

PROFESSIONAL TRAINING REPORT
at
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be university)

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

by

Solapuri Man Mohan
[Reg. No.3812080]



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119, TAMILNADU.

AUGUST 2020



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be university)
Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **SOLAPURI MAN MOHAN [Reg.No:38120080]** who carried out the training entitled **"COLOR DETECTION USING OPENCV"** under our supervision from April 2020 to August 2020.

Internal Guide

Dr. P.AJITHA, M.E., Ph.D.,

Head of the Department

Dr. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **SOLAPURI MAN MOHAN [Reg.No:38120080]** hereby declare that the Professional Training Report entitled “**COLOR DETECTION USING OPENCV**” done by me under the guidance of **Dr. P.AJITHA , M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in INFORMATION TECHNOLOGY.

DATE:

PLACE: CHENNAI
SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this professional training and for completing it successfully.

I convey my thanks to **Dr. T .SASIKALA, M.E., Ph.D.,**Dean,School of Computing and **Dr.SUBHASHINI, M.E., Ph.D.,AND Dr. L. LAKSHMANAN, M.E., Ph.D.,** Heads of the Department of INFORMATION TECHNOLOGY for providing me necessary support and details at the time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Guide **Dr. P.AJITHA, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my Professional Training work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of my Professional Training.

TRAINING CERTIFICATE



ABSTRACT

The data used in this work are collected in an indoor environment using a laptop webcam under different illumination conditions. Segmentation refers to a technique of picking out a particular part of any object. If we talk about image segmentation it is basically a method of picking out a particular area of that image. As we know a video is nothing but a sequence of frames in an orderly manner, hence video segmentation could also be achieved in this manner. Video segmentation is in great demand now a day with the advancement of Artificial Intelligence. Few important applications of video segmentation could be a face, color detection system, an object tracking system etc. Detection and tracking of any moving object is a very important part of video surveillance and hence a lot of emphasis is done on it. Lots of researches had been taking place in the domain of object detection and tracking from live stream and various algorithms were suggested to obtain the most effective results. The survey conducted would cover all important researches and works done on object detection and tracking especially the color as an object part and will also discuss the benefits of doing this. In short we will make out how video segmentation is going to open the gate for a great and smart future ahead.

TABLE OF CONTENTS

CHAPTER NO

TITLE

Page

N
o
.

	ABSTRACT	vi
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1.	INTRODUCTION	xi
	1.1 THE BGR COLOR MODEL	xi
	1.2 THE HSV COLOR MODEL	xii
	1.3 THE HLS COLOR MODEL	xiii
2.	LITERATURE SURVEY	xvi
3.	PROPOSED ALGORITHM	xxii
4.	SYSTEM IMPLEMENTATION	xxiv
	4.1 GENERAL	
	4.1.1 PYTHON INSTALLATION	xxiv
	4.1.2 OPEN CV INSTALATION	xxvi
	4.2 SUMMARY	xxvi
	RESULTS AND DISCUSSIONS	xxx
5.	CONCLUSION AND FUTURE WORK	xxxiii
	5.1 CONCLUSION	xxxiii
	5.2 FUTURE WORK	xxxiii
	REFERENCES	xxxiv
	APPENDIX	xxxvi
	A. SOURCE CODE	xxxvi
	B. SCREEN SHOTS	xli

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
1.1	RGB color model	xii
1.2	HSV color model	xiii
1.3	HLS color model	xiv
4.1	Python Installation	xxiv
4.2	Python Downloading	xxv
4.3	Opencv Downloading	xxvi
4.4	Opencv Installation	xxvi
4.5	Package Installation	xxvii
4.6	Opencv Shell	xxvii

LIST OF TABLES

TABLE No.	TITLE	PAGE No.
2.1	Various Tracking Methods	xxvii

LIST OF ABBREVIATIONS

BGR	-	Blue Green Red
HSV	-	Hue Saturation Value
HLS	-	Hue lightness Saturation

CHAPTER 1

INTRODUCTION

Image is an image processing library that implements algorithms and utilities for use in research, education and industry applications. Released under the liberal Modified BSD open source license, provides a well-documented API in the Python programming language, and is developed by an active, international team of collaborators. In the paper we highlight the advantages of open source to achieve the goals of the image library, and we showcase several real-world image processing applications that use image.

With the advancements of artificial intelligence, there has been a lot of focus on video segmentation and tracking as it opens the door for future. A lot of research has already been done to accomplish segmentation from any live stream and also many are still in progress. Color segmentation is another important part of object segmentation process. Color segmentation is a process through which colour could be detected from a live stream. The scope of detecting and tracking object especially moving object is quite large in the field of computer vision. Various purposes like video surveillance, live conferencing etc have a lot to do with object detection and tracking. High resolution frames at high frame rates are the major challenges faced in order to track object. There are a number of color models which help us in understanding the object in a better way, some of them are explained below:

1.1 The BGR Color Model

The RGB (Red, Green, and Blue) color model uses a Cartesian coordinate.

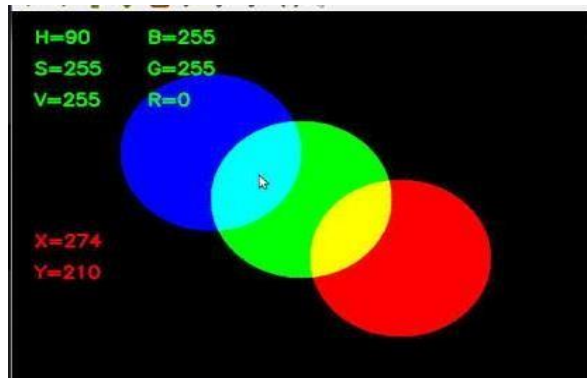


Fig. 1.1: The BGR Color Model

The RGB color model consists of the three primary colors – red, green and blue. The main idea behind this model is that combining these three primary colors in various proportions, various other colors could be produced. As the RGB color

- $r = R/R+G+B$
- $g = G/R+G+B$
- $b = B/R+G+B$

model is hardware oriented which is used in many image capturing, processing and rendering devices.

1.2 The HSV Color Model

The HSV in the HSV color model stands for Hue, Saturation, and Value. The model was introduced by A. Smith and is basically a user governed color model. The model is based on the spontaneous request of the user's tint, shade, and tone. The region within which the HSV color model is defined is a hex cone. Figure 2 shows how a hex cone looks like:

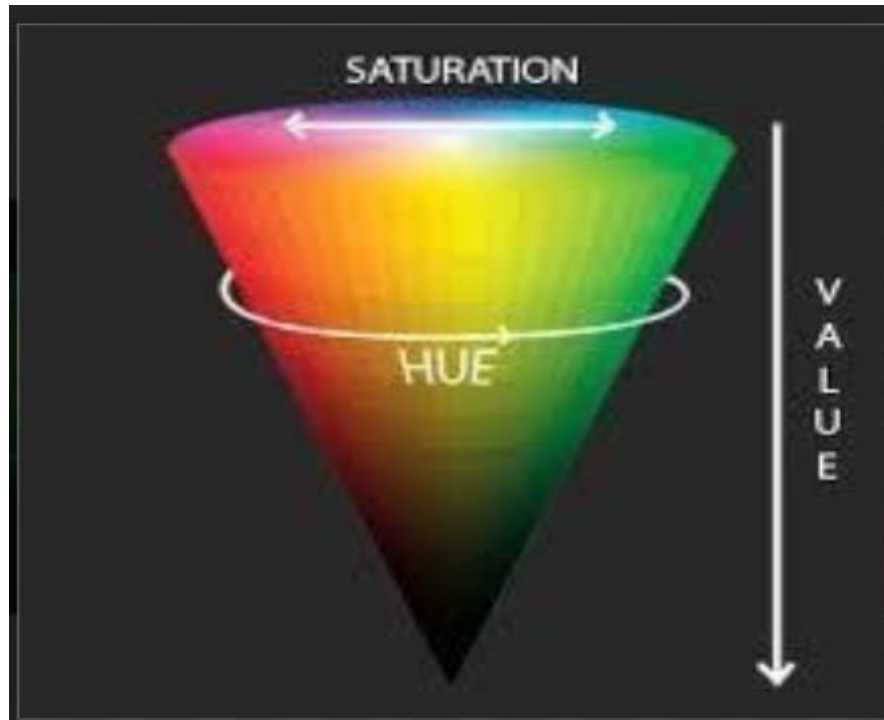


Fig. 1.2 HSV Color Model

1.3 The HLS Color Model

The HLS in the HLS color model stands for Hue, Lightness, and Saturation. This color model was introduced by W. Ostwald. The definition of Hue is the same as in the HSV model. In the HSL color model black occupies the bottom tip of hex cone while white occupies the top tip of hex cone as shown in the figure is actually the measure of lightness. The value of lightness is 0 in case of black and 1 in case of white. In fact, HLS is very much similar to HSV with some deformities, where white is placed on top of the top cone. Saturation is radical measurement from the vertical gray axis, ranging from 0 on the vertical gray axis to 1 on the surface of the hex cone. one can find the maximum saturated hues at $S = 1$ and $L = 0.5$. Hence with the help of various color models one can easily get details of any image like the colors used, hue level, saturation level etc. just like the images the details from a live stream video can also be detected as videos are nothing but sequences of images known as frames. Once the details of an image are obtained the segmentation could be achieved and we can detect and track color from a given live stream which appears like a double hex cone region as if the HSV model with its own image above as shown in Fig. 3.

