OPEN CV PYTHON OBJECT DETECTION

[1]E Muralidhar Reddy, Assistant Professor , Mail ID:krishna81.reddy@gmail.com

[2]G Harika, Assistant Professor ,Mail ID:harikagora@gmail.com

[3]N Ch Ravi, Assistant Professor, Mail ID:ravi@saimail.com

[4]Dr.K.G.S.Venkatesan, Professor ,Mail ID:venkatesh.kgs@gmail.com

Department of CSE Engineering,

Pallavi Engineering College Hyderabad, Telangana 501505

ABSTRACT:

The concept of object detection in Python using OpenCV library and how you can utilize it to perform tasks like Facial detection. An object detection system finds objects of the real world present either in a digital image or a video, where the object can belong to any class of objects namely humans, cars, etc. In order to detect an object in an image or a video the system needs to have a few components in order to complete the task of detecting an object, they are a model database, a feature detector, a hypothesizer and a hypothesizer verifier. This paper presents a review of the various techniques that are used to detect an object, localize an object, categories an object, extract features, appearance information, and many more, in images and videos. An idea about the possible solution for the multi class object detection is also presented.

INTRODUCTION

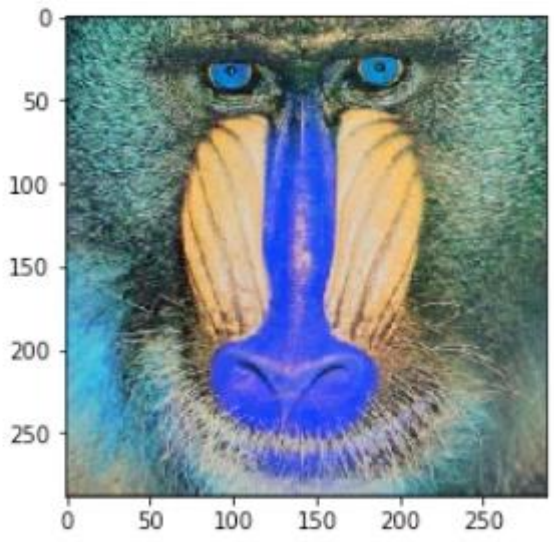
Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

1.1.1 Introduction to OpenCV-Python

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel’s Russian software Open CV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, Open CV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day. Currently Open CV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and Open CL are also under active development for high-speed GPU operations. Open CV-Python is the Python API of Open CV. It combines the best qualities of Open CV C++ API and Python language. Open CV-Python Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how Open CV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives MATLAB-style syntax. All the Open CV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with Open CV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So Open CV-Python is an appropriate tool for fast prototyping of computer vision problems.

Images and OpenCV

Before we jump into the process of face detection, let us learn some basics about working with Open CV. In this section we will perform simple operations on images using Open CV like opening images, drawing simple shapes on images and interacting with images through callbacks. This is necessary to create a foundation before we move towards the advanced stuff.



What we get as an output is a bit different concerning color. We expected a bright colored image but what we obtain is an image with some bluish tinge. That happens because OpenCV and matplotlib have different orders of primary colors. Whereas OpenCV reads images in the form of BGR, matplotlib, on the other hand, follows the order of RGB. Thus, when we read a file through OpenCV, we read it as if it contains channels in the order of blue, green and red. However, when we display the image using matplotlib, the red and blue channel gets swapped and hence the blue tinge. To avoid this issue, we will transform the channel to how matplotlib expects it to be using the function.

LITERATURE SURVEY

The research purpose of computer vision aims to simulate the manner of human eyes directly by using computer. Computer vision is such kind of research field which tries to percept and represent the 3D information for world objects. Its essence is to reconstruct the visual aspects of 3D object by analyzing the 2D information extracted accordingly [1]. Real life 3D objects are represented by 2D images. Object detection is the base for object tracking and object recognition, whose results directly affect the process and accuracy of object recognition. The common object detection method is: color-based approach, detecting objects based on their color values. The method is strong adaptability and robustness, however, the detection speed needs to be improved, because it requires test all possible windows by exhaustive search and has high computational complexity.

