

SYNOPSIS

SYNOPSIS

Object detection is the process of detecting an object in an image. Object detection presents a challenging problem in the field of identifying an object in an image by using computer vision techniques with machine learning libraries in python. A computer vision technique is the process of finding number of instances in an image. Object detection is a subset of object recognition. These object detection not only identified an object, but also located in an image. These object detection has salient features like detecting multiple objects within the same image. It has a ability to count objects in an image(scene).Here, object detection is done using HSV Color space with machine learning libraries namely OpenCV and numpy.

ABSTRACT

The project is entitled as "Object detection using HSV color space with machine learning libraries in python". Where it going to have single RGB input image and output is going to be a set of detected object. Object detection is a computer vision technique it is the process of finding instances of objects in image. Object detection it is a subset of object recognition, where the object is not only identified but also located in an image. In addition it allows identifying and locating multiple objects within the same image. Object detection is used to count objects in a scene. Here object detection is done by using "HSV color space" with machine learning libraries namely "OpenCV and numpy " in python. Now a day s object detection plays a vital role in the field of "Autonomous driving, Animal detection in agriculture, Object detection in retail, People detection in security, Vehicle detection with Artificial intelligence (AI) in transportation, Medical feature detection in healthcare, Vehicle counting, Image annotations, Activity recognition, Face recognition, face detection, Vehicle object co-segmentation.

SYSTEM SPECIFICATION

1.1 SYSTEM SPECIFICATION

1.1.1 SOFTWARE SPECIFICATION:

Operating system. : Windows 10
Programming language : Python
Interface : IDLE
Packages : Opencv,numpy

1.1.2 HARDWARE SPECIFICATION:

Processor : AMD Dual Core A6 9225
RAM : 2133 MHZ
Hard disk : 1 TB
Processor Speed : 2.6 GHZ
Display : 15.6"(39.62 cm),1366*768 PX

SYSTEM STUDY

2. SYSTEM STUDY

Analysis may be defined as the process of dividing the project into parts, identifying each part and establishing relationships between them. Analysis is a detailed study of various operations performed by a system and their relationship within and outside of the system. One aspect of analysis is identifying the boundaries of system and determining whether or not a candidate system should consider other related system. In the specific content of users, system analysis comprises of taking known facts concerning a system breaking them into elements and establishing logical relationship between the elements with the objective of producing the specification requirements.

Analysis is a continued activity at all stages of the project. It is the process of studying the problem to find the feasible solution to that problem, by which the existing system is learnt and understood, objectives and requirements are the system defined and the solution are evaluated. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 EXISTING SYSTEM:

The existing system was handled by HSV color space .So to overcome all this problems in object detection, the Computerized Machine learning libraries and computer vision techniques is used in the proposed system.

2.1.1 DRAWBACKS OF RGB COLOR SPACE:

- Mixing of colour information and luminance, same channels encode color and brightness information.
- It can't be used for effective segmentation of Color.
- Example: Low light and bright light colors will be read as different.

2.1.2 DRAW BACKS OF LAB COLOR SPACE:

- Mixing of colour information and luminance, same channels encode color and brightness information.
- 'A' range only from green to Magenta.
- 'B' range only from blue to yellow.

2.2 PROPOSED SYSTEM:

The proposed system was handled by Computerized Machine learning libraries with computer vision techniques using HSV color space in python. It gives the accurate result in object detection. It is able to detect multiple objects within a same image. It has main features like count objects in scene. It can detect object in a images like object detection in retail, Medical feature detection in healthcare, surveillance camera, Vehicle counting in traffic, Autonomous driving., etc.

2.2.1 FEATURES:

- Consume less man power and Takes low memory
- Accurate result
- Saves human power and time
- Fast and convenient and Easy to use

2.2.2 APPLICATION AREA USE THIS OBJECT DETECTION METHOD:

- Object detection in retail
- Autonomous driving
- Animal detection in agriculture
- People detection in security
- Vehicle detection with AI (Artificial Intelligence) in transportation
- Medical feature detection in healthcare
- Image annotations
- Vehicle counting
- Activity recognition
- Face recognition
- Face detection
- Video object co-segmentation

■ Face recognition and Face detection



■ Medical feature detection in healthcare



■ Image annotations



■ Autonomous driving



■ Object detection in retail



■ Vehicle counting



2.2.3 ALGORITHM :

To detect an object in an image, algorithm are used namely.

- OpenCV
- Numpy
- HSV color space
- Thresholding
- Masking
- Bitwise And operation

2.2.4 ADVANTAGES OF HSV COLOR SPACE:

It resembles the way humans perceive the colors.widely used for the purposes and projects where the color segmentation is required. It gives the accurate result compared to RGB and LAB color space. It is used for image processing and design an real world applications related to color space.

LANGUAGE SPECIFICATION

2.3 LANGUAGE SPECIFICATION:

Python is high level language developed by Guido Van Rossum at Centrum Wiskunde and Informatica (CWI) it is a National research institute for mathematics and computer science, Netherlands in late 1980's. He named that programming language python, because he is a big fan of tv show called " Monty Python's " flying circus and not after the python the snake. Rossum published the first version of python code (0.9.0) in February 1991 at the Centrum Wiskunde Informatica (CWI) in the Netherlands, Amsterdam. Python is derived from ABC programming language, which is a general purpose programming language that had been developed at CWI. Python is now maintained by a core development team at the institute, although Rossum still holds a vital role in directory it's progress.

HISTORY OF PYTHON:

Python 1.0 was released in 1994. Python 1.5 was released in 1997. Python 1.6 was released in 2000. Python 2.0 was released in 2000. Python 2.1 was released in 2001. Python 2.2 was released in 2002. Python 2.3 was released in 2003. Python 2.4 was released in 2004. Python 2.5 was released in 2006. Python 2.6 was released in 2008. Python 2.7 was released in 2010. Python 2.7.15 was released in May 2018. Python 3.0 was released in December 2020. Python 3.1.0 was released in 2022.

ABOUT PYTHON:

Python is very simple and straightforward Syntax. Python is case sensitive. It is object oriented language and dynamically typed. Indentation is used in place of curly braces. We can use variable without declaration.

FEATURES:

Emphasis on code readability. Automatic memory management. Large library. Multi-paradigm programming language (object oriented, procedural, etc). Interactive interpreter and Platform independent (Apple, Microsoft, Mac OS, Linux, Ubuntu, MS-DOS, Windows server). It is free and open-source. It is embedded and extensible and portable.

LIBRARIES AVAILABLE:

Graphical user interface, Web frameworks, Multimedia, Databases, Networking, Test frameworks, Automation, web scripting, Documentation, System administration, Scientific computing, Text processing, Image processing, IOT, 3D CAD applications, Enterprise applications, Audio or video based application, Game development.

DESCRIPTION OF MODULES

3. DESCRIPTION OF MODULES

3.1 LOAD AN IMAGE DATA:

To load an image, take an image with png extension and the image be an coloured image. Because object detection is possible only for an high resolution and then coloured image(i.e., 800x600) and create a new window to show that image inside the window. Load an image with the help of OpenCV 's default function.

3.2 CONVERT COLORED IMAGE INTO BGR IMAGE:

Now to process that loaded colored image, it was first converted into BGR image. Basically computer see images as RGB(Red,Green,Blue) format then convert to grey scale(black (0) and white (1)).But here, OpenCV packages use BGR(Blue,Green,Red) format, OpenCV automatically take values of loaded image in BGR format.

3.3 CONVERT BGR IMAGE INTO HSV COLOR SPACE IMAGE:

This module convert the BGR image format into HSV color space image format by using default function in OpenCV packages.HSV stands for Hue saturation value of an image. Hue corresponds to the color components (base pigment).Saturation is the amount of color(depth of pigment).Value is basically the brightness of the color.

3.4 CREATE A TRACK BAR TO ADJUST THE HUE, SATURATION, VALUE OF AN HSV IMAGE:

Now, create a track bar it contains the value of lower boundation, lower hue, lower saturation, lower value. Likewise upper boundation, upper hue, upper saturation, upper value for HSV color space. This track bar is used to adjust the lower bound, upper bound parameter values namely lh,ls,lv and uh,us,uv. By using this trackbar multiple objects can be detected within a same image.

3.5 APPLY THRESHOLD VALUE TO THE COLOR RANGE OF HSV IMAGE:

Now, apply Threshold value to that track bar variables created namely lower boundation as "lb", lower hue as "lh", lower saturation as "ls", lower value as "lv". Similarly upper boundation as "ub", upper hue as "uh", upper saturation as "us", and upper value as "uv". Lower bound and upper bound are calculated by taking image into form of matrix. With the help of "numpy" packages it take images into form of array.

EXAMPLE:[110,50,50] as lh,ls,lv.

3.6 APPLY MASK TO THE COLOR RANGE OF HSV IMAGE:

Now, apply mask function to the color range of an HSV image. By passing the lb, lh, ls, lv and ub, uh, us, uv variables to that function. A special masking function in OpenCV called mask in range allows to mask image with the color range of HSV image.

3.7 APPLY BITWISE AND OPERATION ON THE COLOR RANGE OF HSV IMAGE:

Now, apply "bitwise and operation" on the image. Binary Operation "bitwise_and" is used to give the accurate result in masking an image. It mask the values from the range of HSV namely (lower boundation and upper boundation). Then create a new variables and then assign the source 1, source2, and mask values in it.

3.8 DISPLAYING THE DETECTED OBJECT:

Now, show the detected object in an image, by using default function in OpenCV packages. With track bar, we can detect multiple objects within the same image. Imshow function is used to display an image in a window. This window automatically fits to the image.

CONCLUSION

4. CONCLUSION:

This new system eliminates difficulties in object detection by existing system. It is developed in a user defined friendly manner. The aim of the project is to detect object in an image and it has an ability detect multiple objects within a same image. With special features trackbar in OpenCV can detect specific object in an image. It gives accurate object detection.