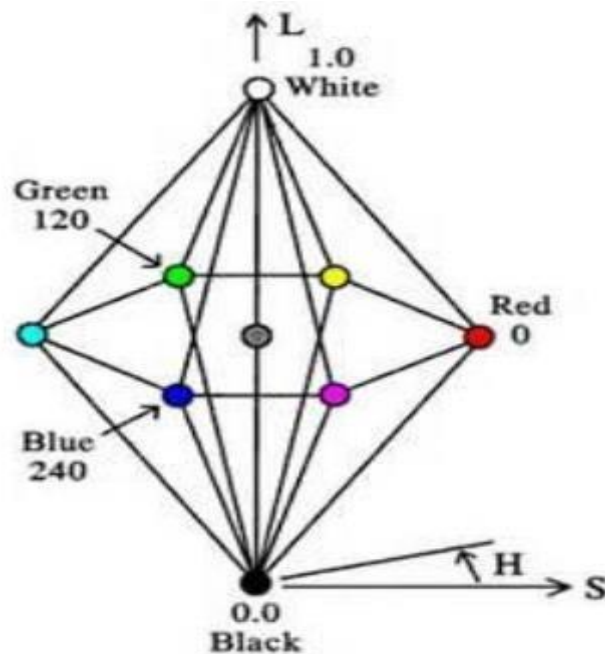


Fig. 1.3 HLS Color Model

HLS color model defines colors by three parameters hue(H), lightness(L), saturation(S). which was introduced by tektronix inc. hue lies on a circle, saturation increases from centre to edge of the circle, lightness goes from black to white. The model uses the same hue plane as the HSV model, but it replaces value V by an extended lightness access so that the maximum color gamut is at $L=0.5$ and decreases in each direction towards white ($L=1$) and black ($L=0$). the HLS color model is represented by a double hexagonal cone, with white at the top apex and black at the bottom.

Hue-saturation based color spaces were introduced when there was a need for the user to specify color properties numerically. They describe color with intuitive values, based on the artist's idea of tint, saturation and tone. Hue defines the dominant color (such as red green, purple and yellow) of an area, saturation measures the colorfulness of an area in proportion to its brightness The intensity, lightness or value is related to the color luminance.

The intuitiveness of the color space components and explicit discrimination between luminance and chrominance properties made these color spaces popular in the works on skin color segmentation.

$$H = \arccos[(1/2)((R-G) + (R-B))/\sqrt{(R-G)^2 + (R-B)(G-B)}]$$

$$S = 1-3[(\min(R,G,B))/R+G+B]$$

$$V = 1/3(R+G+B)$$

CHAPTER 2

LITERATURE SURVEY

Kalisa Wilson et .al in her paper on “Real- Time Tracking for Multiple Objects Based on Implementation of RGB Color Space in Video” proposed an algorithm to track multiple color objects in real time. The framework suggested by her was basically based on using the steps of image processing like the thresholding. Using her algorithm in the real-time video first pre-processing was done and then the color discrimination part was achieved and finally segmentation and tracking part. Ssu-Wei Chen et .al in his paper on “Moving Object tracking Based on Background Subtraction Combined Temporal Difference” proposed a new tracking method that uses Three Temporal Difference(TTD) and the Gaussian Mixture Model(GMM) approach for object tracking. The three temporal differences i.e. TTD method suggested by them make use of continuous image subtraction

Vandana S. Bhatai et .al in his paper on “Face detection system using HSV color model and morphing operations” proposed an algorithm which uses the fact that the human faces have unique color features. Using skin color to detect face regions is a much faster approach. First the image was read in the RGB format and then converted to the HSV model. Then the image gets converted to the binary format and finally segmentation was done and face gets detected.

Ravikant Gupta et. al in his paper on” Human Face Detection By YCbCrHs Technique” proposed a feature based algorithm to detect faces. The suggested algorithm used spatial filters to detect feature points from the image and then does the grouping part using various constraints at geometric and gray level. A probabilistic framework was then used to identify if the features extracted matches to a particular face or not.

Prasad Kalane et. al in his paper on “Target Tracking Using Kalman Filter” used kalman filter technique to detect any target. A target can be

anything, maybe a human or an umbrella or anything. He added that there are two key steps in target tracking:

1. Detecting the moving objects on which we are interested
2. Keeping the track of the interested objects from frame to frame.

Isaac Cohen et. al in his paper on " Detecting and Tracking Moving Objects for Video Surveillance" proposed to choose an affine model which approximates well the general perspective projection, while having a low numerical complexity.

R E Kalman et. al states that objects can be represented in three ways: Point tracking, Kernel tracking and Silhouette Tracking. Point tracking can be achieved using various Deterministic and statistical methods available. Kernel tracking makes the use of various Templates, multi-view appearance models and density based appearance models. Silhouette Tracking uses methods like matching the shapes method and contour evolution methods.

Table 2.1 Various Tracking Methods

Type of tracking	Methodology	Advantages	Limitations
Point tracking	MGE tracker	Can handle entries	Costly
Point tracking	GOA tracker	Handles misdetection	Assume no object entries and exists
Kernel Tracking	Mean shift	1. Mean shift is an application independent tool suitable for real data analysis. 2. Does not assume any predefined	1. Non trivial selection of window size. 2. "Shallow" modes can arise due to inappropriate window

		shape on data clusters.	sizes.
Kernel method	Layering method	Suited for multiple object detection	High cost
Kernel method	KLT Method	It is faster than traditional techniques for examining far fewer potential matches between the images	Efficient but don't satisfy today AI applications
Silhouette tracking	State space models	Can handle complex models for rigid and non-rigid objects	Entry and exit of objects are difficult to handle, gestures recognition is not so accurate

Andres Alarcon Ramirez et .al in his paper on “A New Algorithm for Tracking Objects in Videos of Cluttered Scenes” proposed an algorithm that constituted of two steps: the first step is estimating the direction of the object. Once the direction of the interested object is estimated, we can know where the object is going to lie in the next frame and the process continues. Hence with this approach, we can detect the frames and thereby the motion and changing directions of the interested object and hence our interested object could be detected and tracked.

Chris Harris et. al in his paper on “A COMBINED CORNER AND EDGE DETECTOR” proposed a technique to detect edges and corners in an image, junction would then be considered as points where various edges meet. To pursue the above approach he started from the Moravec's corner detector Moravec's corner detector considers a local window in the image, and then determined the average changes in the intensity of the image that resulted

due to shifting of the window by a little amount in various directions. Three cases needed to be considered:

1. If the image that was windowed has almost constant intensity, then all shifts will result in only a small change;
2. If the window stands or sits on an edge, then any shift along the edge will result in a small change, whereas any shift perpendicularly to the edge will result in a large change;
3. If the windowed patch is either a corner or an isolated point, then all shifts will result in a large change. A corner can thus be detected easily by finding when the shift is large even for any minimum change.

Katja Nummiaro et. al proposed a color-based object tracking in multi-camera environments. In this the object was placed in an environment having multiple cameras. To support multiple camera they used more than one histogram for a target. The cameras were placed at different angles in order to extract the features of the object from various angles. Since all the cameras keep a track of the object independently there must be a correspondence between the histograms

Marcus Thaler et. al proposed a real time person detection and tracking in panoramic video method. This is a useful technique for detecting any player in particular sports. As we know in any live event of a sport there are a number of cameras trying to take the best scene of the play, this technique can be used to detect any player.

