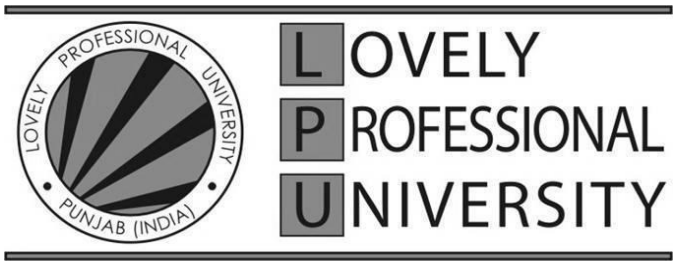


SIX WEEKS SUMMER TRAINING REPORT

on

"LPU - Object Oriented Programming using Python – Internship"

Submitted by :-
STUDENT NAME : Narendra Teja Thatavarthi
REGISTRATION NO : 11911625
PROGRAM NAME : B.TECH(CSE)

Under the Guidance of

E-BOX

School of Computer Science & Engineering Lovely Professional University, Phagwara

(15 June- 20 July, 2021)

DECLARATION

I hereby declare that I have completed my six weeks summer training at EBOX from 15 June 2021-20 July 2021 under the guidance of E-BOX. I have declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of B.TECH, Lovely Professional University, Phagwara.

DATE:-25 JULY-2021

Acknowledgement

A Project is a golden opportunity for learning and self development. I consider myself very lucky and honoured to have so many wonderful people lead me through in completion of summer training internship.

I wish to express my indebted gratitude I express my deepest thanks to E-BOX organisation for taking me as a part of this organisation.

I am very thankful to everyone who all supported me , for this project I have completed my internship training efficiently and, moreover , on time.

This resource pack would not have been possible without the support of many individuals inside and outside E-BOX.

I thanks to all the authors, editors, producers, publishers, educators and others who have graciously agreed to the use of, or access to, their materials. I hope they will be pleased with the interest and use of this resource pack that we anticipate from people all over the world.

CERTIFICATE

OF COMPLETION

This is to certify that

Thatavarthi Narendra Teja

has successfully completed the E-Box Online Certification Course on

"LPU - Object Oriented Programming using Python - Internship"

during the period Jun 2021 - Jul 2021.

Managing Director

Amphisoft

INTRODUCTION:

Object-oriented programming (OOP) is a method of structuring a program by bundling person related properties and behaviors into individual **objects**. In the training period I have learnt the basics of object-oriented programming in Python.

Conceptually, objects are like the components of a system. Think of a program as a factory assembly line of sorts. At each step of the assembly line a system component processes some material, ultimately transforming raw material into a finished product.

An object contains data, like the raw or preprocessed materials at each step on an assembly line, and behavior, like the action each assembly line component performs.

Object Oriented Programming is a way of computer programming using the idea of objects to represents data and methods. It is also, an approach used for creating neat and reusable code instead of a redundant one. the program is divided into self-contained objects or several miniprograms. Every Individual object represents a different part of the application having its own logic and data to communicate within themselves.

For instance, an object could represent with properties like a name, age, and address and behavior such as walking, talking, breathing, and running. Or it could represent an email with properties like a recipient list, subject, and body and behaviors like adding attachments and sending.

Put another way, object-oriented programming is an approach for modeling concrete, real-world things, like cars, as well as relations between things, like companies and employees, students and teachers, and so on. OOP models real-world entities as software objects that have some data associated with them and can perform certain functions.

Another common programming paradigm is **procedural programming**, which structures a program like a recipe in that it provides a set of steps, in the form of functions and code blocks, that flow sequentially in order to complete a task.

The key takeaway is that objects are at the center of object-oriented programming in Python, not only representing the data, as in procedural programming, but in the overall structure of the program as well**.**

TECHNOLOGY LEARNT:

It deals with declaring Python classes and objects which lays the foundation of OOPs concepts. The object-oriented programming python will walk you through declaring python classes instantiating objects from them along with the four methodologies of OOPs.