ADVANTAGES:

- Detected multiple objects within the same image
- Detected specified object in an image using trackbar window
- Accurate result in object detection
- Display detected object in as a colored image

BIBLIOGRAPHY

5. BIBLIOGRAPHY:

REFERENCE BOOKS:

- "Python Crash Course" by "Eric Matthews"
- " Learn python 3 the hard way" by "Zed A. Shaw"
- " Fluent python" by " Luciano Ramalho"
- " Python Codebook" by " David Beazely"
- " Programming python" by " Mark Lutz"
- " Introduction to machine learning with python" by " Sarah Guido and Andreas"

WEBSITES:

- <http://books.google.com>
- <http://www.programiz.com>
- <http://www.w3schools.com>
- <http://www.geeksforgeeks.org>
- <http://www.tutorialspoint.com>
- <http://docs.python.org>

APPENDICES

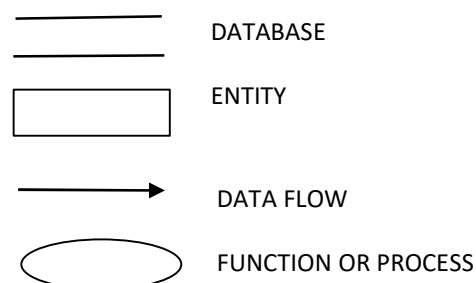
DATA FLOW DIAGRAM

A. DATA FLOW DIAGRAM

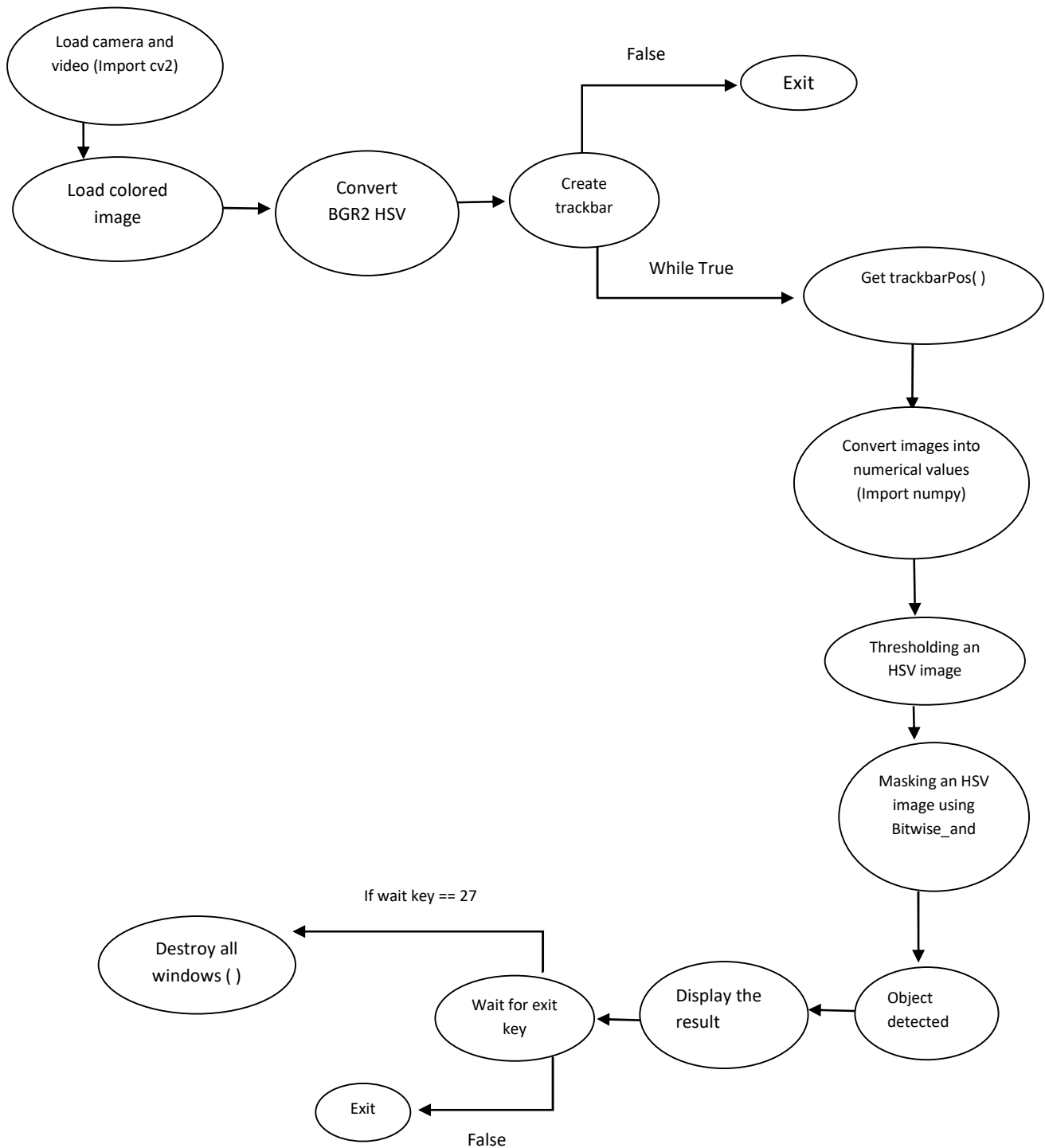
System analyst may use one or more of the following traditional and structured analysis tools to represent system data and process graphically.

1. Data flow diagram(DFD)
2. System flow chart

A DFD is the graphical representation of the flow of data from one component to another component in any information system. Through DFD, we can give the overview of the system without going into the deep detail of the system. DFD also called as "Bubble Chart". It also express the requirement of the system and shows how the current system is implemented. The DFD takes an input-process-output view of a system.(i.e.,) data objects flow into the software are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labelled arrows and transformation are represented by circles also called as bubbles. DFD is presented in hierarchical fashion.i.e., The first data flow model represents the system as a whole . Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level. The DFD enables the software engineers to develop models of the information domain and functional domain at the same time. As the DFD is defined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embodies the application. A context - level DFD for the system the primary external entities produce information for use by the system and consume generated by the system. The labelled arrow represents data objects or object hierarchy.



1. DATA FLOW DIAGRAM:



2. SYSTEM FLOW DIAGRAM

SYSTEM FLOW DIAGRAM:

This diagram which represents SYSTEM FLOW with some standard notations. Mostly useful for PROCESS ENGINEERS/PROGRAMMERS/DESIGNERS etc. Need to be modified if process sequence is changed. Sequence -- Order of Execution/work done . Sequence can be, e.g. TOP TO BOTTOM, BOTTOM TO TOP, Event driven approach. Selection -- Branching of statement depending on the condition e .g. IF,ELSE, SELECT CASE, SWITCH etc. Loop -- Repeating bunch of statements until, we get a desired result. It will be of two types. a.)Entry control loop b.) Exit control loop. Loop has three major points. a.) Start b.) End c.)++ d.)- -.

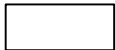
MAJOR SYMBOLS FOR CHART:



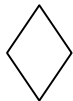
Eclipse -- used for Start/Stop Only once in a flow chart



Skew Rectangle -- used for Input / Output like print, read etc.



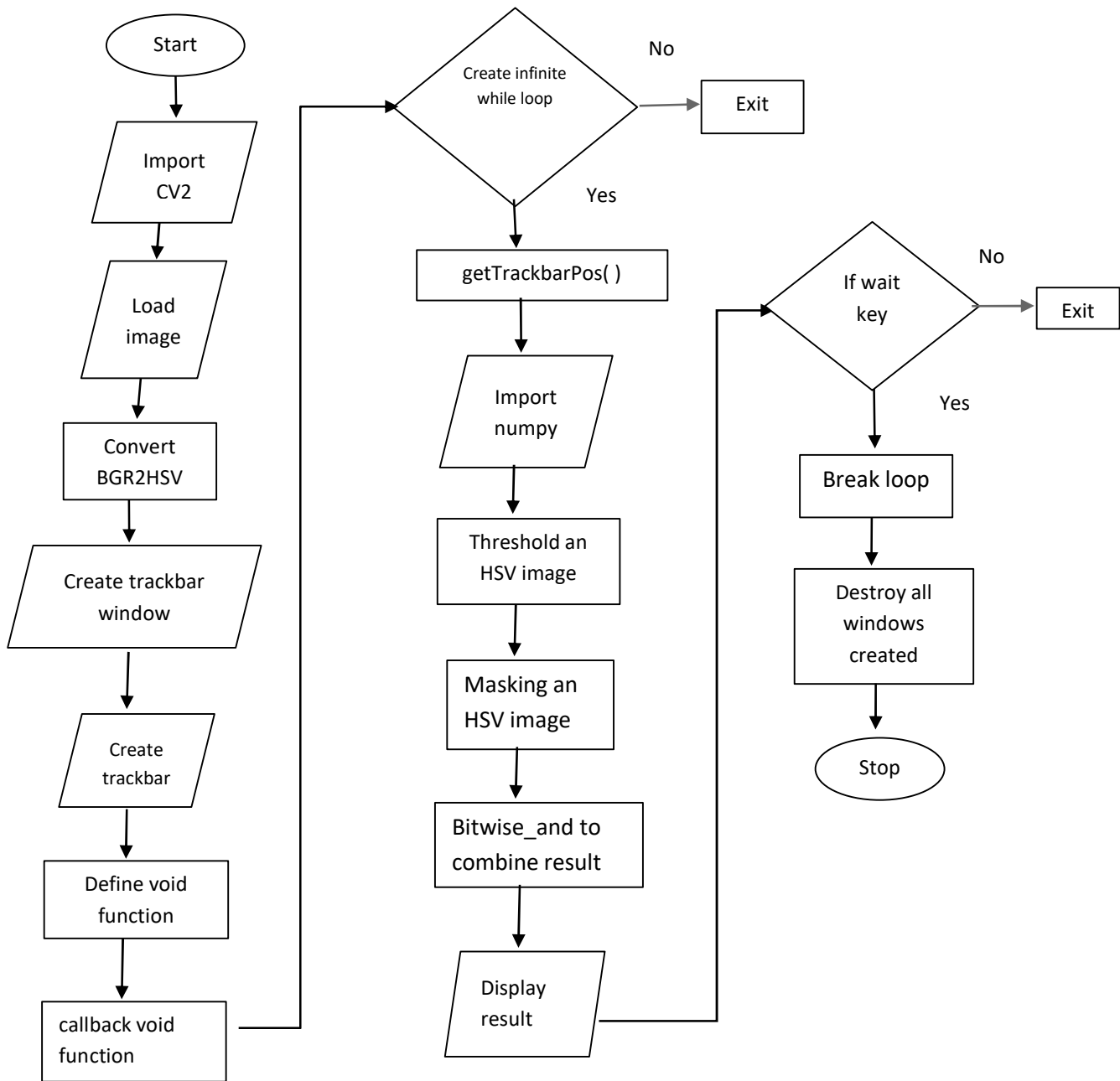
Rectangle -- used for Arithmetic processes & data storage.



Diamond -- used for Decision making . It may have one input and many

It is normally used for branching.

2. SYSTEM FLOW DIAGRAM



ALGORITHM EXPLANATION

B. ALGORITHM EXPLANATION

To detect an object in an image, algorithm are used namely.

- OpenCV
- Numpy
- HSV color space
- Thresholding
- Masking
- Bitwise And operation

1. OPENCV PACKAGE INTRODUCTION AND HISTORY:

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel. It was supposed by Willow Garage than Itseez (which was later acquired by Intel).The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011,OpenCV features GPU acceleration for real-time operations.**BSD License -(Berkley Source Distribution) low restriction type of license for open source software that does not put requirements on redistribution.** It is free for both academic and commercial use.

USE OF OPENCV:

OpenCV is mainly used for developing advanced image processing and computer vision applications. It's library has around **2500** efficient Algorithm that support standard computer vision and machine learning algorithms.

APPLICATIONS OF OPENCV:

These OpenCV algorithm let us to create the following types of applications.