Wen-cheng Wang et. al proposed a face detection method to be used in color images. He analysed four kinds of color space: RGB, HIS, YUV, and YCbCr. The main aim was to study the clustering features of the skin color in different color spaces and establish a practical and reliable color mixing model, and finally, use YCbCr color space for skin color detection, screen the candidate regions by integrating the geometric characteristics of the face, and then remove the similar skin region and other regions of body exposed, and continue the process to narrow search scope.

Hani k. Mohair et .al proposed a method to detect human skin color. In this method first the original image was transferred to color space and then skin detection was done. Once the skin areas were detected, after dilation and erosion, face was detected.

Monika Deswal et. al in her paper on" A Fast HSV Image Color and Texture Detection and Image Conversion Algorithm" proposed an algorithm to detect the texture of an image. First the RGB image was converted to the HSV image by removing noise and a corresponding database of HSV values was prepared. Then the level of H, S, and V were found and was normalized and finally the texture was calculated.

Douglas Chai et.al in his paper on" Face Segmentation Using Skin-Color Map in Videophone Applications" proposed a method of face segmentation using the skin-color map. The approach of his algorithm is as follow: first the head and shoulder image was taken as input and then color segmentation followed by density regularization, luminaries' regularization, geometric correction, contour extraction was done and finally segmented facial region was obtained.

Umapathy Eaganathan et.al explained about different color models such as RGB, YCbCr, HSV, HSI, CIELab in medical image processing and it additionally explored how these color models can be derived from the basic color model RGB. He did an analysis of various color models in medical image processing.

S. Chitra et.al did a comparative study for two color spaces HSCBCr and YCbCr in skin color detection. The paper concentrated on how the input images get converted to the HSV and YCbCr model and how the values of H, S, Cb, and Cr are collected and then to verify if these values satisfies the threshold values. If the pixels lie in the threshold range then that pixels will be considered as skin region otherwise it is a nonskin region. Thus using this skin and nonskin area could be detected easily. The algorithm concluded that HSCbCr gave better results than YCbCr model.

Intaek Kim et.al proposed a multi-target tracking using color information proposed a method that tracks multiple objects in a video applying extended Kalman filter and color information.

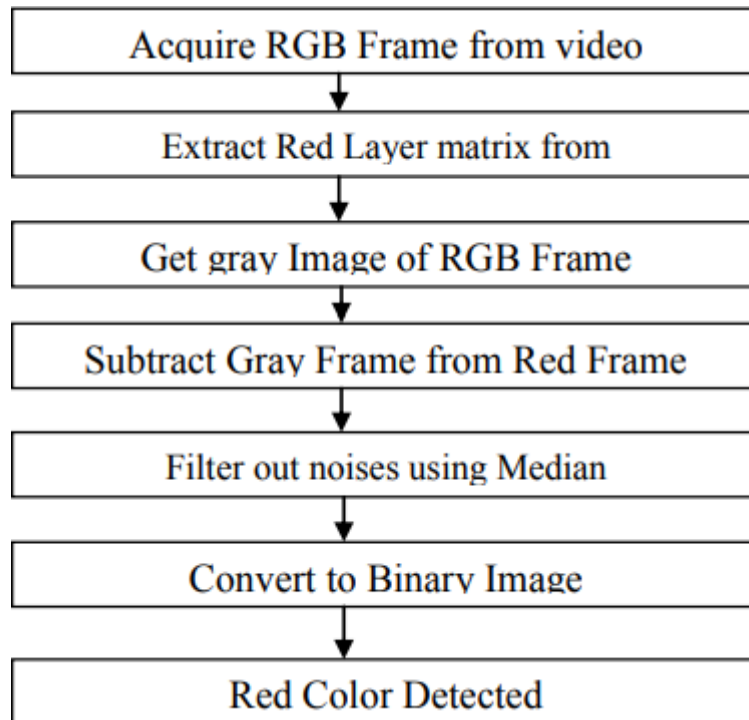
Akriti Kaushik et.al proposed a RGB color sensing technique. They used sensor to detect RGB color .color sensing is of great use in the field of medical science. The main idea behind this approach was that when the color object is placed before the sensor the color gets detected and the LED of the same color get on; it uses 8 colour LEDs that include three primary colors- red, green, blue and also include magenta, yellow and cyan as well as black and white.

Hasan Fleyeh et.al proposed a road and traffic detection sign color detection and segmentation technique. The color segmentation algorithm is carried out by placing a RGB camera on the moving car. The camera captures images from the road side and then converts it to the HSV color model as the HSV color model deals with the hue condition and hue is invariant under any type of light condition be it dark or bright.

CHAPTER 3

PROPOSED ALGORITHM

The following Flow chart shows how the Color detection and tracking algorithm is processed:



Algorithmic Steps:

Step 1: To acquire RGB framework from a live stream, once done the red color matrix gets extracted from the RGB framework.

Step 2: After this the gray image of RGB framework is obtained and the gray Frame is subtracted from the Red Frame.

Step 3: All the noises are filtered out using the median filter.

Step 4: The diff Frame is then converted to corresponding binary image with proper threshold value.

Step 5: Finally the red color is detected and outline is made in order to identify it from the live stream.

To detect colors in images, the first thing you need to do is define the upper and lower limits for your pixel values. Once you have defined your upper and lower limits, you then make a call to the `cv2.inRange` method which returns a mask, specifying which pixels fall into your specified upper and lower range. Finally, now that you have the mask, you can apply it to your image using the `cv2.bitwise_and` function.

How exactly do we perceive color? A number of theories have emerged to explain this phenomenon, and one of the earliest and best-known was the trichromatic theory. Two renowned researchers, Thomas Young and Hermann von Helmholtz, contributed to the trichromatic theory of color vision. The theory began when Thomas Young proposed that color vision results from the actions of three different receptors. As early as 1802, Young suggested that the eye contained different photoreceptor cells that were sensitive to different wavelengths of light in the visible spectrum.

The identification of the three receptors responsible for color vision did not occur until more than 70 years after the proposal of the theory of trichromatic vision. Researchers discovered that cone pigments have different levels of absorption. Cones are receptors located in the retina that are responsible for the vision of both color and detail. The cone receptors differ in absorption amounts due to the amount of opsin amino acids in the receptor. The three different cone receptors are:

- Short-wavelength cone receptors.
- Middle-wavelength cone receptors.
- Long-wavelength cone receptors.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 GENERAL

4.1.1 Python Installation

We used python to implement opencv. it is very easy to code and installation steps are quite common to all and can be installed easily with our own knowledge. the installation steps for python goes as follows:

open the chrome->select the version->click python download-> the file name python version (as selected) should start download in to your standard download folder. might take a while to download fully if you are on a slow internet connection. move this file to a more permanent location, so that you can install python.

