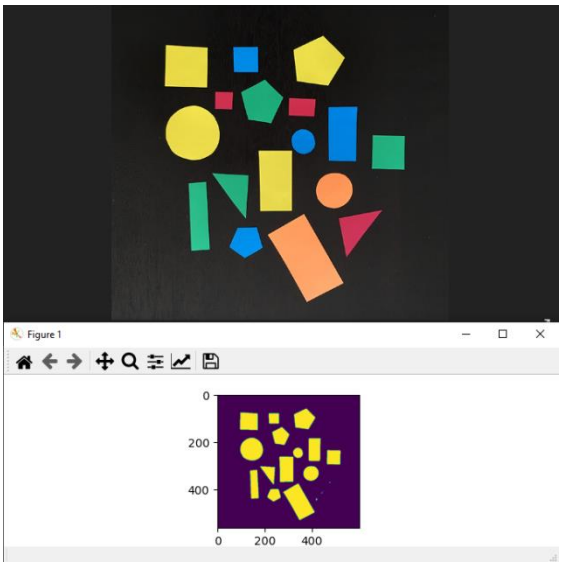


Fig 15: Segmentation using Connected Component Analysis.

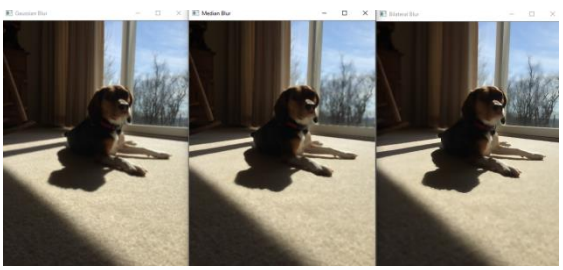

Fig 16: Image Smoothening [8] using Gaussian Blur, Median Blur, Bilateral Smoothening.

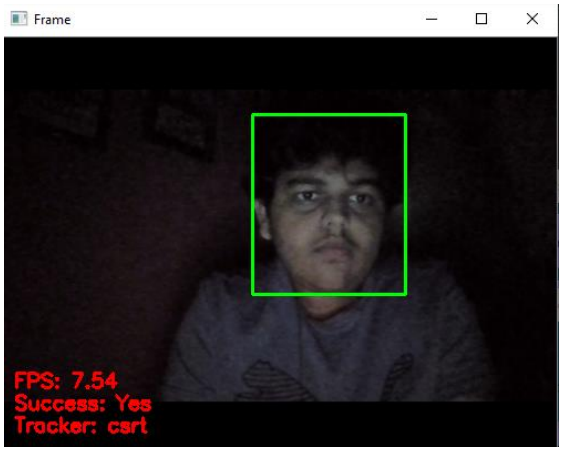


Fig 17: Tracking the region of interest using the inbuilt tracker in OpenCV.

The above images are glimpse of the coding in our project which heavily dependent on Python and its modules. The features of OpenCV amd how it can be used in real life or atleast be applied to is shown above.

## IV. Conclusion and Future Scope

Thus it is seen that Python is a very vast language with a lot of inbuilt modules and functions which can be incorporated into Computer Vision which can be used to perform tasks on both images and videos. This process is called as image processing. Using OpenCV images can both be processed and enhanced according to the user's requirement as well as it can be used for various other purposes as well contributing to different industries like security and automobiles.

Functions in OpenCV give us a brief glimpse of what a computer sees or observes as a result of the code written into it. With proper logic executed and the algorithm being error free Computer Vision can help computers perceive the real world in a virtual space.

The future scope of Computer Vision, image processing is very vast. Right from the implementation of Google lens to Self driving cars everything is possible due to Computer Vision. Information that is extracted from a processed image can be acted upon or even be used as additional information or as a dataset. The concept of a self-driving car that adjust speed limits or takes turns when needed to can be made possible with the implementation of Computer Vision Algorithms.

## References

[1] Automatic Detection and Recognize different shapes in an Image- Nidhal El Abbadi.
[2] Colour Thresholding Method for Image Segmentation of Natural Images- Nilima Kulkarni.
[3] An application of Face Recognition System using Image Processing and Neural Networks.
[4] Tom Mitchell – Neural net and Face images.
[5] Shape Detection by Adrian Rosenbrock.
[6] Improved Line Detection based-on Pixel Coordinates: A New Approach to Line Detection- El-Feghi.
[7] OpenCV Face Recognition by Adrian Rosebrock.
[8] Smoothing of images OpenCV.