Object-oriented design centers on finding an appropriate set of classes and defining their contents and behavior. It involves determining the proper use set of classes and then filling in the details of their implementation. Object-oriented design is fundamentally a three-step process: identifying the classes, characterizing them, and then defining the associated actions
.
In this internship, I have discovered how to create modular, flexible and reusable software, by applying object oriented design principles and guidelines. By the completion of this internship, I have a solid knowledge of methods and techniques in Object Oriented Design and Programming.

TOPICS COVERED:

1.Basics of Python programming/Control structures

2. Strings / List and Dictionary

3.Function and Recursion

4.Modules and Packages / Date and Time

5.Classes and objects

6.Relationship with classes

7.Inheritance

8.Abstract classes & Interface

9.multi-Threading

10.Lamda function

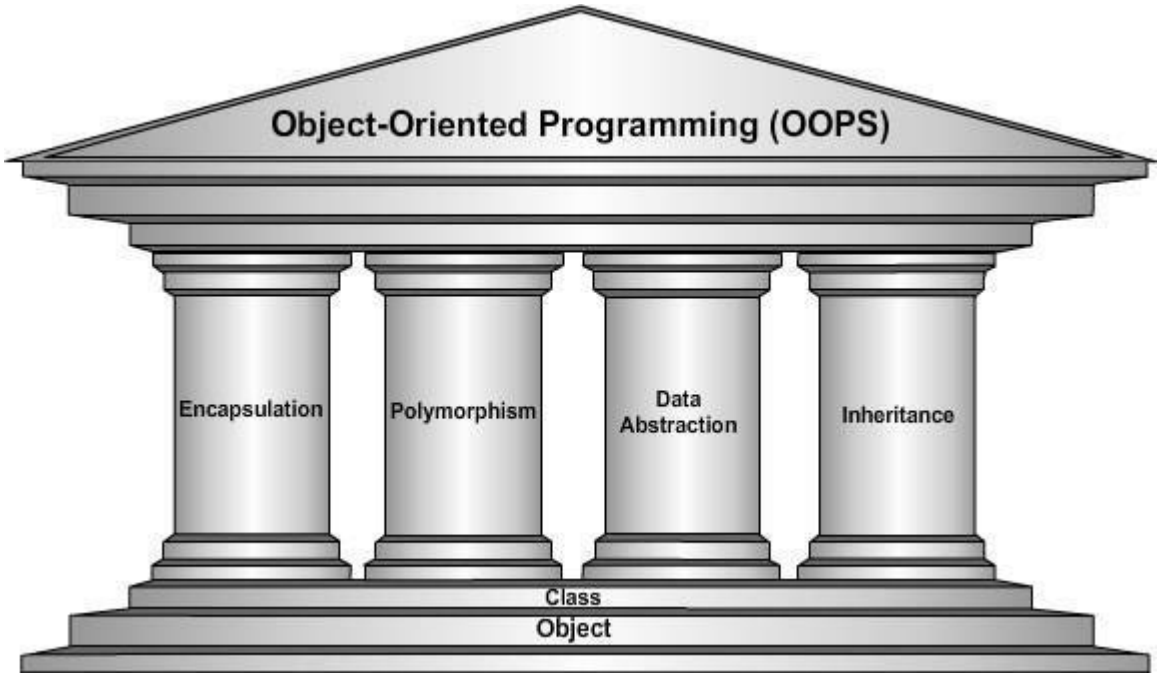
11.Streams

12.DB Connection

REASON FOR CHOOSING THIS TECHNOLOGY:

Object Oriented means directed towards objects. In precise words, it means functionally directed towards modeling objects which is one of the best techniques used for modelling complex systems.

Object-Oriented Python, the heart of Python programming is a way of programming that focuses on using objects and classes to design and build applications. Major pillars of Object-Oriented Programming (OOP) are Inheritance, Polymorphism, Data Abstraction, and Encapsulation.



Why Choose Object Oriented Programming in Python?
Python was designed with OOP approach, and it offers the following advantages:

1.Provides a clear program structure and a clean code

2.Facilitates easy maintenance and modification of existing code

3.Since the class is sharable**,** the code can be reused

4.Since the class is sharable, the code can be reused and many other such Advantages.

PROJECT

Project title**: COLOR DETECTION USING OPEN CV**

INTODUCTION

The 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 by analyzing the 2D information extracted accordingly. Real life 3D objects are represented by 2D images.