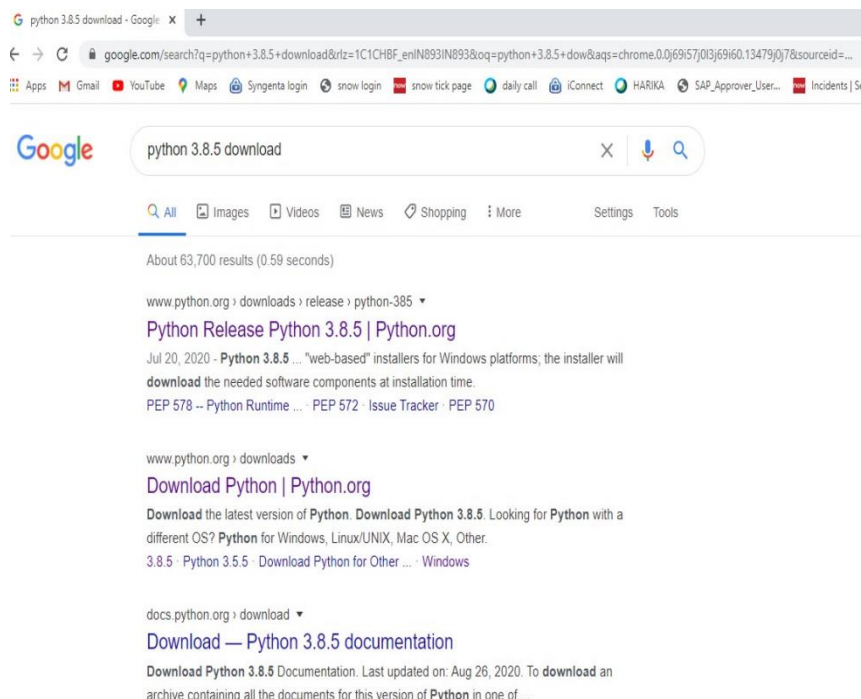


Fig 4.1 python installation

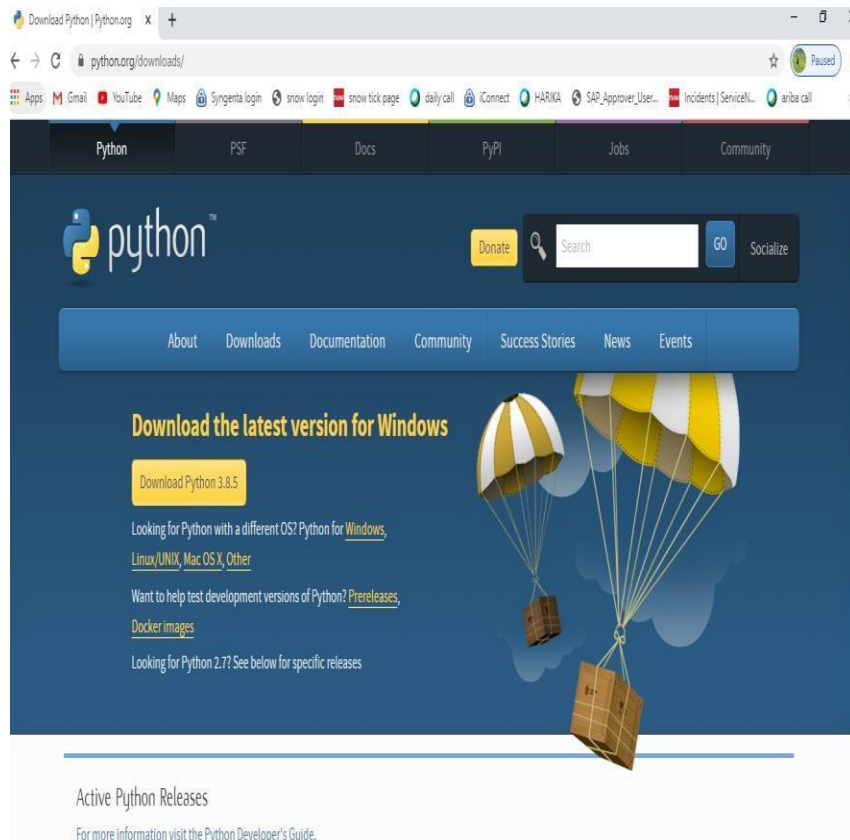


Fig 4.2 python downloading

4.1.2 Open CV

Open CV (open source computer vision). it is a library of programming functions mainly aimed at real time computer vision. it is used in modern technologies in such a way like face recognition, multiple color detection, object detection and so on. open cv was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. the library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. open cv leans mostly towards real time vision applications. here is the installation of the open cv and its steps to follow:

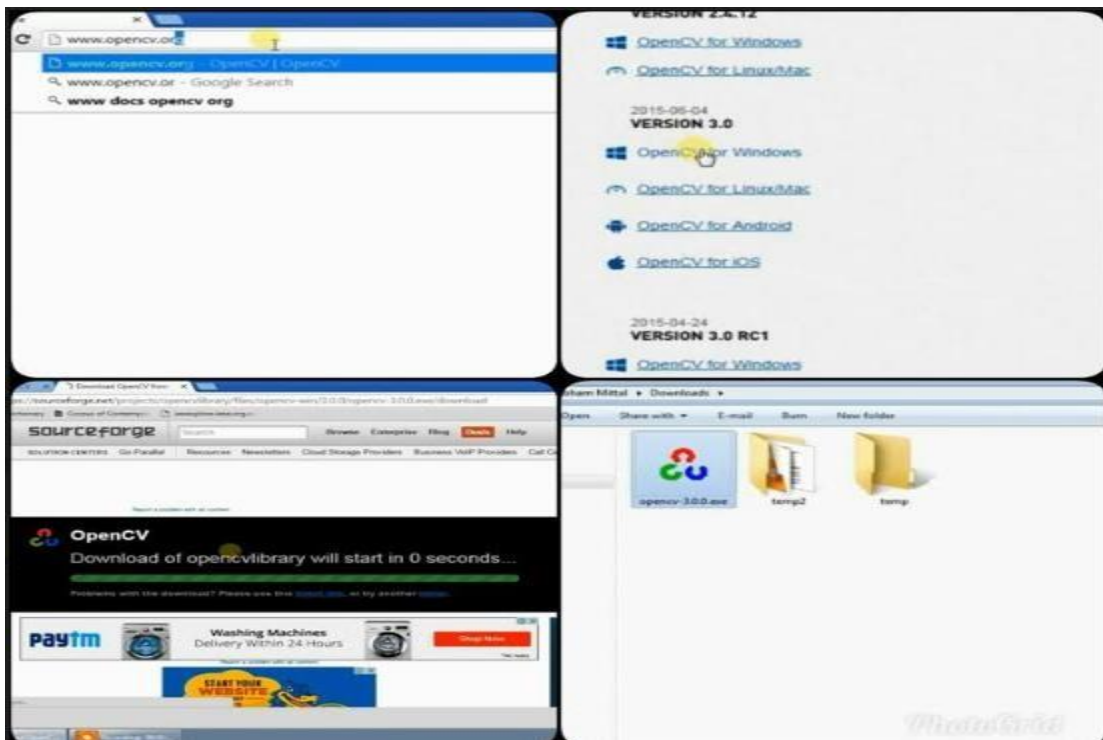


Fig 4.3 Opencv Downloading

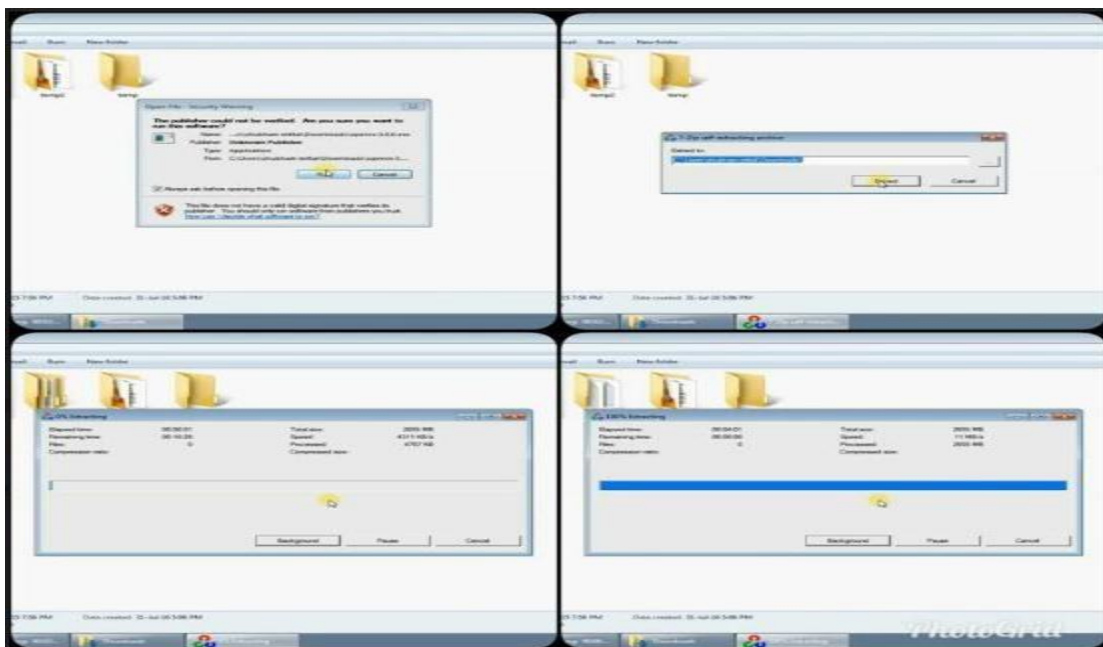


Fig 4.4 Opencv Installation

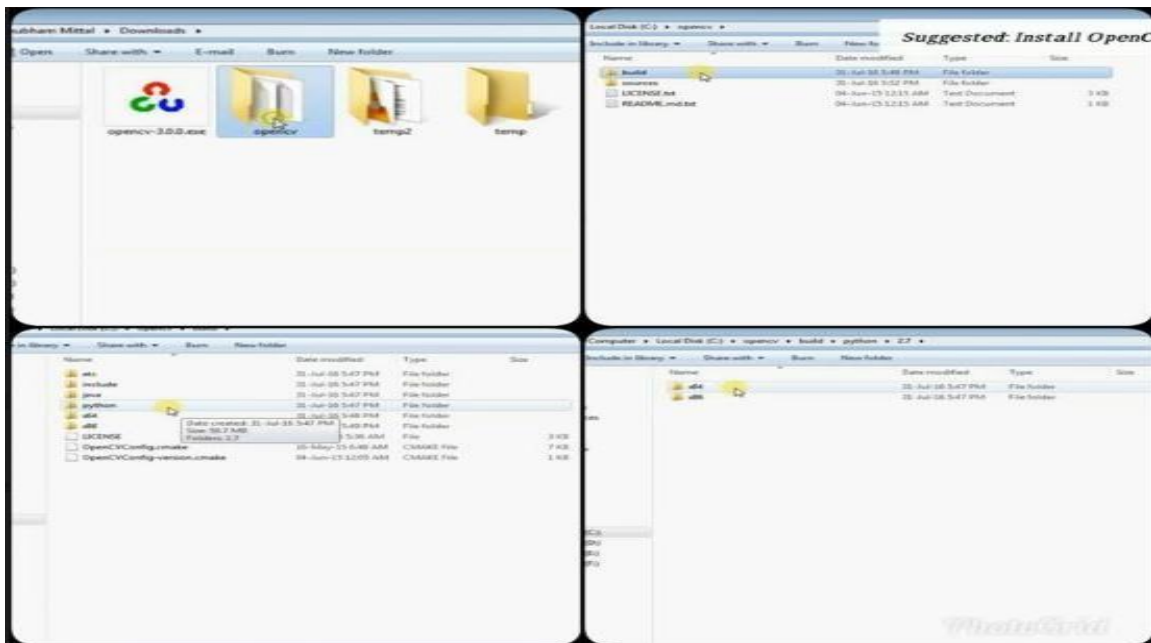


Fig 4.5 Package Installation

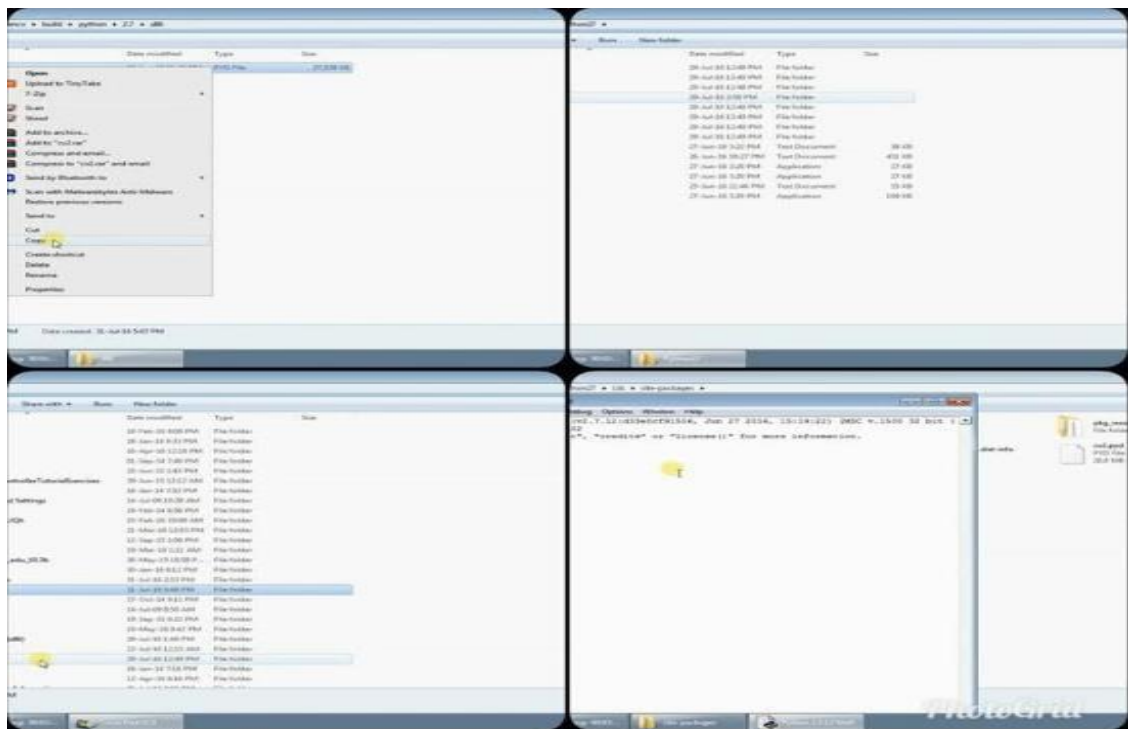


Fig 4.6 Opencv Shell

The main libraries used in the open cv on the project named multiple color detection are as follows:

Camera Settings:

In order 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.

Capturing frames:

The infinite loop is used so that the webcam 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, one uses the function `cv2.cvtColor(input_image, flag)` where flag determines the type of conversion. For BGR to HSV use the flag `cv2.COLOR_BGR2HSV`. Now we know how to convert BGR image to HSV, one 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, the range of blue color is specified. Whereas you can enter the range of any color you wish.

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.

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.

Display an image:

Use the function cv2.imshow() to display an image in a window. The window automatically fits to the image size.

cv2.waitKey():

wait key is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke which can also be set to detect specific key strokes like, if key a is pressed etc.

cv2.destroyAllWindows():

simply destroys all the windows we created. If you want to destroy any specific window, use the function cv2. destroyAllWindows() where now pass the exact window name as the argument.

Morphology:

Morphology is a non-linear operation that modifies the shape or morphology of objects in an image using a structuring element.

Basic morphology operations are:

- Dilation
- Erosion

- Opening
- Closing

open CV:

OpenCV allows us to open an image and store it in a 3-dimensional array or matrix where the x and y axis designate the location of the pixel in the image and the z axis designates the RGB color channel.

Each RGB pixel contains an 8-bit red component, an 8 bit green component, and an 8 bit blue component. When combined they give us the colour of the pixel in question.

An RGB value of (255, 255, 255) means that the intensity of each colour component in this pixel is 255, which is the maximum intensity that the 8 bit value can be in decimal.