- Face detection applications
- Object detection applications
- Security application tracking human actions in videos
- Application requiring tracking of camera/ object movements
- 3D modeling application
- Image matching application to find similar images from a picture database.
- Image editing application

MODULES IN OPENCV:

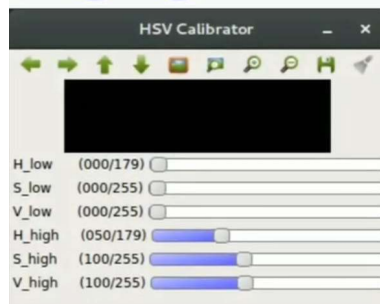
1.Core	5. ImageProc	9. Dnn	13. Ml	17. Calib3d
2.Video	6.Highgui	10.Flann	14. Photo	
3.Stitching	7.Shape	11.Superres	15. Viz	
4.Videostab	8. Imgcodecs	12 .Objdetect	16. Features2d	

USAGE OF OPENCV IN REAL WORLD:

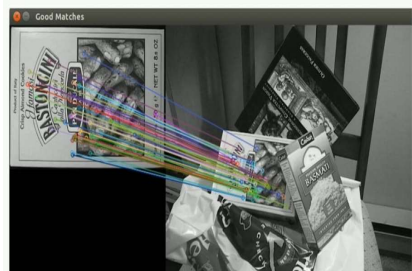
stitching



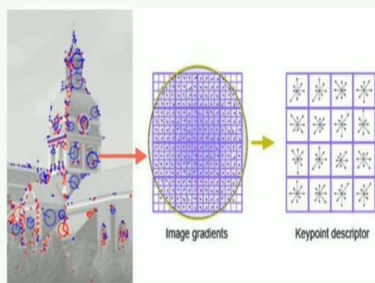
highgui



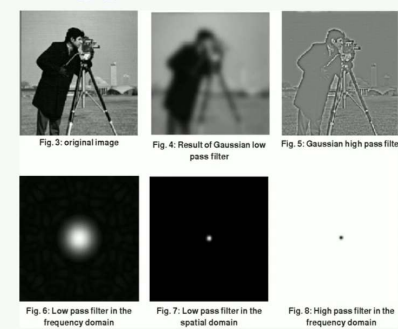
flann



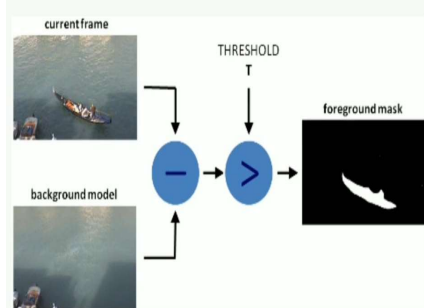
features2D

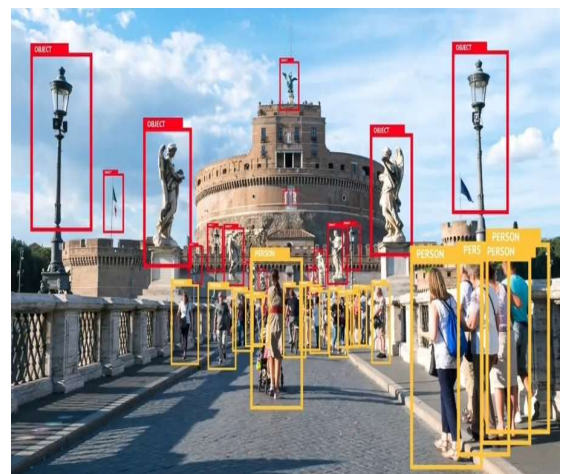
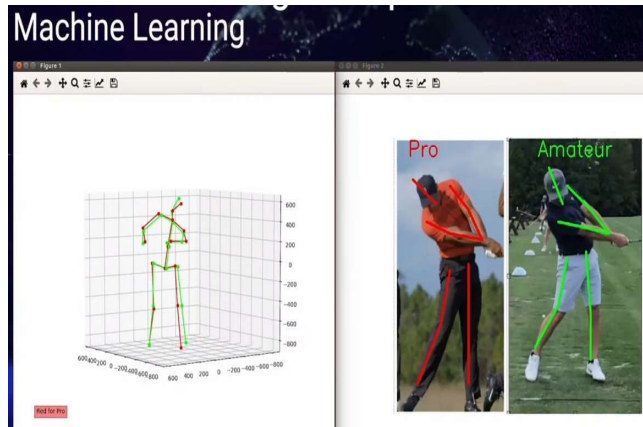


Imgproc



video





A. OPENCV IN OBJECT DETECTION:

The first step is to load an coloured image data by using "imread()" default function available in OpenCV. To store this loaded coloured image, create a variable called "frame". To read an image OpenCV 's imread function need a full location of an image. The object detection in an image is possible only for an coloured image with high clarity. So, the image going to be load in a frame must be ".png" extension. The next step is to convert it to BGR(Blue, Green, Red) format by using OpenCV packages. Because OpenCV can perform object detection process only for BGR image format.

METHOD:

```
frame=cv2.imread('C://Users//sanjivi//Pictures//car.png')
```

COMMAND:

```
Var=cv2.imread("Input_Image","flag")
```

- Imread function is the method used to read an image. It contains two arguments namely fist argument "Input image" is "Path or location of an image" , second argument is a flag which specifies the way image should be read, the values are (1,0, or -1).
- cv2.IMREAD_COLOR => Integer value(1) => Load a color image.
- cv2.IMREAD_GRAYSCALE => Integer value (0) => Loads an image in grayscale mode.
- cv2.IMREAD_UNCHANGED => Integer value (-1) => Loads image such as including alpha channel.

COMMAND USED FOR OBJECT DETECTION FROM OPENCV-PYTHON:

```
Var=cv2.imread("Input_Image","flag")
```

```
var=cv2.cvtColor("Input_Image", cv2.COLOR_BGR2HSV)
```

```
cv2.namedWindow("Window Name").
```

```
cv2.CreateTrackbar(trackbarName, windowName, value,count,onChange)
```

```
Var= cv2.getTrackbarPos("Trackbar name","Trackbar window")
```

```
var=cv2.inRange(src,lower_range,upper_range)
```

```
var= cv2.bitwise_and(scr1,src2,mask)
```

```
cv2.imshow("window name", variable name)
```



```
cv2.waitKey("value")
```

```
cv2.destroyAllWindows()
```

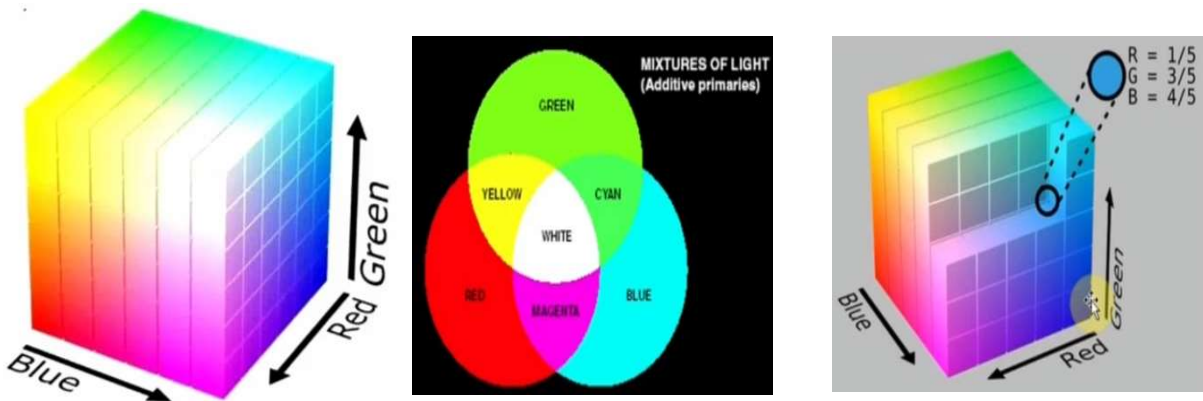
2. HSV COLOR SPACE:

Color space are way to represent color information present in an image. There are three popular color spaces are RGB, HSV, LAB.

RGB COLOUR SPACE:

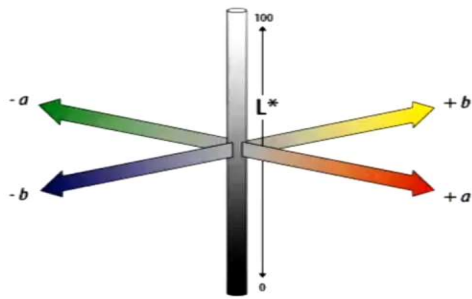
It stores information as Red, Green, Blue channels. It is an additive color model. It is used by computers to display images on monitors.

- R -> Red chroma -> reflects waves of wavelength 620-750nm.
- G -> Green chroma -> reflects waves of wavelength 495-570nm.
- B -> Blue chroma -> reflects waves of wavelength 450-495nm.



LAB COLOR SPACE:

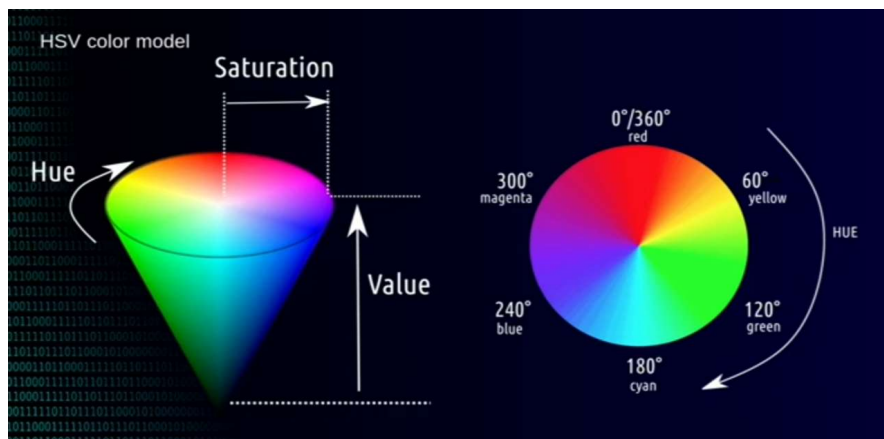
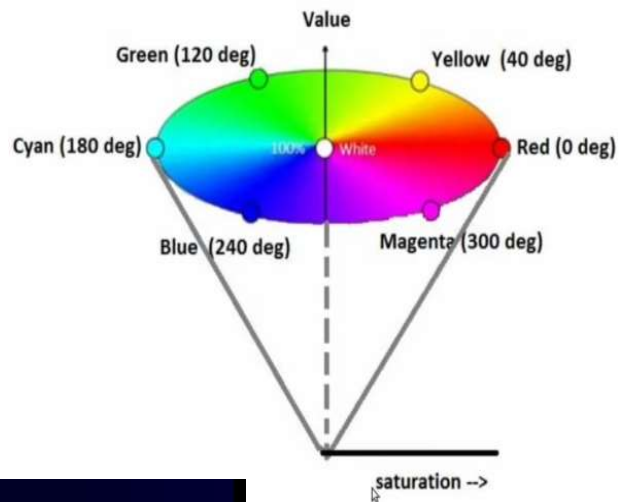
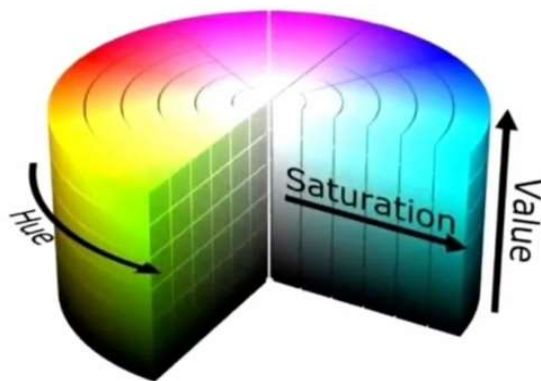
It contains three channels Luminance, a, b. Luminance states the brightness /intensity of the pixels. The parameter a states for how green or red is the pixel (green-red spectrum). The parameter b states for how blue or yellow is the pixel (blue-yellow spectrum).