What is Computer Vision?

Computer Vision is among the hottest fields in any industry right now. It is thriving thanks to the rapid advances in technology and research. But it can be a daunting space for newcomers. There are some common challenges data scientists face when transitioning into computer vision, including:

How do we clean image datasets? Images come in different shapes and sizes.

The ever-present problem of acquiring data. Should we collect more images before building our computer vision model?

Is learning deep learning compulsory for building computer vision models? Can we not use machine learning techniques?

Can we build a computer vision model on our own machine? Not everyone has access to GPUs and TPUs!

Introduction to Open CV

Open CV is an open-source computer vision library. The library is written in C and C++and runs under Linux, Windows, and provides interfaces for Python, Ruby, Mat lab and other languages. Open CV library contains abundant advanced math functions, image processing functions, and computer vision functions that span many areas in vision.

Why use OpenCV for Computer Vision Tasks?

OpenCV, or Open-Source Computer Vision library, started out as a research project at Intel. It's currently the largest computer vision library in terms of the sheer number of functions it holds.
OpenCV contains implementations of more than 2500 algorithms! It is freely available for commercial as well as academic purposes. And the joy doesn't end there! The library has interfaces for multiple languages, including Python, Java, and C++. The first OpenCV version, 1.0, was released in 2006 and the OpenCV community has grown leaps and bounds since then.
Reading, Writing and Displaying Images

Machines see and process everything using numbers, including images and text. How do you convert images to numbers – I can hear you wondering. Two words –Pixel values:

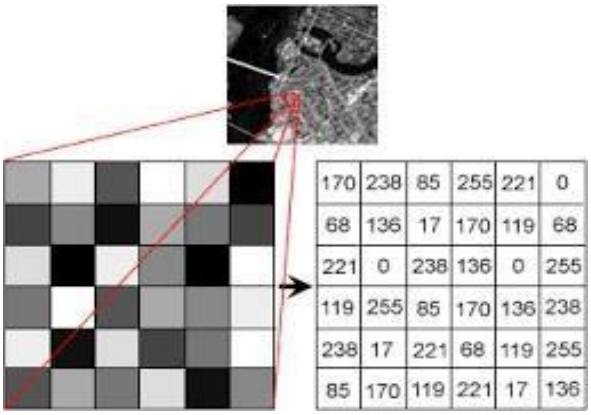


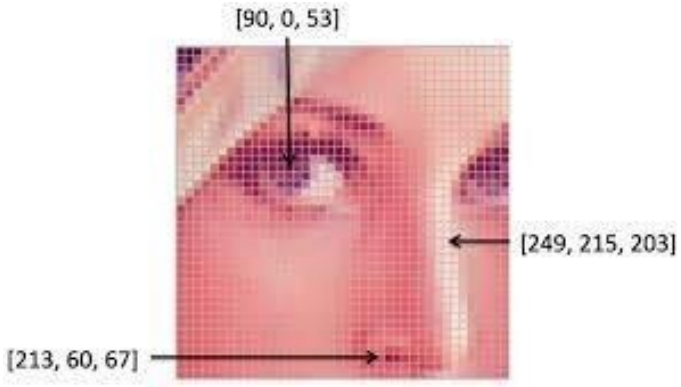
FIG:GREY SCALE IMAGE

COLOR DETECTION USING OPEN CV

Every number represents the pixel intensity at that location. In the above image, I have shown the pixel values for a gray scale image where every pixel contains only one value i.e., the intensity of the black color at that location

.

Note that color images will have multiple values for a single pixel. These values represent the intensity of respective channels – Red, Green and Blue channels for RGB images, for instance. Reading and writing images is essential to any computer vision project. And the OpenCV library makes this function a whole lot easier.



PIXEL VALUE OF IMAGE

System Analysis

Existing System:

Existing system is based on the color joint probability function. In this it will look for the centroid of the colors and color edge co-occurrence histogram. So that the accuracy of this system will less

.

Disadvantages:

1.The output will not be accurate.

2.The compiled time taken by existing system is more than proposed system.

Proposed System:

Here we are going to use the HSV for the color conversion, based on the key points matching the color object is identified.