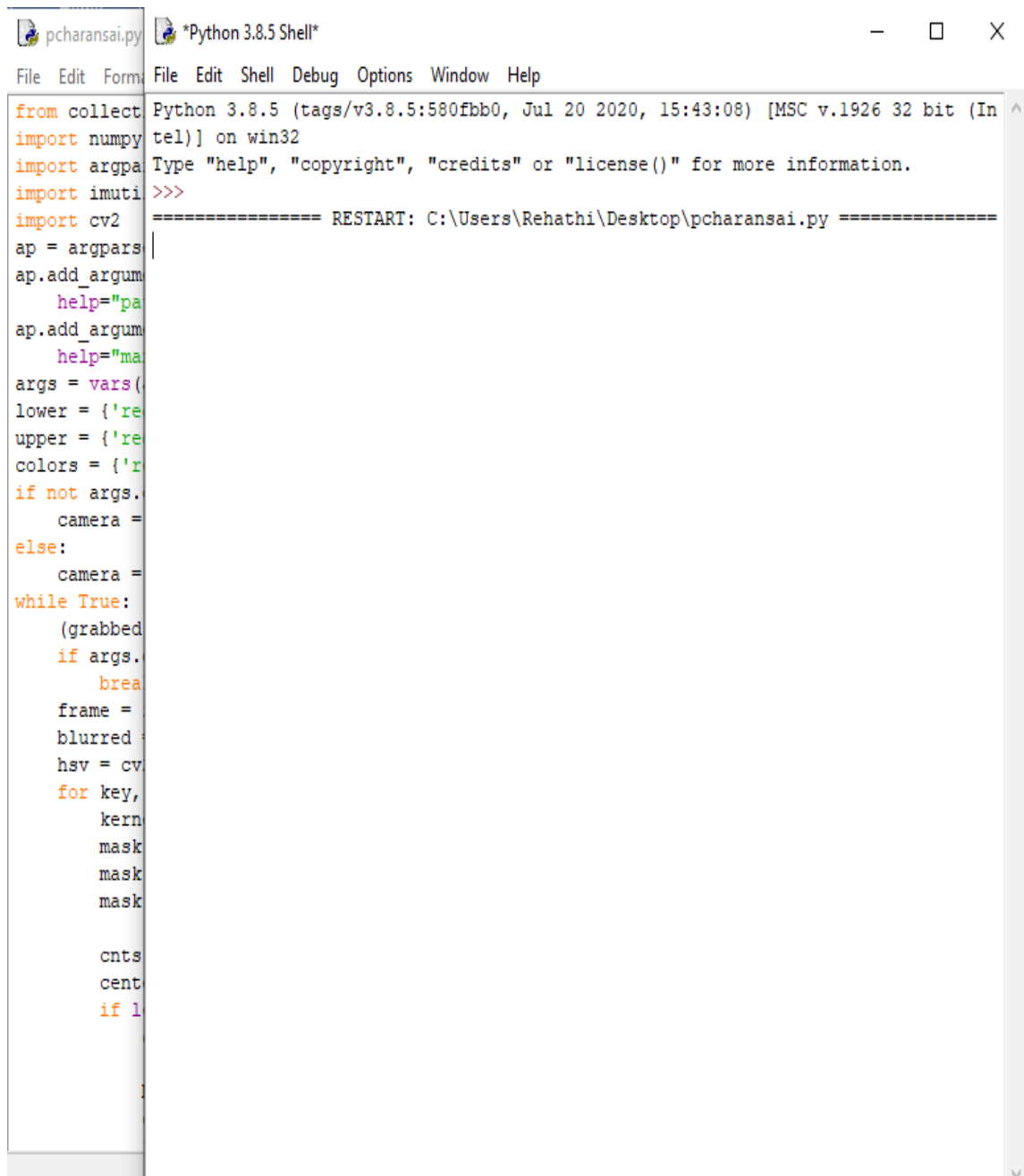
The RGB value (255,255,255) designates the color white as our eyes would see. At the other end of the RGB spectrum when all RGB components are at intensity 0 that is (0,0,0) the screen would render a black pixel.

4.2 Summary

Take each frame of the video and convert each frame from BGR to HSV color- space. there are the few challenges in the opencv they are:

- Loss of information caused by the 3D world on a 2D image.
- Noise in images.
- Complex object motion.
- Non rigid or articulated nature of objects.
- Partial or full object occlusions.
- Complex object shapes.
- Scene illumination changes.

RESULTS AND DISCUSSIONS

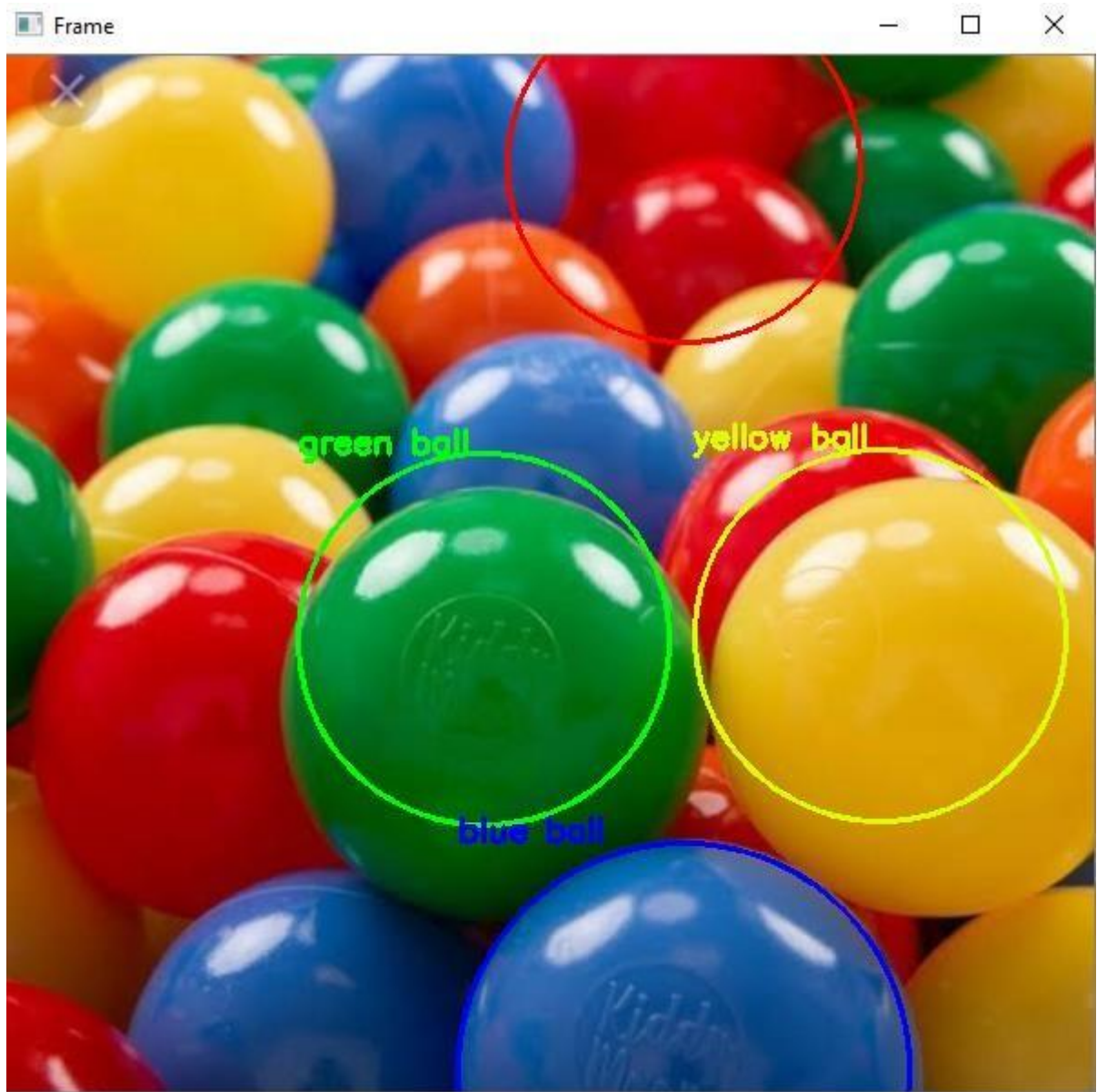


```
File Edit Form File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Rehathi\Desktop\pcharansai.py =====
from collect
import numpy
import argpa
import imuti
import cv2
ap = argpars
ap.add_argum
    help="pa
ap.add_argum
    help="ma
args = vars(
lower = {'re
upper = {'re
colors = {'r
if not args.
    camera =
else:
    camera =
while True:
    (grabbed
    if args.
        brea
    frame =
    blurred :
    hsv = cv
    for key,
        kern
        mask
        mask
        mask

    cnts
    cent
    if 1
```

From the above diagram the video is captured into the frame and then based on the type the colour of the balls is displayed in circular shape. The Frame is captured and then the colour is given. In other instance when there is a image or the video already provided and is grabbed through the urllib library using a

url provided will capture the frame automatically and the colour are directly displayed on the screen with the image given in the url which will be shown below in the diagram



As shown in the above diagram the colours are detected and displayed as blue, green, red colours and will be displayed on the output screen on the frame as shown above.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

With the steps shown in the above section we can easily detect and track color using tools like Python. The above algorithm is simple and easy to implement with high percentage of accuracy.

As of now one can easily conclude that color detection and tracking is one of the key steps towards the future that will be dominated by artificial intelligence. Color detection and tracking has many roles in various fields which also include the medical science.

A lot of researches have already been conducted and many are still going on in the field of object tracking which is a key in the field of artificial intelligence. Further there is a lot of work still left to be done so that the scope of object tracking and its uses could be broadened.

A reliable detection frame work for global symmetries using log-gabor wavelet response. introduction of symmetric metrics using textual and color information around wavelet features. superior performance over single and multiple symmetries over all public datasets.

5.2 Future work

The proposed detection can be improved using a continuous maximal seeking technique to avoid over extended axis.

possibility of integration with in retrieval systems for artistic photographs and paintings in museums.

we have developed an efficient system to match cloths with multiple colors and complex patterns to assist visually impaired and blind people by distinguish both pattern and color information.