HSV COLOR SPACE:

HSV stands for "Hue Saturation Value". HSV is also known as Hack Scorn Color Model. Hue corresponds to the color components (base pigment). Saturation is the amount of color (depth of pigment). Value is basically the brightness of the color (0 - black and 1 - brightest possible color (white)).

EXAMPLE: Object detection, Hand gesture recognition, Rubics Cube Solver, Color tracking, Air Canvas.



- **Hue** it is signified as a point in a 360 degree color circle.
- **Saturation** it is directly connected to the intensity of the color (range of gray in the color space). It is normally represented in terms of percentage. The range is from 0° to 100%. It signifies the intense of color presence.
- **Value** it can be called as Brightness and just like Saturation is represented as percentage. Range is from 0° and goes till 100%."0" represents black and "100%" represents the white.

Total degrees 360° in HSV color space model

- Range 0° / 360° (red)
- Range 60° (yellow)
- Range 120° (Green)
- Range 180° (Cyan)
- Range 240° (Blue)
- Range 300° (Magenta)

B.HSV COLOR SPACE IN OBJECT DETECTION:

The next step is to convert the BRG image format into HSV format by using default function in OpenCV package. The HSV is used because of its accuracy in object detection, it have a special parameter. By converting BGR image format into HSV image format it reflects on image appearance ,this HSV color space helps to process the image for object detection. To convert an BGR image format into HSV image format use the "cvtColor () method". In this method they have two parameters one is name of the variable that contain the BGR image format, another is Flag which we want to convert. Then create a variable called "hsv" to store this Converted HSV image format.

OPENCV (BGR):

[61 69 7] [54 73 11] [48 76 8]

METHOD:

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

- cvtColor is the method used to Convert colored image into any specified format from the flag.
- COLOR_BGR2BGR555
- COLOR_BGR2BGR565
- COLOR_BGR2BRGA
- COLOR_BGR2GRAY
- COLOR_BGR2HLS

- COLOR_BGR2HLS_FULL
- COLOR_BGR2HSV
- COLOR_BGR2HSV_FULL
- COLOR_BGR2LAB
- COLOR_BGR2Lab
- COLOR_GRAY2BGR555
- COLOR_GRAY2BGR565
- COLOR_GRAY2BRGA
- COLOR_GRAY2GRAY
- COLOR_GRAY2HLS
- COLOR_GRAY2HLS_FULL

TRACKBAR IN OBJECT DETECTION:

The next step is to create a trackbar for object detection. First Create a track bar window and named it as "Tracking". To create a window with name a special function is available in OpenCV, that is called "namedWindow()". After creating a trackbar window, next create a trackbar separately for each parameters in HSV color space namely lower boundation (lower hue,lower saturation, lower value) and upper boundation (upper hue,upper saturation,upper value). Using a special function available in OpenCV create a trackbar called "creatTrackbar()".

METHOD:

cv2.namedWindow("Tracking").

- namedWindow is the method used to create a window with a name
- The parameter is "Tracking" is the name of the window that going to be displayed on the screen.

cv2.createTrackbar("LH","Tracking",0,255, nothing)

cv2.createTrackbar("LS","Tracking",0,255, nothing)

cv2.createTrackbar("LV","Tracking",0,255, nothing)

cv2.createTrackbar("UH","Tracking",255,255, nothing)

cv2.createTrackbar("US","Tracking",255,255, nothing)

cv2.createTrackbar("UV","Tracking",255,255, nothing)

COMMAND:

cv2.CreateTrackbar(trackbarName, windowName, value, count, onChange)

"**createTrackbar**" create a trackbar (a slider or range control) with the specified name and range, assigns a variable called "**value**" to be a position synchronized with the trackbar and specifies the callback function "**onChange**" to be called on the trackbar position change. The created trackbar is displayed in the specified window "**window name**".

- **Create trackbar** - Function
- **TrackbarName** - Name of the created trackbar
- **windowName** - Name the window, that will be used as a parent of the created trackbar
- **value** - Optional pointer to an integer variable whose value reflects the position of the slider. Upon creation, the slider position is defined by this variable.
- **count** - Maximal position of the slider. The minimal position is always 0.
- **onChange** - Pointer to the function to be called every time the slider changes position. The function should be prototyped as " **Void Foo(int,void*);** ". Where the first parameter is the trackbar position and the second parameter is the user data. If the callback is the NULL pointer, no callbacks are called, but only "**value**" is updated. userdata - user data that is passed as it to the callback. It can be used to handle trackbar events without using global variables.

DEFINE CALLBACK FUNCTION:

The next step is to create a callback function for the trackbar. Remember that function is going to return nothing, but only "**value**" is updated. To perform operations like that special statement is used called "Pass". Pass is a null statement that does nothing. The pass statement is used to create loops, if- else statement, function and classes with an empty body.

METHOD:

def nothing:

 Pass

COMMAND:

def function_name:

 Statement (s)

- def is the keyword to define a function.
- function_name is the name of the function that we going to define.
- statement is the content inside a function.

EXPLANATION:

cv2 is the OpenCV then dot operator used, then by using default function available in OpenCV, the createTrackbar is used in that method the name of the trackbar is given called lower hue as "LH" for lower boundation. Next the window name is given in which this trackbar is displayed, called "Tracking". After that the value of the trackbar is given called position is equal to "0" for lower boundation of hue, saturation and value. After that count is given called the maximal position is equal to "255" for lower boundation of hue, saturation and value. Note that maximum value is only 255 because the possible colors is 256 ,but in arrays it is consider as 0 to 255. The next one is define the callback function called **"nothing"** with a special a statement called **"pass"** for position onChange ,means no callbacks are called, but only value is updated. That value is collected from trackbar whenever the changes is made in it. Now repeat the steps for each trackbar namely called lower saturation as "LS", lower value as "LV". Now, create a trackbar for upper boundation upper hue as "UH", upper saturation as "US", and upper value as "UV". Now, create the trackbar for upper boundation using cv2 in the OpenCV then dot operator used, then by using default function available in OpenCV, the createTrackbar is used in that method the name of the trackbar is given called upper hue as "UH". Next the window name is given in which this trackbar is displayed, called "Tracking". After that the value of the trackbar is given called position is equal to "255" for upper boundation of hue, saturation and value. After that count is given called the maximal position is equal to "255" for lower boundation of hue, saturation and value. The next one is define the callback function called **"nothing"** with a special a statement called **"pass"** for position onChange ,means no callbacks are called, but only value is updated. That value is collected from trackbar whenever the changes is made in it.

CREATE AN INFINITE WHILE LOOP TO GET THE TRACKBAR POSITION:

After, creating an trackbar for each parameters in HSV color space namely lower boundation such as lower hue, lower saturation and lower value and then for upper boundation such as upper hue, upper saturation and upper value. Now, create a infinite while loop to get the trackbar position. To make the trackbar and all windows open and should not close unnecessarily ,while getting that trackbar position. Then by using an special default function available in OpenCV called **"getTrackbarPos()"**. Whenever there is a change in trackbar position, then by using this method get trackbar position separately for each lower boundation such as lh,ls,lv and upper boundation such as uh,us,uv. To store this values getted , create a separate new variable for each parameters namely l_h for lower hue, l_s for lower saturation and l_v for lower value in lower boundation. u_h for upper hue, u_s for upper saturation and u_v for upper value in upper boundation.

METHOD:

While True:

COMMAND:

while Expression:
 statement(s)

- while is the keyword to create an Entry controlled loop.
- Expression is the condition to be given in the statement.
- while loop is used to repeat a section of code unknown number of times until a specific condition is met.
- While loop falls under the category of indefinite iteration.
- Indefinite iteration means that the number of times the loop is executed isn't specified explicitly in advance.

METHOD:

```
l_h=cv2.getTrackbarPos("LH","Tracking")
```

```
l_s=cv2.getTrackbarPos("LS","Tracking")
```

```
l_v=cv2.getTrackbarPos("LV","Tracking")
```

```
u_h=cv2.getTrackbarPos("UH","Tracking")
```

```
u_s=cv2.getTrackbarPos("US","Tracking")
```

```
u_v=cv2.getTrackbarPos("UV","Tracking")
```

COMMAND:

```
Var= cv2.getTrackbarPos("Trackbar name","Trackbar window")
```

- Var is called as variable it is a user defined.
- OpenCV is renamed as cv2.
- getTrackbarPos () is the special default function in OpenCV to get the trackbar position.
- Trackbar name is the name of the trackbar, in which trackbar want to get the trackbar position.
- Trackbar window is the name of the trackbar window,where trackbar is located.

EXPLANATION:

First step is to create a infinite while loop to make all windows open and should not close unnecessarily while getting that trackbar position. Second step is to use the special default function available in OpenCV called "getTrackbarPos()". Next use cv2 with dot operator then use this getTrackbarPos() and define the trackbar window name as "LH" and the trackbar name as "Tracking", that we want to get the values from it. Then create a variable called "l_h" for

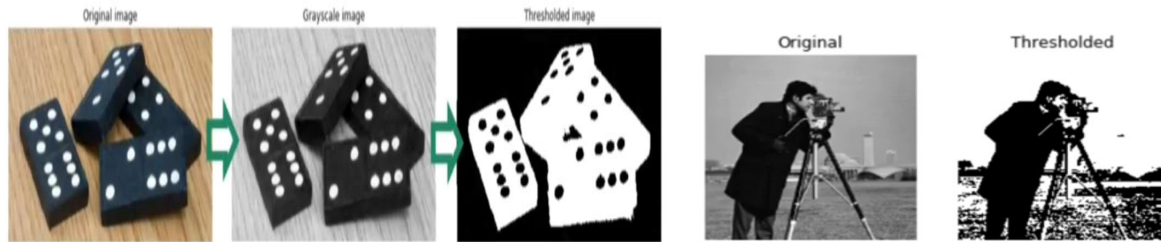
lower hue to store the values retrieved from the trackbar. Then Repeat steps for all the trackbars namely l_s, l_v and u_h, u_s, u_v. Now create a variable called "l_s" for lower saturation then use cv2 with dot and getTrackbarPos (trackbar name as "LS" and trackbar window name as "Tracking"). Next, create a variable called "l_v" for lower value then use cv2 with dot and getTrackbarPos (trackbar name as "LV" and, trackbar window name as "Tracking"). Now repeat same step for upper boundation parameters such as u_h, u_s, u_v. Now, create a variable called "u_h" for upper hue then use cv2 with dot and getTrackbarPos (trackbar name as "UH" and, trackbar window name as "Tracking"). Next, create a variable called "u_s" for upper saturation then use cv2 with dot and getTrackbarPos (trackbar name as "US" and, trackbar window name as "Tracking"). Now, create a variable called "u_v" for lower value then use cv2 with dot and getTrackbarPos (trackbar name as "UV" and, trackbar window name as "Tracking").

3. THRESHOLDING:

Thresholding means partitioning an image into foreground and background. By making it black and white. Thresholding is the binarization of images. Extracting of foreground and objects. Extracting of background objects. we do so by setting each pixel to:

- 255 (white) if pixel > thresh value
- 0 (black) if pixel < thresh value.





4. NUMPY:

Numpy stands for numerical python library. 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. Numpy is a highly optimized library for numerical operations. Array structure is important because digital images are two dimensional arrays of pixels. All the OpenCV array structures are converted to and from numpy array. It is a more convenient indexing system rather than using for loops.