Object recognition has a wide domain of applications such as content-based image classification, video data mining, video surveillance and more. Object recognition accuracy has been a significant concern. Although deep learning had automated the feature extraction but hand crafted features continue to deliver consistent performance. This paper aims at efficient object recognition using hand crafted features based on Oriented Fast & Rotated BRIEF (Binary Robust Independent Elementary Features) and Scale Invariant Feature Transform features. Scale Invariant Feature Transform (SIFT) are particularly useful for analysis of images in light of different orientation and scale. Locality Preserving Projection (LPP) dimensionality reduction algorithm is explored to reduce the dimensions of obtained image feature vector. The execution of the proposed work is tested by using k-NN, decision tree and random forest classifiers. A dataset of 8000 samples of 100-class objects has been considered for experimental work. A precision rate of 69.8% and 76.9% has been achieved using ORB and SIFT feature descriptors, respectively. A combination of ORB and SIFT feature descriptors is also considered for experimental work. The integrated technique achieved an improved precision rate of 85.6% for the same.

In past few years, the detection of Objects in real time and Image processing has become an active area of research and several new approaches have been proposed. Several researchers have conducted many studies about Object detection1.

S.V. Viraktamath, Mukund Katti, Aditya Khatawkar & Pavan Kulkarni has conducted a study of openCV and also have published an IEEE paper for Face Detection and Tracking using OpenCV. Their work is related with converting web cam captured 2D Images and convert them into 3D Images related to human faces by constructing 3D Geometry data outputs [1]. 2. Ashish Pant, Arjun Arora, Sunnet Kumar and Prof. R.P. Arora form DIT Dehradun have researched about Image Processing and encrypting an Image in order to transfer safely over the networks. 3. They entitled their work as Sophisticated Image Encryption Using OpenCV [2]. 4. Kevinhughes, an elite individual in Opencv area has written a number of blogs containing projects tutorials in this area and steps for installing various softwares [3]. 5. Serge Belongie and Jitendra Malik, members of IEEE have done a wast study in the field of Shape Matching and Object Matching Based on their shapes, differentiating two object based on the difference in their shapes. 6. Orlando J. Tobias, and Rui Seara, Member, IEEE, have put their great efforts studding the ways and techniques for Image Segmentation and histogram Thresholding.

Description of Tools

In this section the tools and methodology to implement and evaluate face detection and tracking using Open CV are detailed.

Environment Configuration

Before getting started with the coding part, we have to configure some software to work properly with each other.

Required Soft wares:

1. OpenCV

2. Min GW

3. C make

4. IDE for C++

development Prescribed steps are to be followed for configuration process. Under windows 8.1 and Codeblocks 13.12, the steps that I followed are1. Downloading each software required. 2. Extracting OpenCV in “C:/” drive. 3. Adding Mingw to system path. 4. Installing Codeblocks. 5. Generating Binary makefiles using Cmake. 6. Add OpenCV to System paths. 7. Configuring Codeblocks with Mingw and OpenCV.

WORKING

OpenCV usually captures images in 8-bit, 3 channels, unsigned integer, BGR format. 1. Each image can be considered as combination of three matrices, each one corresponding to a color in RGB color scheme. 2. BGR – BLUE, GREEN, RED. 3. Each with integer value 0-255 (8 bit unsigned)

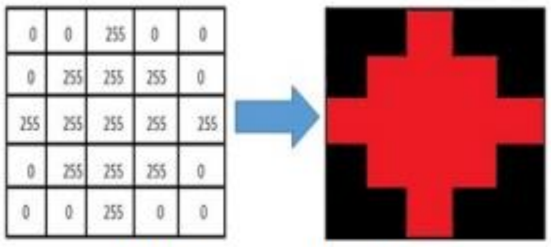


Fig 4: RED matrix example in RGB scheme.