Advantages:

1.Better accuracy in segmentation under various illuminations

.

2.Less time-consuming process

.

3.It is less sensitive to background noise.

Applications

1.People counting.

2.Vehicle detection.

3.Manufacturing industry applications

.

4.Tracking objects.

System Requirements

Hardware Requirements:

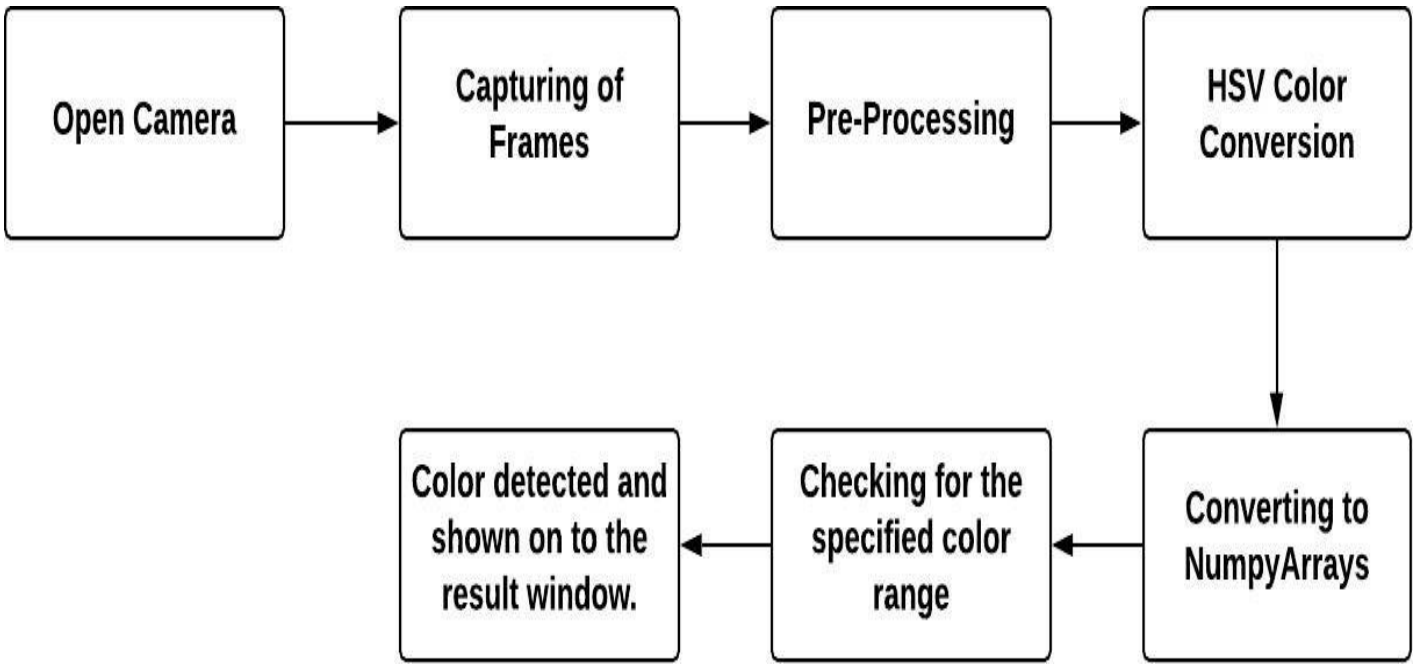
Processor	:	i3 processor
RAM	:	1 GB(min)
Hard Disk	:	50 GB

Software Requirements:

Operating system	:	Windows 7 or higher
Coding Language	:	Python
Front End	:	Python
Back End	:	MYSQL

SYSTEM ANALYSIS

Context diagram



Step 1: Capture video through webcam.

Step 2: Read the video stream in image frames

Step 3: Convert the imageFrame in BGR(RGB color space represented as three matrices of red, green and blue with integer values from 0 to 255) to HSV(hue-saturation-value) color space. Hue describes a color in terms of saturation, represents the amount of gray color in that color and value describes the brightness or intensity of the color. This can be represented as three matrices in the range of 0-179, 0-255 and 0-255 respectively.