ADVANTAGES:

- Use Less memory and Faster execution
- Convenient
- SIMD vector processing
- No type checking, when iterating through objects.
- Effective cache utilization
- Save coding time

APPLICATIONS OF NUMPY:

- Mathematics (MATLAB Replacement)
- Plotting (matplotlib)
- Backend (pandas, connect , digital photography)
- Machine learning.

ARRAY FUNCTIONS:

- | | | |
|-------------|--------------|----------|
| 1.sum() | 4. mean(). | 7.std() |
| 2.median() | 5.median () | |
| 3.max() | 6. var() | |

ARRAY OPERATIONS:

- | | | |
|-------------------------------------|------------------|--------------|
| 1. Accessing and Slicing Operations | 6. concatenate() | 11. split() |
| 2. Copy() | 7. view sort() | 12. dstack() |
| 3. reshape() | 8. vstack() | 13. where() |
| 4. append() | 9. hstack() | 14. stack() |
| 5. searchsorted() | 10. insert() | 15. delete() |

D. NUMPY IN OBJECT DETECTION:

To process the HSV image for object detection, we first convert it into arrays using numpy package in Python. Import numpy and rename it as np. Then using array() function convert the image into array. Here, we going apply array function to lower boundation, parameters namely lower hue as l_h, lower saturation as l_s, lower value as l_v and upper boundation parameters namely upper hue as u_h, upper saturation as u_s, upper value as u_v. Note that is above parameters l_h, l_s, l_v and u_h, u_s, u_v is the variable which contains the trackbar position, which is brought from the trackbar by getTrackbarPos() method in OpenCV. To store this thresholding values by numpy create a variable called lower boundation as l_b it has parameters as l_h, l_s, l_v. Next create a variable called upper boundation as u_b to store the thresholding values of parameters such as u_h, u_s, u_v.

METHOD:

```
l_b = np.array([l_h, l_s, l_v])
```

```
u_b = np.array([u_h, u_s, u_v])
```

- Numpy is renamed as np
- l_h, l_s, l_v is the range of lower boundation to be thresholded.
- u_h, u_s, u_v is range of upper boundation to be thresholded.
- array() is the function to create a one dimensional, two dimensional, multidimensional arrays, etc.

COMMAND:

- `Var= np.array([dimensions parameters])`
- Var is the variable ,it store the threshold value.
- Numpy is renamed as np
- Array is the default function available in numpy

ARRAY (...):

`array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)`

ARRAY PARAMETERS:

Object dtype (data-type is optional) copy(bool, optional) order['K','A','C','F:'] optional.

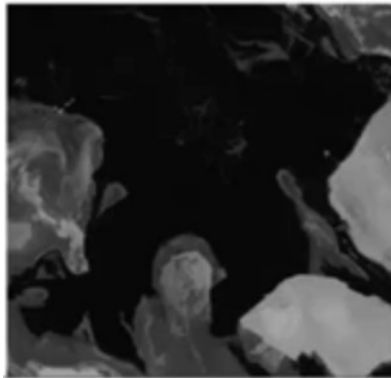
EXPLANATION:

First create a new variable to store a threshold value of lower boundation.as `l_b`, these values are retrieved from the trackbar created using the `getTrackbarPos()` available in OpenCV. Next using the numpy package convert the HSV image into arrays for that call numpy as np (means rename it as np), then use dot operator with a special default function available in numpy called "`array()`". Then pass this retrieved trackbar values to the arrays function as `l_b = np.array([l_h,l_s,l_v])`. Now, repeat the same process for upper boundation as `u_b`,these values are retrieved from trackbars of upper hue as `u_h`, upper saturation as `u_s` and upper value as `u_v`. Now using the numpy package convert the HSV image into arrays for that call numpy as np (means rename it as np), then use dot operator with a special default function available in numpy called "`array()`". Then pass this retrieved trackbar values to the arrays function as `u_b = np.array([u_h,u_s,u_v])`. Remember that, now using numpy array function we created an one dimensional array in lower boundation as well as for upper boundation.

5. MASKING:

Masking is used to recover a portion of a image of the specified colour. A masking pixels value of 0 and 1. Advantages of applying a mask are Better contrast stretch, Remove meaningless pixels and Crisper display, etc.

**Goal is to visualize/analyze
Only the water in this image**

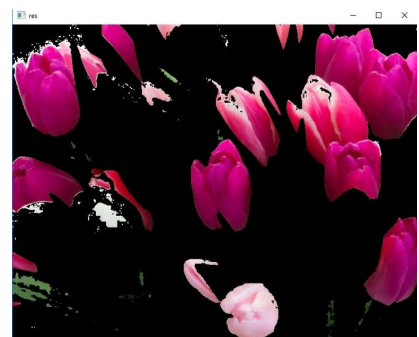
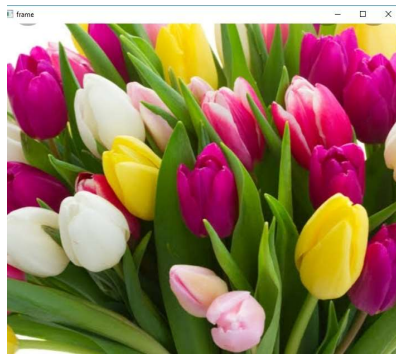
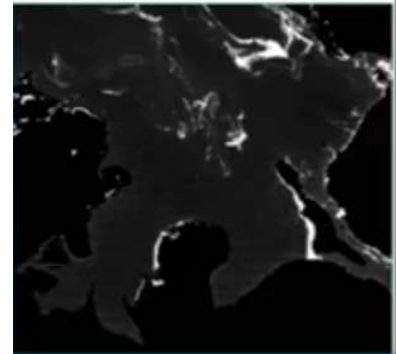


**Create a masking raster
Active pixels =1
Masked pixels =0**



**Multiply the mask raster by
the image:**

**Active pixels = same value
Masked pixels = 0**



APPLICATIONS OF MASKING:

- Medical imaging
- Surveillance Robotics
- Automotive safety
- Consumer electronics
- Geospatial computing
- Machine vision, etc.



E. MASKING IN OBJECT DETECTION:

The next step is to apply masking to the threshold value of the lower bound and upper bound such as `l_b` and `u_b`. To apply masking the special default function available in OpenCV is used called "`inRange()`". This function allows to the values only belongs to that thresholded HSV image's Lower and upper bound range . Next create a variable called "mask" to store this masked image.

METHOD:

`Mask= cv2.inRange(hsv,l_b,u_b)`

- Mask - Destination image (variable)
- `cv2.inRange()` - Function
- Src -source image (hsv)
- Lower -The intended lower threshold (`l_b`)
- Upper -The intended upper threshold (`u_b`)

COMMAND FOR PYTHON:

- cv2.inRange(src,lower, upper,dst)
- src - First input array
- lowerb - Inclusive lower boundary array or a scalar.
- upperb - Inclusive upper boundary array or a scalar.
- dst - output array of the same size as src and CV_8U type.

The function checks the range as follows:

FOR ONE DIMENSIONAL ARRAY:

$dst(I)_0 = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0$

FOR TWO DIMENSIONAL ARRAYS:

$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0 \wedge lowerb(I)_1 \leq src(I)_1 \leq upperb(I)_1$.

That, is dst(I) is set to 255 (all 1 bits) if src(I) is within the specified 1D,2D,3D,...box and 0 otherwise. When the lower and/or upper boundary parameters are scalars, the indexes(I) at lowerb and upperb in the above formula should be omitted.

EXPLANATION:

The next step is to mask an threshold parameters namely lower boundation variable such as lower hue lower saturation, lower value and upper boundation variable such as upper hue, upper saturation, upper value. First create a variable called "mask" then using numpy package function and rename it as np with dot operator, then use the special default function available in numpy called "inRange()". This function is used to mask an image within a range. Then define the range of the mask, the first parameter is source. Here, source is hsv image (which is the variable contain the HSV image used earlier `hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)`). The second parameter is the lower boundation such as `l_h,l_s,l_v` (which is specified earlier as `l_h=cv2.getTrackbarPos("LH","Tracking")` and so on). The third parameter is the upper boundation such as `u_h,u_s,u_v` (which is specified earlier as `u_h = cv2.getTrackbarPos("UH","Tracking")` and so on). Next, the function checks for the range as for one dimensional array it use this formula " $dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0$ ". To process this formula HSV images and parameters used are converted into array. Then the masked image is stored in the mask variable.

6. BITWISE AND OPERATION:

Bitwise operators are used to perform operations at binary digit level. These operators are commonly used only in special applications, where optimized use of storage is required. There are types of bitwise operators listed below:

- & - Bitwise AND
- | - Bitwise OR
- ^ - Bitwise exclusive OR / Bitwise XOR
- ~ - Bitwise inversion (one's complement)
- << - Shifts the bits to left /. Bitwise Left Shift
- >> - Shifts the bits to right / Bitwise Right Shift

BITWISE AND (&):

OPERAND 1	OPERAND 2	RESULT(OP1 & OP 2)
True 1	True 1	True 1
True 1	False 0	False 0
False 0	True 1	False 0
False 0	False 0	False 0

F. BITWISE AND OPERATION IN OBJECT DETECTION:

The next step is to perform bitwise_and operation. The OpenCV have a special default function available called "bitwise_and()". To store this result of masking create a variable called "res". Then use cv2 with dot operator, next use the bitwise_and operation. The first parameter is the first "frame" ,the second parameter is "frame" ,the third parameter is "mask".

METHOD:

```
res=cv2.bitwise_and(frame,frame,mask=mask)
```

COMMAND:

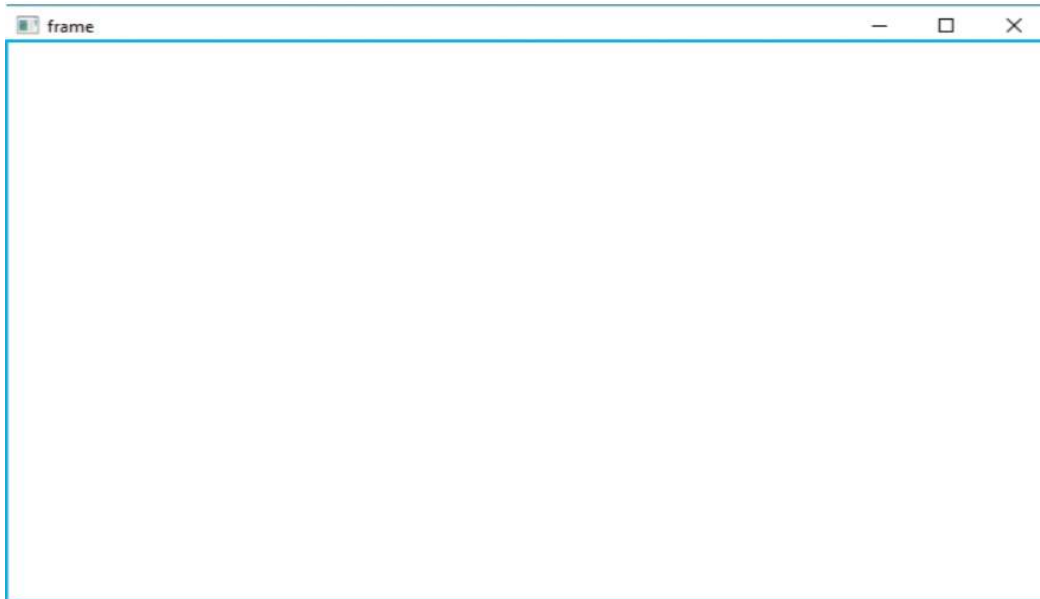
`res = cv2.bitwise_and(src1,src2,mask)`

- `res` - Resultant image(Destination image).
- `src1` -The first source image.
- `src2` - The second source image.
- `mask` - The binary mask image obtained.
- `bitwise_and` - It combine both the source 1 and source 2 and perform `bitwise_and` operation. By using bitwise and operator to ,get the resultant (final image)mask.