REFERENCES

- Kalisa Wilson, Real- Time Tracking for Multiple Objects Based on Implementation of RGB Color Space in Video, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.9, No.4, (2016), pp.331-338.
- Ssu-Wei Chen, Luke K. Wang, Jen-Hong Lan, Moving Object tracking Based on Background Subtraction Combined Temporal Difference, International Conference on Emerging Trends in Computer and Image Processing (ICETCIP'2011) Bangkok Dec., 2011.
- Vandana S. Bhata* and Jagadeesh D. Pujaria, Face detection system using HSV color model and morphing operations, International Journal of Current Engineering and Technology.
- Ravikant Gupta, Satyaprakash Pandey Yogesh Tayal, Pramod Kumar Pandey, D.V.B Singh, Human Face Detection by YCbCrHs Technique, International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS).
- Prasad Kalane, Target Tracking Using Kalman Filter, International Journal of Science & Technology, Vol. 2 Issue 2, April 2012d.
- Isaac Cohen G´erard Medioni, Detecting and Tracking Moving Objects for Video Surveillance, IEEE Proc. Computer Vision and Pattern Recognition Jun. 23-25, 1999. Fort Collins CO.
- Andres Alarcon Ramirez and Mohamed Chouikha, A New Algorithm for Tracking Objects in Videos of Cluttered Scenes, International Journal of Information Technology, Modeling and Computing (IJITMC) Vol.1, No.2, May 2013.
- Chris Harris & Mike Stephens, A COMBINED CORNER AND EDGE DETECTOR, Plessey Research Roke Manor, United Kingdom © The Plessey Company pic. 1988.
- Katja Nummiaro, Esther Koller-Meier, Tom´ aš Svoboda, Daniel Roth, and Luc Van Gool, Color-Based Object Tracking in Multi-Camera Environments, In Proceedings of the DAGM'03, Springer LNCS 2781, pp. 591-599, Sep 2003.

- Marcus Thaler, Werner Bailer, Real-time Person Detection and Tracking in Panoramic Video, JOANNEUM RESEARCH, DIGITAL – Institute for Information and Communication Technologies Steyrergasse 17, 8010 Graz, Austria.
- Wen-cheng Wang, A Face Detection Method Used for Color Images, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 8, No. 2 (2015), pp. 257-266
- Hani K. Al-Mohair, Junita Mohamad-Saleh and Shahrel Azmin Suandi, Human skin color detection: A review on neural network perspective, International Journal of Innovative Computing, Information of Control Volume 8, Number 12 December 2012.
- Monika Deswal, Neetu Sharma, A Fast HSV Image Color and Texture Detection and Image Conversion Algorithm, International Journal of Science and Research (IJSR).
- Douglas Chai, Face Segmentation Using Skin-Color Map in Videophone Applications, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 9, NO. 4, JUNE 1999.
- S. Chitra, Comparative Study for Two Color Spaces HSCbCr and YCbCr in Skin Color Detection, Applied Mathematical Sciences, Vol. 6, 2012, no. 85, 4229 – 4238.
- Intaek Kim, Malik Muhammad Khan, Tayyab Wahab Awan, and Youngsung Soh, Multi-Target Tracking Using Color Information, International Journal of Computer and Communication Engineering, Vol. 3, No. 1, January 2014.
- Akriti Kaushik¹, Aastha Sharama, RGB COLOR SENSING TECHNIQUE, International Journal of Advance Research in Science and Engineering.
- Hasan Fleyeh, Road and Traffic Sign Color Detection and Segmentation A Fuzzy Approach, MVA2005 IAPR Conference on Machine Vision Applications, May 16-18, 2005 Tsukuba Science City, Japan.

APPENDIX

A .SOURCE CODE:

```
#python color_tracking.py --video balls.mp4

#python color_tracking.py

# import the necessary packages

from collections import deque #to append or pop elements

import numpy as np #multidimensional arrays and matrices

import argparse

import imutils

import cv2

import urllib #for reading image from URL

# construct the argument parse and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-v", "--video",

    help="path to the (optional) video file")

ap.add_argument("-b", "--buffer", type=int, default=64,

    help="max buffer size")

args = vars(ap.parse_args())

# define the lower and upper boundaries of the colors in the HSV color space
```

```
lower = {'red':(166, 84, 141), 'green':(66, 122, 129), 'blue':(97, 100, 117), 'yellow':(23, 59, 119), 'orange':(0, 50, 80)} #assign new item lower['blue'] = (93, 10, 0)
```

```
upper = {'red':(186,255,255), 'green':(86,255,255), 'blue':(117,255,255), 'yellow':(54,255,255), 'orange':(20,255,255)}
```

```
# define standard colors for circle around the object
```

```
colors = {'red':(0,0,255), 'green':(0,255,0), 'blue':(255,0,0), 'yellow':(0, 255, 217), 'orange':(0,140,255)}
```

```
#pts = deque(maxlen=args["buffer"])
```

```
# if a video path was not supplied, grab the reference
```

```
# to the webcam
```

```
if not args.get("video", False):
```

```
    camera = cv2.VideoCapture(0)
```

```
    # otherwise, grab a reference to the video file
```

```
else:
```

```
    camera = cv2.VideoCapture(args["video"])
```

```
# keep looping
```

```
while True:
```

```
    # grab the current frame
```

```
    (grabbed, frame) = camera.read()
```

```
    # if we are viewing a video and we did not grab a frame,
```

```
    # then we have reached the end of the video
```

if args.get("video") and not grabbed:

break

#IP webcam image stream

#URL = 'http://10.254.254.102:8080/shot.jpg'

#urllib.urlretrieve(URL, 'shot1.jpg')

#frame = cv2.imread('shot1.jpg')

resize the frame, blur it, and convert it to the HSV

color space

frame = imutils.resize(frame, width=600)

blurred = cv2.GaussianBlur(frame, (11, 11), 0) #contours

hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

#for each color in dictionary check object in frame

for key, value in upper.items():

 # construct a mask for the color from dictionary`1, then perform

 # a series of dilations and erosions to remove any small

 # blobs left in the mask

 kernel = np.ones((9,9),np.uint8)#white

 mask = cv2.inRange(hsv, lower[key], upper[key])

 mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

 mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

```

# find contours in the mask and initialize the current

# (x, y) center of the ball

cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,

center = None

    # only proceed if at least one contour was found

if len(cnts) > 0:

    c = max(cnts, key=cv2.contourArea)

    ((x, y), radius) = cv2.minEnclosingCircle(c)

    M = cv2.moments(c)

    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

    if radius > 0.5:

        # draw the circle and centroid on the frame,

        # then update the list of tracked points

        #cv2.circle(frame,(384,0),63,(0,255,0),3)

        cv2.circle(frame, (int(x), int(y)), int(radius), colors[key], 2)

        cv2.putText(frame,key + " ball", (int(x-radius),int(y-radius)),

# show the frame to our screen

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