Various Data types to store image in opencv

1. CV_8U (8 bit unsigned integer)

2. CV_16U (16 bit unsigned integer)

3. CV_16S (16 bit signed integer)

4. CV_32S (32 bit signed integer)

5. CV_32F (32 bit floating point number)

6. CV_64F (64 bit float floating point number)

BUT, HSV color space is the most suitable color space for color based image segmentation. So, in our project, we will convert the color space from BGR to HSV image. Hue values of basic colors under study1. Orange 00-22 2. Yellow 22-38 3. Green 38-75 4. Blue 75-130 5. Violet 130-160 6. Red 160-179 Saturation and Value depend on the lighting condition of the environment.

```
▪ import cv2
▪ import numpy as np
```

```
img = cv2.imread('chess.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
corners = cv2.goodFeaturesToTrack(gray, 100, 0.01, 15)

for corner in corners:
    x, y = corner[0]
    x = int(x)
    y = int(y)
    cv2.rectangle(img,(x-10,y-10),(x+10,y+10),(0,255,0), 2)

cv2.imshow("Corners Found", img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Results and Analysis

By choosing the range for Hue, Saturation and Value according to a particular color, we can easily detect any color object.

For example:

values used in our project are Hue- 170-179 (for Red color) Saturation- 150-255 Value- 60-255

With a RED object-

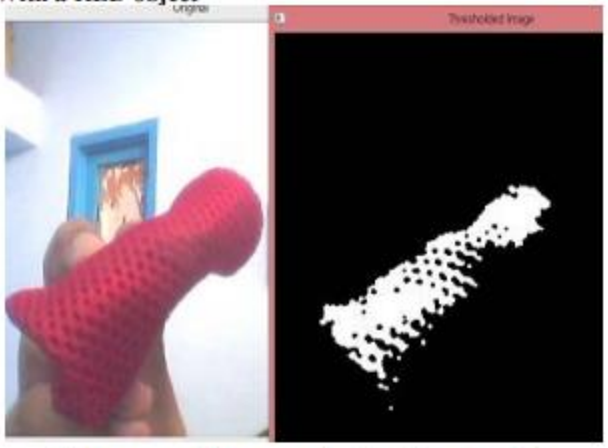


Fig 7: output of the program showing Object Detection.

Without a RED Object-

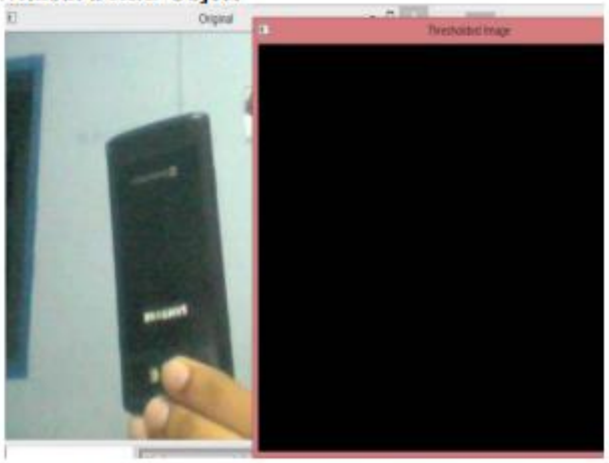
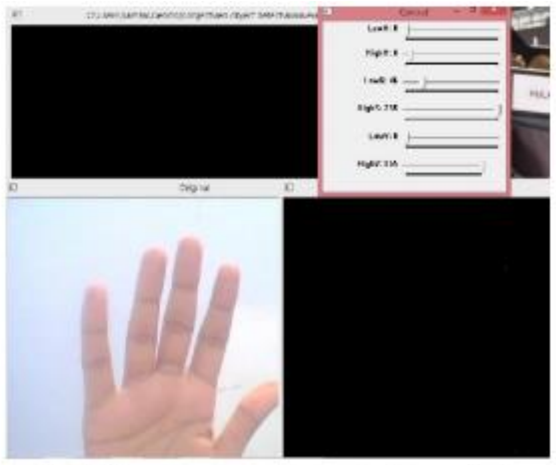


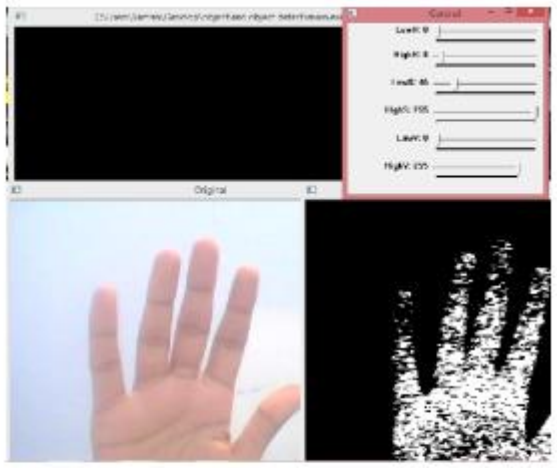
Fig 8: output of the program showing Object Detection.

So detection of any object by the computer web cam is done on the basis on the various color schemes values. [Figure 7] As the program was made just to detect the RED color object, it does so very well and give in output a thresholded image stream on the monitor of the computer. The program can’t detect any other color object, as shown in above [figure 8]. The main problem I was facing while working woth this project was as we developed program just for RED object detection, whenever we want to detect any object with different color, we have to make changes, every time in the source code. The best solution for this problem that I came up with is why not to make our program enough flexible that we can make changes according to the color of the object at runtime. So, I included the value bares for HUE, SATURATION, and value range on the top of the program at run time.