EXPERIMENTAL ANALYSIS

C. EXPERIMENTAL ANALYSIS:

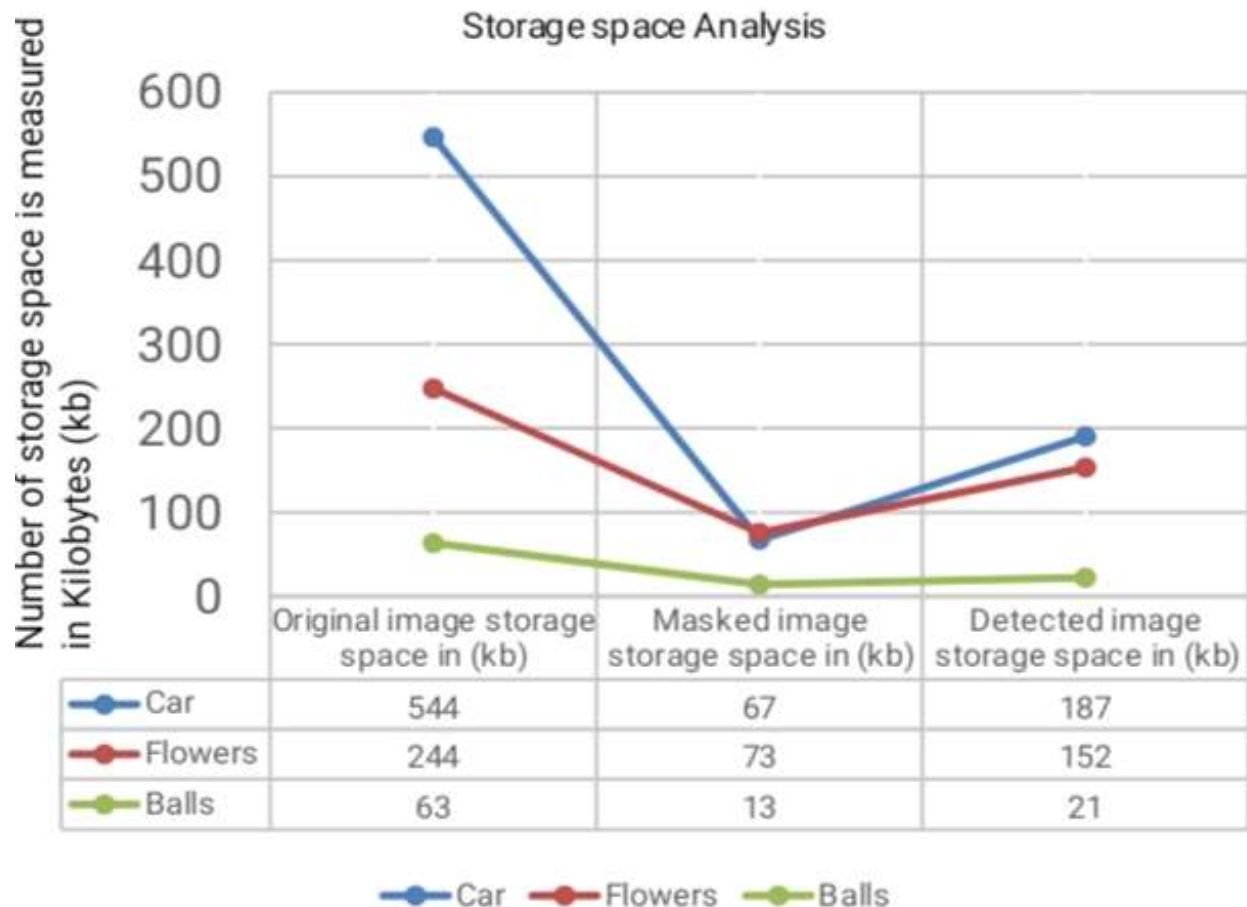
C.1 FORM INPUT DESIGN:





C.2 GRAPH STRUCTURE:

C.2.1 STORAGE SPACE ANALYSIS:



EXPLANATION:

This experimental analysis belongs to the storage space Analysis of the object detection using HSV color space with machine learning libraries in python. This graph gives the details about, different image storage space and it varies from each other. The x-axis is the name of the image like original image, masked image, and detected image storage space. The y-axis is the number of storage space and it is measured in Kilobytes (kb). The names mentioned in the left bottom corner are the names of the images used and they have a ".png" extension (Car.png, Flowers.png, Balls.png). Now, take a look at the graph; there are three lines. The first line is blue color (Car.png). The second line is red color (Flowers.png). The third line is green color (Balls.png). Now,

compare the car.png image storage space. First, take the original image of Car.png has a storage space is 544 kb, Second the masked image it is a monochrome image (Black and white) its bit depth=1. So the masked image of Car.png has the storage space is 67 kb. Third, detected image of Car.png it has a storage space of 187 kb. Now, take the original image of Flowers.png it has the storage space of 244 kb. Second, the masked image of Flowers.png has the storage space of 73 kb. Third, the detected image of Flowers.png has the storage space of 152 kb. Now consider the original image of Balls.png it has the storage space of 63 kb. Second, the masked image of Balls.png it has a storage space is 13 kb. Third, the detected image of Balls.png it has a storage space of 21 kb. So, by comparing all these three images namely Car.png, Flowers.png, Balls.png we conclude that Original image storage space is greater than compared to masked image and detected image storage space. We can also conclude that masked image storage space is less compared to detected image and original image storage space. So, here the formula is, Detected image storage space is less compared to Original image storage space and greater than Masked image storage space. The object detection using HSV color space with machine learning libraries in python, should satisfy the below conditions.

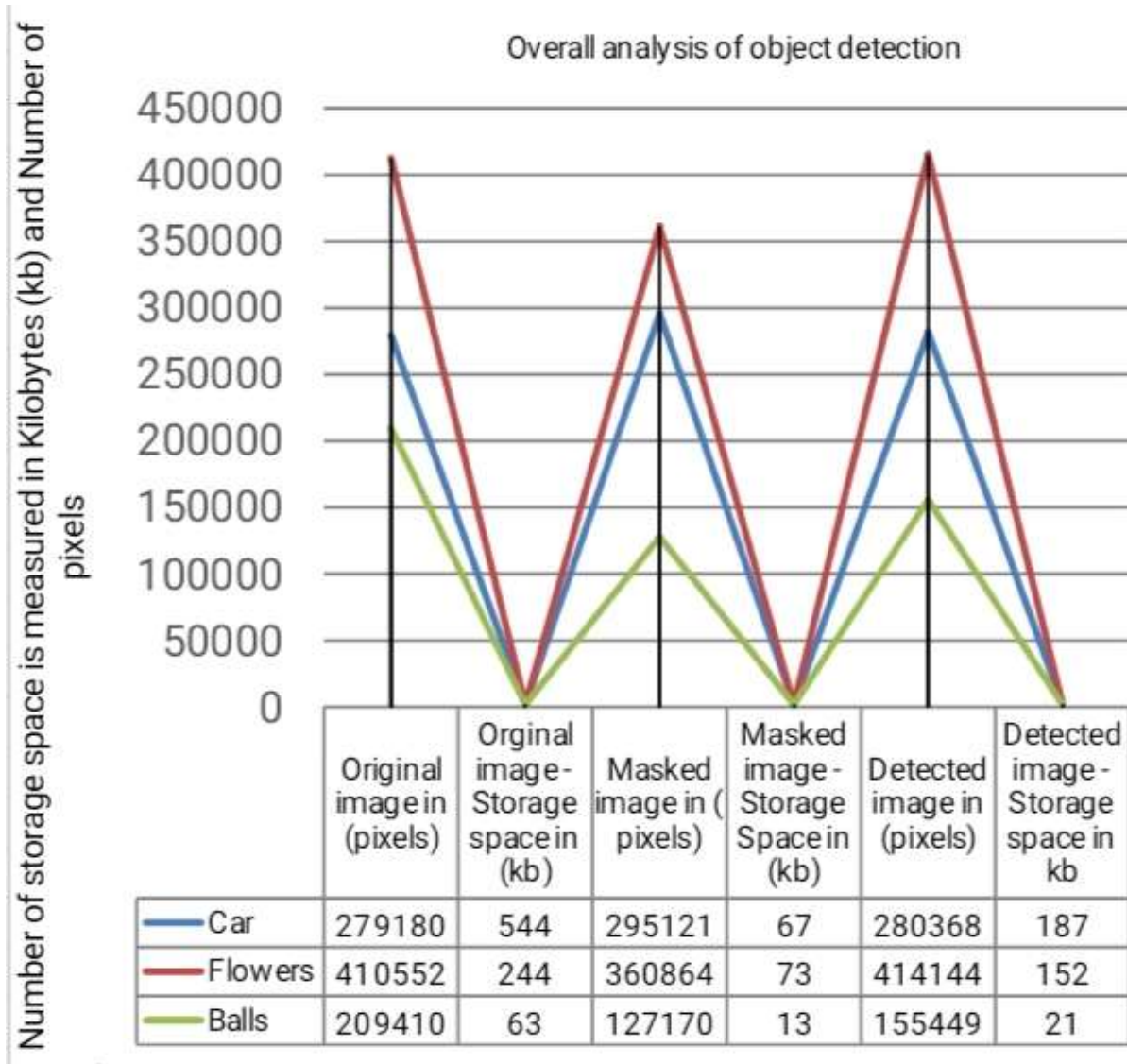
- $D < O \ \& \ D > M$ (i.e., Detected image storage space is less than Original image storage space and Detected image storage space is greater than Masked image storage space).
- $O \neq D \ \& \ M$ (i.e., Original image storage space is not equal to Detected image storage space and Masked image storage space).
- $D \neq O \ \& \ M$ (i.e., Detected image storage space is not equal to Original image storage space and Masked image storage space).
- $M \neq O \ \& \ D$ (i.e., Masked image storage space is not equal to Original image storage space and Detected image storage space).

NOTE:

- $O \Rightarrow$ Original image storage space
- $D \Rightarrow$ Detected image storage space
- $M \Rightarrow$ Masked image storage space

Above, mentioned condition is for image's storage space occupied.

C.2.2 OVERALL ANALYSIS OF OBJECT DETECTION:



EXPLANATION:

This experimental analysis is belongs to the Overall analysis of the object detection using HSV color space with machine learning libraries in python. This graph gives the details about, different image storage space and it's Pixels(dimensions) and it is from vary from each other. The x-axis is name of the image like original image, masked image, and detected image

storage space and the number of pixels of these three images. The y-axis is the number of storage space and it is measured in Kilobytes (kb) and the number of pixels of the image. The name mention in the left bottom corner is the name of the images used and it is ".png" extension (Car.png, Flowers.png, Balls.png). The values in left or y-axis 0 - 4,50,000 carries the number of pixels (i.e., 750x240) and number of storage space is measured in Kilobytes (kb). Now, we can view the graph lines are appear as a up and down (heart pluse). The first line is Red colour up and down is (Flowers.png). The second line is the blue colour up and down is(Car.png). The third line is green colour up and down is(Balls.png).