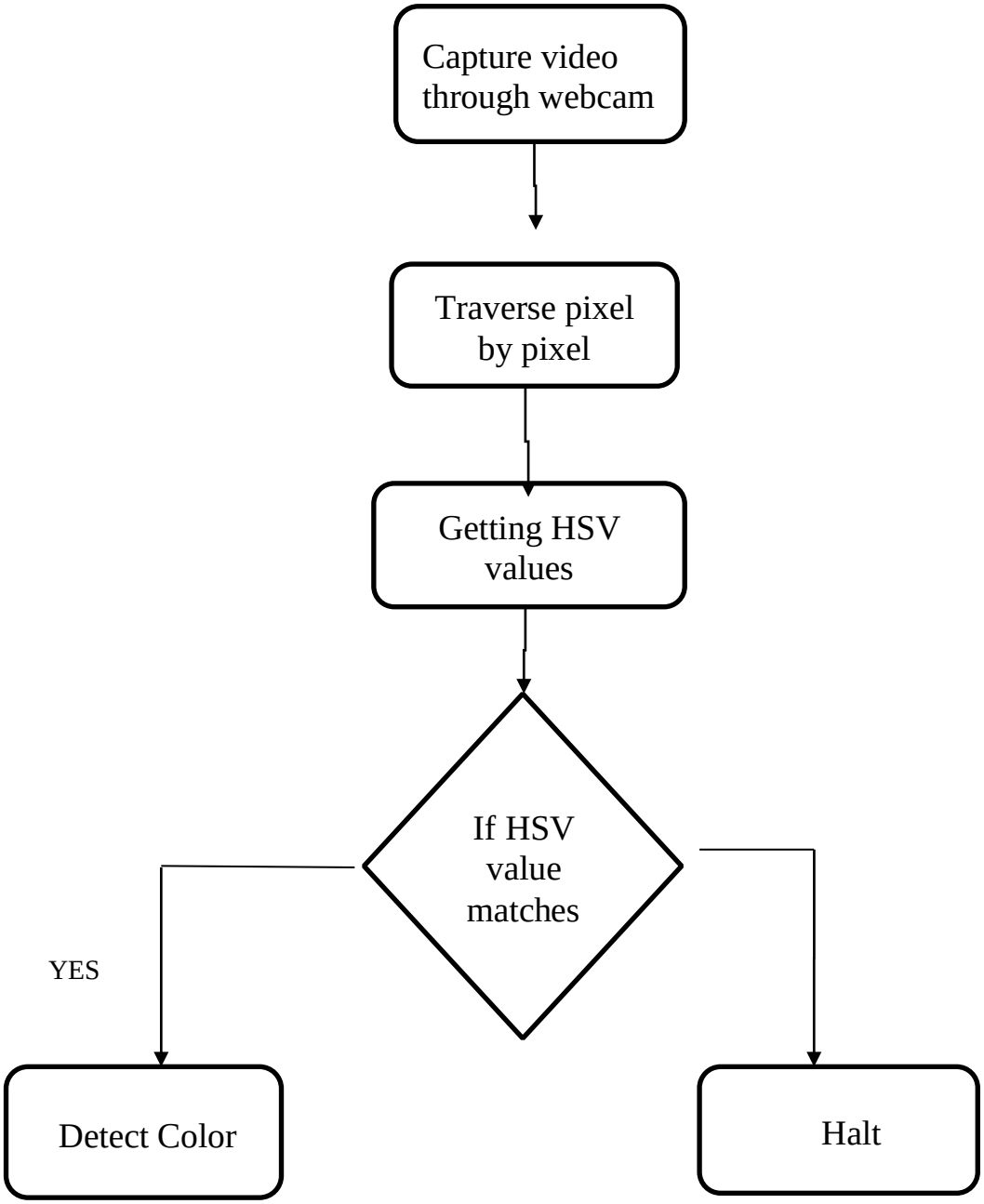
Step 4: Define the range of each color and create the corresponding mask

Step 5: Create contour for the individual colors to display the detected colored region distinguishly.

Step 6: Output Detection of the colors in real-time.

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



Modules

Module 1: Initialize

In the initialize module, we imported the **OpenCV and NumPy in** the first and second line. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. After that, you can capture video frame by frame.