LowH: 0
HighH: 0
LowS: 46
HighS: 218
LowV: 0
HighV: 155

Fig 9: No output when we try to detect human hand



LowH: 0
HighH: 8
LowS: 46
HighS: 255
LowV: 0
HighV: 235

Fig 10: Detecting object with different colors with help of the value bars at runtime.

Future Work

Future work will include the Human Face Detection based on the generation of the 3D geometric data from the 2D image input. Face recognition may also be implemented and we can try to generate a security alert system that will give alerts when an intruder is going to trespass any unauthenticated area based on his/her face recognition. But all authenticated people face imprints data has to be provided to the system in advance. A Virtual mouse controller may also be implemented by time, controlling the movement of mouse pointer on the computer screen virtually just by waving our hands. Hand Gesture Recognition will also be an area open to work upon, making your computer system to work just by changing your hand positions and gestures.

CONCLUSION

Prototype system for color based object detection is successfully implemented and tested. The test results show that the detection method used in the paper can accurately detect and trace any object in real time. This paper shows the methods of Image processing and detecting an object in it based on its specific color, by using OpenCV real time implementation is possible. Thresholding of the generated image is necessary in order to segment the image pixels and let them free from each other. Future work includes so many possibilities related to the hand gestures recognition, virtual mouse controller, and also the face detection and recognition.

REFERENCES

1. Viraktamath SV, Mukund Katti, Aditya Khatawkar, Pavan Kulkarni, "Face Detection and Tracking using OpenCV," The SIJ Transaction on Computer Networks & Communication Engineering (CNCE), 2013, 1(3).

2. Pant A, Arora A, Kumar S, Arora RP. "Sophisticated Image Encryption Using OpenCV," International Journal of Advances Research in Computer Science and Software Engineering, 2012, 2(1).

3. Kevin Hughes – One more robot learn to see (http://kevinhughes.ca)

4. Belongie S, Malik J, Puzicha J. "Shape Matching and Object Recognition using shape contexts," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002; 24(4):509-522,

5. Tobias OJ, Seara R. "Image Segmentation by Histogram Thresholding Using Fuzzy Sets," IEEE Transactions on Image Processing, 2002; 11(12):1457-1465.

6. http://www.opencv.org