Lets, start comparing first take the original image of (Flowers.png) it has the pixels resolution of (703x584), the summation of height and width is equal to 410552. So, it has the storage space of 244 Kilobytes (kb). Now, take the masked image of (Flowers.png) it has the pixels resolution of (716x504), summation is equal to 360864. So, it has the storage space of 73 Kilobytes (kb). Now, consider the detected image of (Flowers.png) it has the pixels resolution of (719x576) summation is equal to 41444. So, it has the storage space of 152 Kilobytes (kb). Now, take the original image of (Cars.png) it has the pixels resolution of (705x396), the summation of height and width is equal to 279180. So, it has the storage space of 544 Kilobytes (kb). Now, take the masked image of (Cars.png) it has the pixels resolution of (701x421), summation is equal to 295121. So, it has the storage space of 67 Kilobytes (kb). Now, consider the detected image of (Flowers.png) it has the pixels resolution of (708x396) summation is equal to 280368. So, it has the storage space of 187 Kilobytes (kb). Now, take the original image of (Balls.png) it has the pixels resolution of (487x430), the summation of height and width is equal to 209410. So, it has the storage space of 63 Kilobytes (kb). Now, take the masked image of (Balls.png) it has the pixels resolution of (405x314), summation is equal to 127170. So, it has the storage space of 13 Kilobytes (kb). Now, consider the detected image of (Balls.png) it has the pixels resolution of (419x371) summation is equal to 155449. So, it has the storage space of 21 Kilobytes (kb).

So, we can conclude that the Final output (Detected image) it's storage space is less than Original image storage space and greater than Masked image storage space. But it's Dimensions (pixels) value is greater than Original image pixels and Masked image pixels. Because, of the final outcome (detected image) has good clarity, that's why it's pixel value (414144) greater than the original image pixels values(410552) and the masked image pixels values (360864).

The object detection using HSV color space with machine learning libraries in python, should satisfy the below conditions

- $D < O \ \& \ D < M$ (i.e., Detected image's pixels value is greater than Original image's pixels values space and Masked image's pixels value).
- $O \neq D \ \& \ M$ (i.e., Original image's pixels value should be not equal to Detected image's pixels value and Masked image's pixels value.)
- $D \neq O \ \& \ M$ (i.e., Detected image's pixels value should be not equal to Original image's pixels value and Masked image's pixels value.)

- $M \neq O \ \& \ D$ (i.e., Masked image's pixels value should be not equal to Original image's pixels value and Detected image's pixels value.)

Above, mentioned condition is for image's pixels value (Dimensions).

- $D < O \ \& \ D > M$ (i.e., Detected image storage space is less than Original image storage space and Detected image storage space is greater than Masked image storage space).
- $O \neq D \ \& \ M$ (i.e., Original image storage space is not equal to Detected image storage space and Masked image storage space).
- $D \neq O \ \& \ M$ (i.e., Detected image storage space is not equal to Original image storage space and Masked image storage space).
- $M \neq O \ \& \ D$ (i.e., Masked image storage space is not equal to Original image storage space and Detected image storage space).

Above, mentioned condition is for image's storage space occupied.

NOTE:

- $O \Rightarrow$ Original image
- $D \Rightarrow$ Detected image
- $M \Rightarrow$ Masked image

C.3 TABLE STRUCTURE:

C.3.1 IMAGE CHARACTERISTICS TABLE:

This table contains the details about image name, storage space and pixels value for three categories namely Original image, Masked image and Detected image.

S.no	Original image name	OImage Storage space	OImage Dimensions (pixels)	Extension	MImage Dimensions (pixels)	MImage Storage space	DImage Dimensions (pixels).	DImage Storage space
1.	Car	544 kb	705x396	.png	701x421	67 kb	708x396	187 kb
2.	Flowers	244 kb	703x584	. png	716x504	73 kb	719x576	152 kb
3.	Balls	63 kb	487x430	. png	405x430	13 kb	419x371	21 kb

C.3.2 HSV COLOR SPACE PARAMETERS TABLE:

This table contains the details about the HSV COLOR SPACE parameters namely lower boundation (lower hue, lower saturation and lower value) and upper boundation (upper hue, upper saturation and upper value).

S.no	Image name	Lower Hue	Lower Saturation	Lower Value	Upper Hue	Upper Saturation	Upper Value
1	Car	40	2	0	255	236	255
2	Flowers	12	102	47	246	255	243
3	Balls	61	238	178	121	255	255









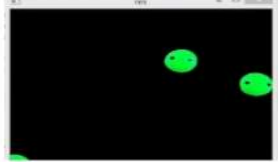
C.3.3 IMAGE APPERANCE CHARACTERISTICS TABLE:

This table contains the details about the image's apperance characteristics namely count of colours and background colours.

S.no	Img name	Cou nt	Red	Gree n	Blue	yell ow	whit e	blac k	light gree n	viol et	bab y Pink	bro wn	oran ge	Pink
1	car	1	0	0	Bac kgro und	1	6	4	7	0	0	0	0	0
2	flow er	40	1	infin ite	0	4	3	0	0	8	4	1	0	3
3	ball s	13	4	3	3	o	Bac kgro und	0	0	0	0	1	2	0










C.3.4 IMAGES VIEW:

This table contains the about the view of image in three categories namely Original image, Masked image and Detected image

S.no	Original Image	Masked Image	Detected Image
1			
2			
3			

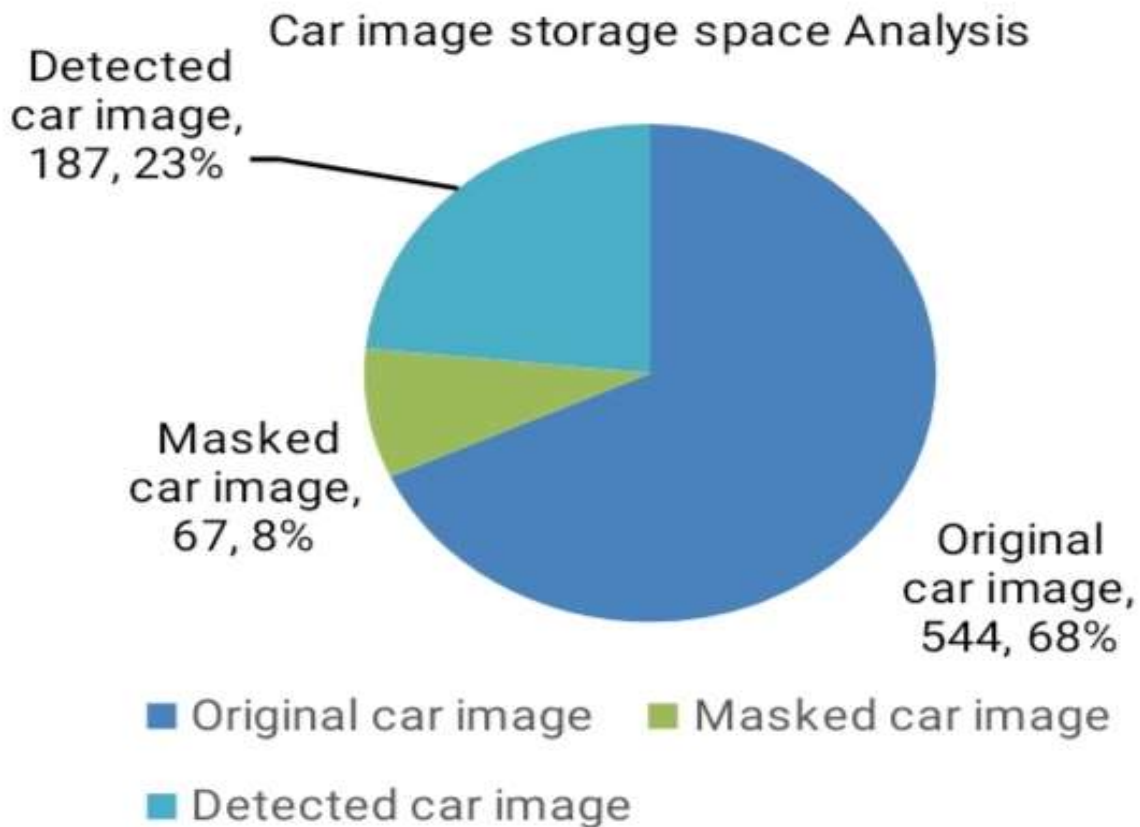
C.3.5 OVERALL CHARACTERISTICS OF IMAGE:

This table contains the details about the overall characteristics of image namely image's storage space and it is measured in Kilobytes (kb). The colors involved for original image, masked image and detected image. Next it contains the HSV color space parameters namely lower boundation (Lower hue, Lower Saturation and Lower Value) and upper boundation (Upper Hue, Upper Saturation and Upper Value).

S.no	Image name	Colors Involved	Lb(lh,ls,lv) ub(uh,us,uv)	Masked image color	Detected image color
1 	car.png O - 544 KB M - 67 KB D - 187 KB	Yellow, white, light green, blue, grey, black	lb(40,2,0) ub(255,236,255)	Black and white 	Blue, light green, white, grey 
2 	flowers.png O - 244 KB M - 122 KB D - 272 KB	white, yellow, light pink, green, violet, red, baby pink	lb(12,102,47) ub(246,255,243)	Black and white 	Violet, yellow, green, pink 
3 	balls.png O - 63 KB M - 6 KB D - 17 KB	red, brown, green, blue, orange	lb(61,238,178) ub(121,255,255)	Black and white 	Green 

C.4 CHART STRUCTURE:

C.4.1 CAR IMAGE STORAGE SPACE ANALYSIS:



EXPLANATION:

This experimental analysis belongs to the Overall analysis of the object detection using HSV color space with machine learning libraries in python. This pie chart gives the details about, Car image storage space and it is measured in Kilobytes (kb). Where the Car image is in .png extension. Take the pie chart in clockwise direction. So, starts from right it is a Original Car image storage space (544 Kilobytes (kb)) it covers 68%. Now move to the left it is a Masked Car image storage space it has (67 Kilobytes (kb)) it covers 8%. Then move to the left top it is a Detected Car image storage space it has (187 Kilobytes (kb)) it covers 23%.

Let's see the reason why there is a variation in the storage space. First of all consider the colored Original Car image storage space has high clarity with Car.png extension. Because, Object

detection gives accurate result only for an image with high quality (.png extension). Second consider the Masked Car image storage space of the same Car.png image. It has less storage space compared to original image storage. Because, while Masking an image it has only two colors namely (black and white) or Monochrome image, that's why it has low storage space. Finally take the Detected Car image storage space of the same Car.png image. It has the storage space greater than the Masked Car image and lesser than the Original Car image storage space. Because, the detected image has only the object that have been detected by the tracker and it displayed as a colored detected image. So, it has greater storage space than Masked Car image and less storage space than Original Car image storage space. So, we can conclude that there is a variation in the storage space. The object detection using HSV Color space must satisfy the following conditions,

- $D < O \ \& \ D > M$ (i.e., Detected Car image storage space is less than Original Car image storage space and Detected Car image storage space is greater than Masked Car image storage space).
- $O \neq D \ \& \ M$ (i.e., Original Car image storage space is not equal to Detected Car image storage space and Masked Car image storage space).
- $D \neq O \ \& \ M$ (i.e., Detected Car image storage space is not equal to Original Car image storage space and Masked Car image storage space).
- $M \neq O \ \& \ D$ (i.e., Masked Car image storage space is not equal to Original Car image storage space and Detected Car image storage space).