Module 2: Detection

In the detection module above, for color conversion, we use the flag cv2.COLOR_BGR2HSV. Now we know how to convert BGR to HSV, we can use this to extract a colored object. In HSV, it is easier to represent a color than RGB color space. In specifying the range, we have specified the range of color. Whereas you can enter the range of any color to wish.

Module 3: Displaying Colour

In the above code, we are creating a mask that comprises the object in specified color. After that we have used a bitwise_ and on the input image and the threshold image so that only specified color objects are highlighted and stored in res. We then display the frame, res, and mask on separate windows using imshow function

Code and Implementation

Code

```
import cv2 import numpy
as np cap =
cv2.VideoCapture(0)
boundaries = [([45,100,50],[75,255,255]),([0, 70, 50], [10, 255, 255]), ([170, 70, 50],[180, 255,
255])] while(True):
  ret, frame = cap.read()
```

```
  hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
  lower_green = np.array(boundaries[0][0])
upper_green = np.array(boundaries[0][1])    lower_red
= np.array(boundaries[1][0])    upper_red =
np.array(boundaries[1][1])    lower_red2 =
np.array(boundaries[2][0])    upper_red2 =
np.array(boundaries[2][1])    mask1 = cv2.inRange(hsv,
lower_green, upper_green)    mask2 =
cv2.inRange(hsv, lower_red, upper_red)    mask3 =
cv2.inRange(hsv, lower_red2, upper_red2)    mask =
mask1 | mask2 | mask3
  res = cv2.bitwise_and(frame, frame, mask= mask)
cv2.imshow("frame", frame)
#cv2.imshow("mask", mask)    cv2.imshow("res",
res)    k = cv2.waitKey(60) &0xFF    if k == 27:
cap.release()    cv2.destroyAllWindows()
   break
```

Explanation of Code:

Object detection and segmentation is the most important and challenging fundamental task of computer vision. It is a critical part in many applications such as image search, scene understanding, etc. However, it is still an open problem due to the variety and complexity of object classes and backgrounds.

The easiest way to detect and segment an object from an image is the color-based methods. The object and the background should have a significant color difference in order to successfully segment objects using color-based methods.

- **Camera Settings:** To perform runtime operations, the device's web camera is used. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected, so we simply pass 0. You can select the second camera by passing 1 and

so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture. Moreover, if anyone wants to apply this color detection technique on any image it can be done with little modifications in the code which I'll discuss later.

- **Capturing frames:** The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program. After capturing the live stream frame by frame we are converting each frame in BGR color space (the default one) to HSV color space. There are more than 150 color-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV. For color conversion, we use the function cv2.cvtColor(input_image, flag) where flag determines the type of conversion. For BGR to HSV, we use the flag cv2.COLOR_BGR2HSV. Now we know how to convert BGR image to HSV, we can use this to extract a colored object. In HSV, it is more easier to represent a color than RGB colorspace.In specifying the range , we have specified the range of blue color. Whereas you can enter the range of any colour you wish.

- **RGB to HSV Conversion:**OpenCV usually captures images and videos in 8- bit, unsigned integer, BGR format. In other words, captured images can be considered as 3 matrices; BLUE, GREEN and RED (hence the name BGR) with integer values ranges from 0 to 255.

  The following image shows how a color image is represented using 3 matrices. In the above image, each small box represents a pixel of the image. In real images, these pixels are so small that human eye cannot differentiate.

Image representation using three matrices:

Blue matrix

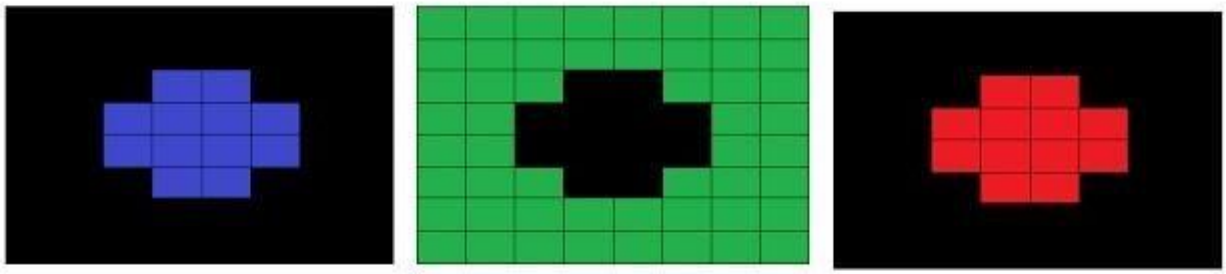
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Green matrix