```

```
# if the 'q' key is pressed, stop the loop

if key == ord("q"):

    break

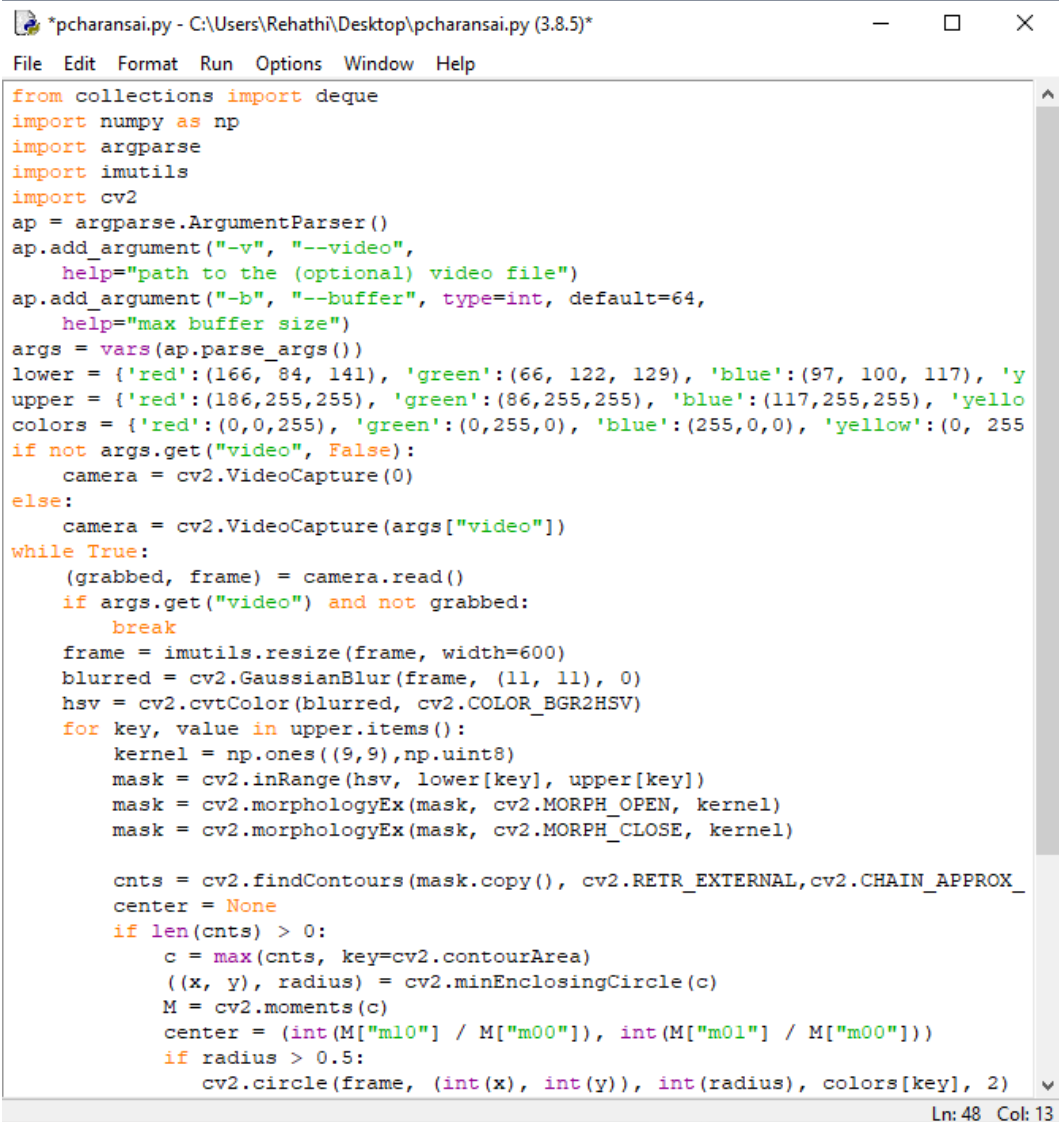
# cleanup the camera and close any open windows

camera.release()

cv2.destroyAllWindows()
```

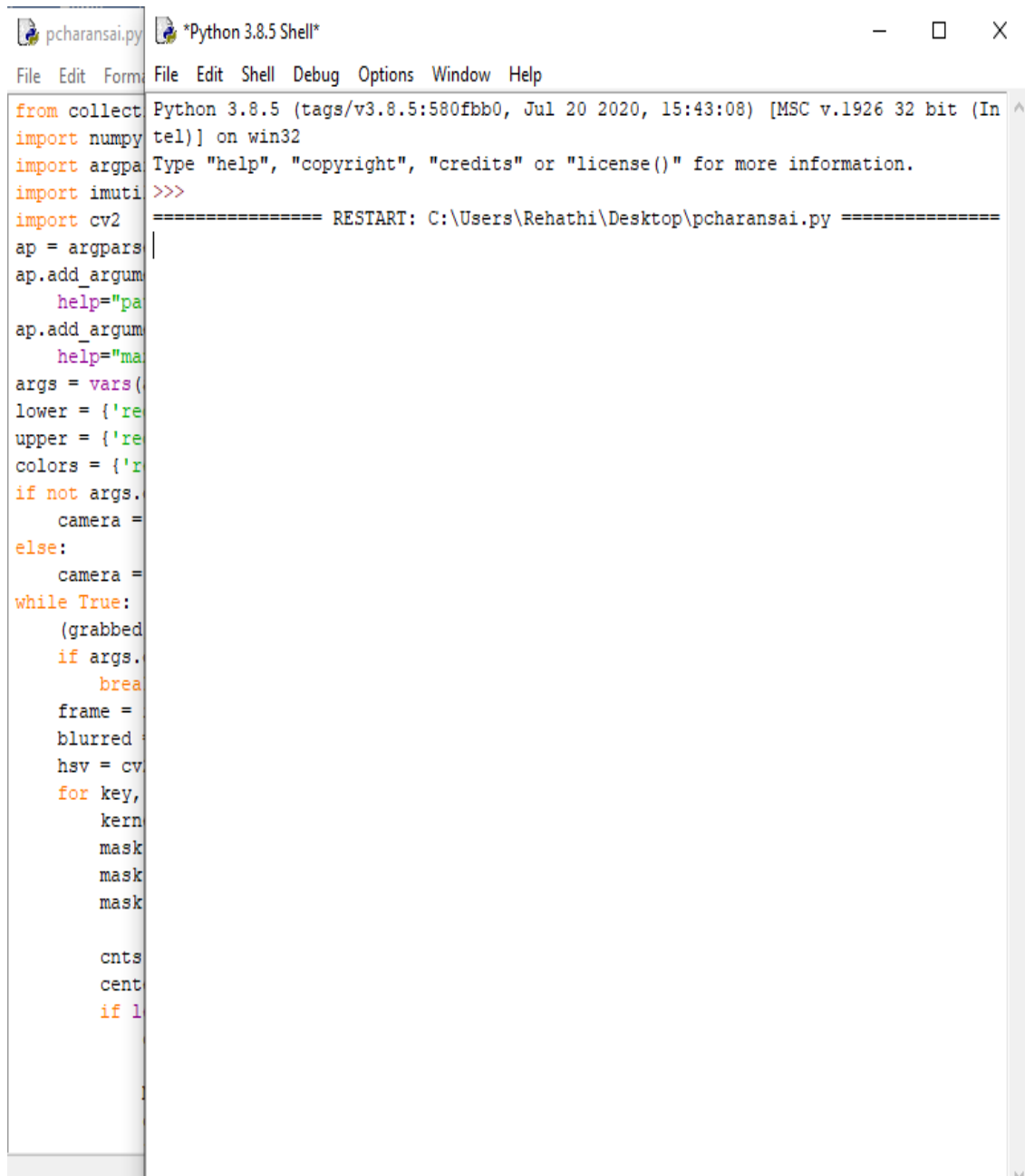
B .SCREEN SHOTS

Output:



```
*pcharansai.py - C:\Users\Rehathi\Desktop\pcharansai.py (3.8.5)*
File Edit Format Run Options Window Help
from collections import deque
import numpy as np
import argparse
import imutils
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video",
    help="path to the (optional) video file")
ap.add_argument("-b", "--buffer", type=int, default=64,
    help="max buffer size")
args = vars(ap.parse_args())
lower = {'red':(166, 84, 141), 'green':(66, 122, 129), 'blue':(97, 100, 117), 'y
upper = {'red':(186,255,255), 'green':(86,255,255), 'blue':(117,255,255), 'yellow
colors = {'red':(0,0,255), 'green':(0,255,0), 'blue':(255,0,0), 'yellow':(0, 255
if not args.get("video", False):
    camera = cv2.VideoCapture(0)
else:
    camera = cv2.VideoCapture(args["video"])
while True:
    (grabbed, frame) = camera.read()
    if args.get("video") and not grabbed:
        break
    frame = imutils.resize(frame, width=600)
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
    for key, value in upper.items():
        kernel = np.ones((9,9),np.uint8)
        mask = cv2.inRange(hsv, lower[key], upper[key])
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_
    center = None
    if len(cnts) > 0:
        c = max(cnts, key=cv2.contourArea)
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        if radius > 0.5:
            cv2.circle(frame, (int(x), int(y)), int(radius), colors[key], 2)
Ln: 48 Col: 13
```



```
from collect
import numpy
import argparse
import imutils
import cv2

ap = argparse.ArgumentParser()
ap.add_argument(
    '-h', '--help',
    help='pa
ap.add_argument(
    '-m', '--ma
args = vars(
lower = {'re
upper = {'re
colors = {'r
if not args:
    camera =
else:
    camera =
while True:
    (grabbed
    if args:
        brea
    frame =
    blurred =
    hsv = cv
    for key,
        kern
        mask
        mask
        mask

    cnts
    cent
    if 1
```