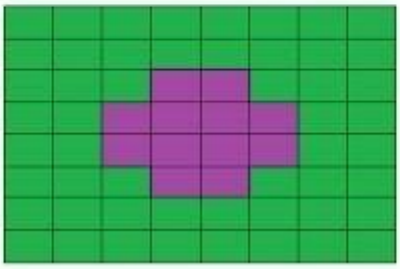
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	0	0	255	255	255
255	255	0	0	0	0	255	255
255	255	0	0	0	0	255	255
255	255	255	0	0	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

Red matrix

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Resultant Image



Usually, one can think that BGR color space is more suitable for color based segmentation. But HSV color space is the most suitable color space for color based image segmentation. So, in the above application, I have converted the color space of original image of the video from BGR to HSV image.

HSV color space is also consists of 3 matrices, HUE, SATURATION and VALUE. In OpenCV, value range for HUE, SATURATION and VALUE are respectively 0179, 0-255 and 0-255. HUE represents the color, SATURATION represents the amount to which that respective color is mixed with white and VALUE represents the amount to which that respective color is mixed with black.

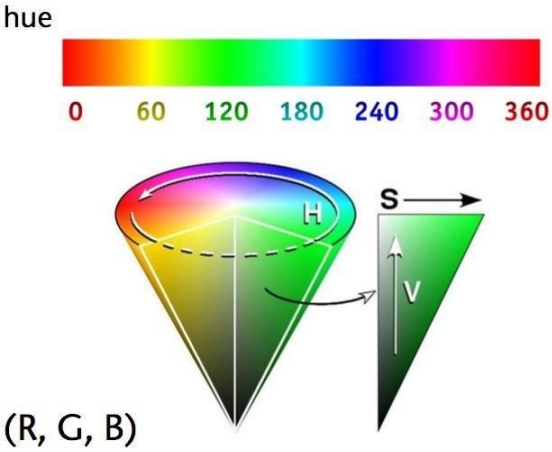
In the above application, I have considered that the red object has HUE, SATURATION and VALUE in between 170-180, 160-255, 60-255

respectively. Here the HUE is unique for that specific color distribution of that object. But SATURATION and VALUE may be vary according to the lighting condition of that environment.

Hue values of basic colors

- Orange 0-22
- Yellow 22- 38
- Green 38-75
- Blue 75-130
- Violet 130-160
- Red 160-179

These are approximate values. You have to find the exact range of HUE values according to the color of the object. I found that the range of 170-179 is perfect for the range of hue values of my object. The SATURATION and VALUE is depend on the lighting condition of the environment as well as the surface of the object.



- colour cone
  - H = hue / colour in degrees $\in$ [0,360]
  - S = saturation $\in$ [0,1]
  - V = value $\in$ [0,1]
- conversion RGB $\rightarrow$ HSV
  - V = max = max (R, G, B), min = min (R, G, B)
  - S = (max – min) / max (or S = 0, if V = 0)
  - $$H = 60 \times \begin{cases} 0 + (G - B) / (\max - \min), & \text{if } \max = R \\ 2 + (B - R) / (\max - \min), & \text{if } \max = G \\ 4 + (R - G) / (\max - \min), & \text{if } \max = B \end{cases}$$

    $H = H + 360$, if $H < 0$

Masking technique: The mask is basically creating some specific region of the image following certain rules. Here we are creating a mask that comprises of an object in blue color. After that I have used a bitwise_and on the input image and the threshold image so that only the blue coloured objects are highlighted and stored in res.We then display the frame, res and mask on 3 separate windows using imshow function.

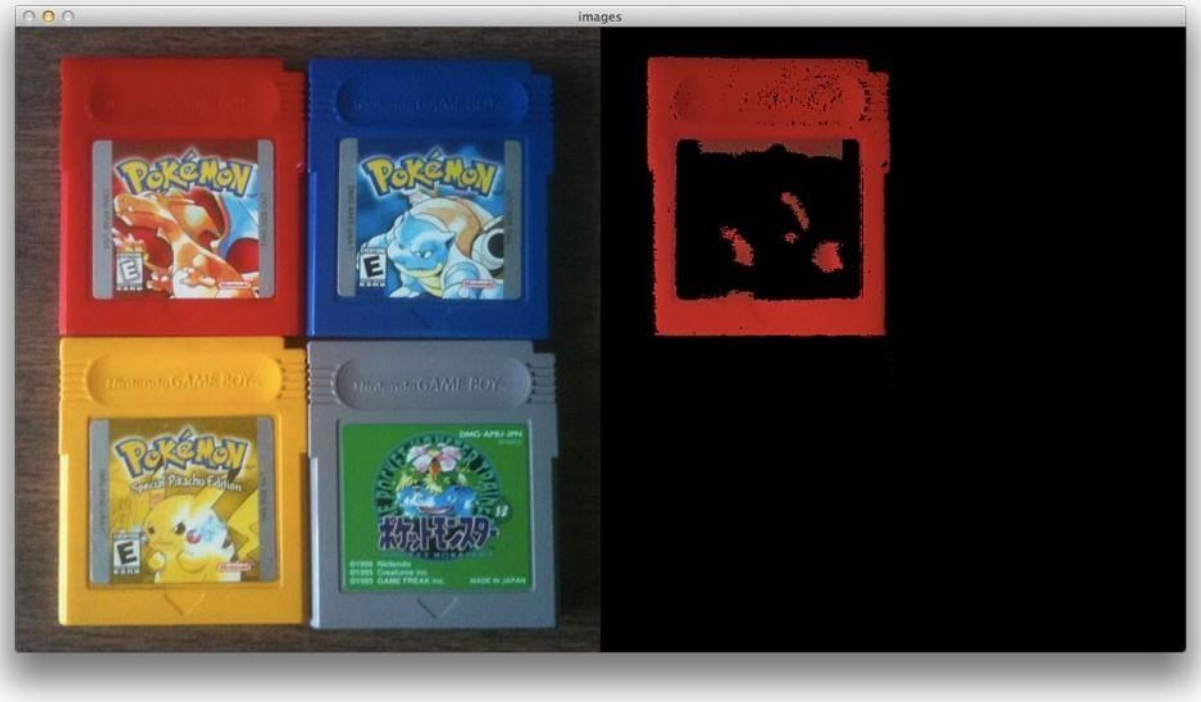
Display the frame: As imshow() is a function of HighGui it is required to call waitKey regularly, in order to process its event loop.The function waitKey() waits for key event for a "delay" (here, 5 milliseconds). If you don't call waitKey, HighGui cannot process windows events like redraw, resizing, input event etc. So just call it, even with a 1ms delay .

Summarizing the process:

1. Take each frame of the video.
2. Convert each frame from BGR to HSV color-space.
3. Threshold the HSV image for a range of blue color **Result**

Output:

We have specified the ranges of RED color, so in the below image it has only extracted the specified color into the resultant window by ignoring remaining colors.



FUTURE ENHANCEMENT

- The technology is implemented to separate objects and objects can be tracked based on its color.
- Color Detection is the basic and important step for proceeding in computer vision.
- The spy robots are made to identify objects in the place where they are launched. Object's color, shape, size is important to robot.

Conclusion

Computer Vision can be used to solve the most intriguing problems with utmost sophistication. All the basics regarding the detection technique along with different ways to achieve it . During the course of programming, we can use either python or MATLAB for Computer Vision, but we prefer python because it takes less stimulation time than MATLAB. Colors, shapes, contours can be easily detected by using this module.

References

- https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/

- https://www.geeksforgeeks.org/detection-specific-colorblue-using opencv- python/

- https://www.analyticsvidhya.com/blog/2019/03/opencv-functionshttps://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/computer-vision-python/

- https://www.opencv-srf.com/2010/09/object-detection-usinghttps://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.htmlcolor- seperation.html