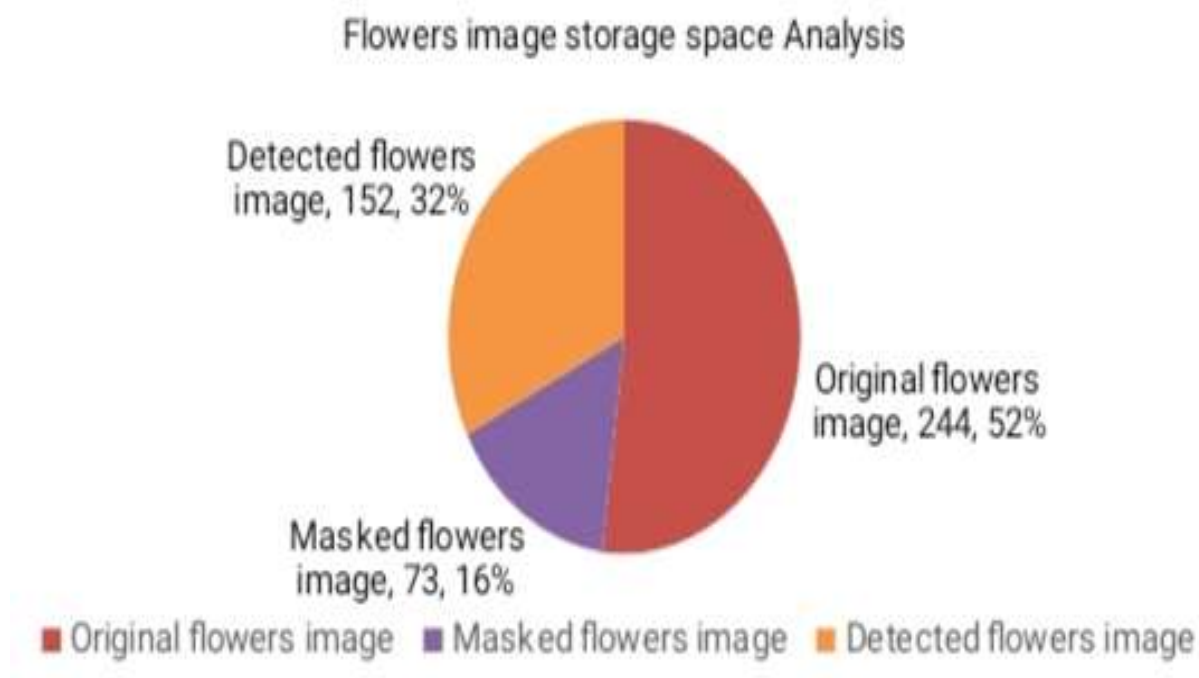
NOTE:

- $O \Rightarrow$ Original Car image storage space
- $D \Rightarrow$ Detected Car image storage space
- $M \Rightarrow$ Masked Car image storage space

Above, mentioned condition is for image's storage space occupied.

.

C.4.2. FLOWERS IMAGE STORAGE SPACE ANALYSIS:



EXPLANATION:

This experimental analysis belongs to the Overall analysis of the object detection using HSV color space with machine learning libraries in python. This pie chart gives the details about, Flowers image storage space and it is measured in Kilobytes (kb). Where the Flowers image is in .png extension. Take the pie chart in clockwise direction. So, starts from right it is a Original Flowers image storage space (244 Kilobytes (kb)) it covers 52%. Now move to the left it is a Masked Flowers image storage space it has (73 Kilobytes (kb)) it covers 16%. Then move to the left top it is a Detected Flowers image storage space it has (152 Kilobytes (kb)) it covers 32%.

Let's see the reason why there is a variation in the storage space. First of all consider the colored Original Flowers image storage space has high clarity with Flowers.png extension. Because, Object detection gives accurate result only for an image with high quality (.png extension). Second consider the Masked Flowers image storage space of the same Flowers.png image. It has less storage space compared to original Flowers image storage. Because, while Masking an image it has only two colors namely (black and white) or Monochrome image, that's why it has low storage space. Finally take the Detected Flowers image storage space of the same Flowers.png image. It has the storage space greater than the Masked image and lesser than the

Original image storage space. Because, the detected Flowers image has only the object that have been detected by the trackbar and it displayed as a colored detected image. So, it has greater storage space than Masked Flowers image and less storage space than Original Flowers image storage space. So, we can conclude that there is a variation in the storage space. The object detection using HSV Color space must satisfy the following conditions,

$D < O \ \& \ D > M$ (i.e., Detected Flowers image storage space is less than Original Flowers image storage space and Detected Flowers image storage space is greater than Masked Flowers image storage space).

$O \neq D \ \& \ M$ (i.e., Original Flowers image storage space is not equal to Detected Flowers image storage space and Masked Flowers image storage space).

$D \neq O \ \& \ M$ (i.e., Detected Flowers image storage space is not equal to Original Flowers image storage space and Masked Flowers image storage space).

$M \neq O \ \& \ D$ (i.e., Masked Flowers image storage space is not equal to Original Flowers image storage space and Detected Flowers image storage space).

NOTE:

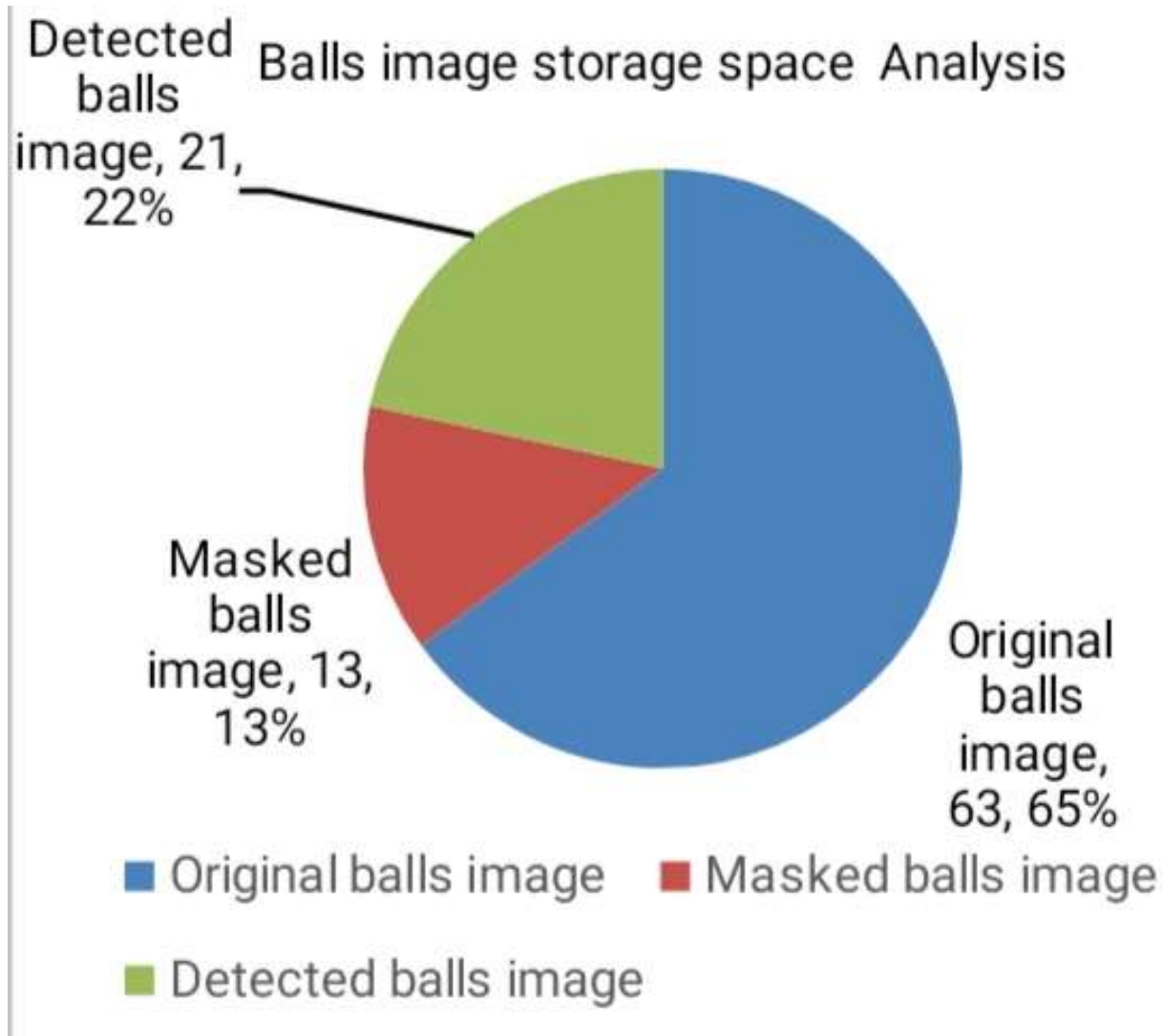
$O \Rightarrow$ Original Flowers image storage space

$D \Rightarrow$ Detected Flowers image storage space

$M \Rightarrow$ Masked Flowers image storage space

Above, mentioned condition is for image's storage space occupied.

C.4.3. BALLS IMAGE STORAGE SPACE ANALYSIS:



EXPLANATION:

This experimental analysis is belongs to the Overall analysis of the object detection using HSV color space with machine learning libraries in python. This pie chart gives the details about, Balls image storage space and it is measured in Kilobytes (kb). Where the Balls image is in . png extension. Take the pie chart in clockwise direction. So, starts from right it is a Original Balls

image storage space (63 Kilobytes (kb)) it covers 65%. Now move to the left it is a Masked Balls image storage space it has (13 Kilobytes (kb)) it covers 13%. Then move to the left top it is a Detected Balls image storage space it has (21 Kilobytes (kb)) it covers 22%.

Let's see the reason why there is a variation in the storage space. First of all consider the colored Original Balls image storage space has high clarity with Balls.png extension. Because, Object detection gives accurate result only for an image with high quality (.png extension). Second consider the Masked Balls image storage space of the same Balls.png image. It has less storage space compared to original Balls image storage. Because, while Masking an image it has only two colors namely (black and white) or Monochrome image, that's why it has low storage space. Finally take the Detected Balls image storage space of the same Balls.png image. It has the storage space greater than the Masked Balls image and lesser than the Original Balls image storage space. Because, the detected Balls image has only the object that have been detected by the trackbar and it displayed as a colored detected image. So, it has greater storage space than Masked Balls image and less storage space than Original Balls image storage space. So, we can conclude that there is a variation in the storage space. The object detection using HSV Color space must satisfy the following conditions,

$D < O \ \& \ D > M$ (i.e., Detected Balls image storage space is less than Original Balls image storage space and Detected Balls image storage space is greater than Masked Balls image storage space).

$O \neq D \ \& \ M$ (i.e., Original Balls image storage space is not equal to Detected Balls image storage space and Masked Balls image storage space).

$D \neq O \ \& \ M$ (i.e., Detected Balls image storage space is not equal to Original Balls image storage space and Masked Balls image storage space).

$M \neq O \ \& \ D$ (i.e., Masked Balls image storage space is not equal to Original Balls image storage space and Detected Balls image storage space).

NOTE:

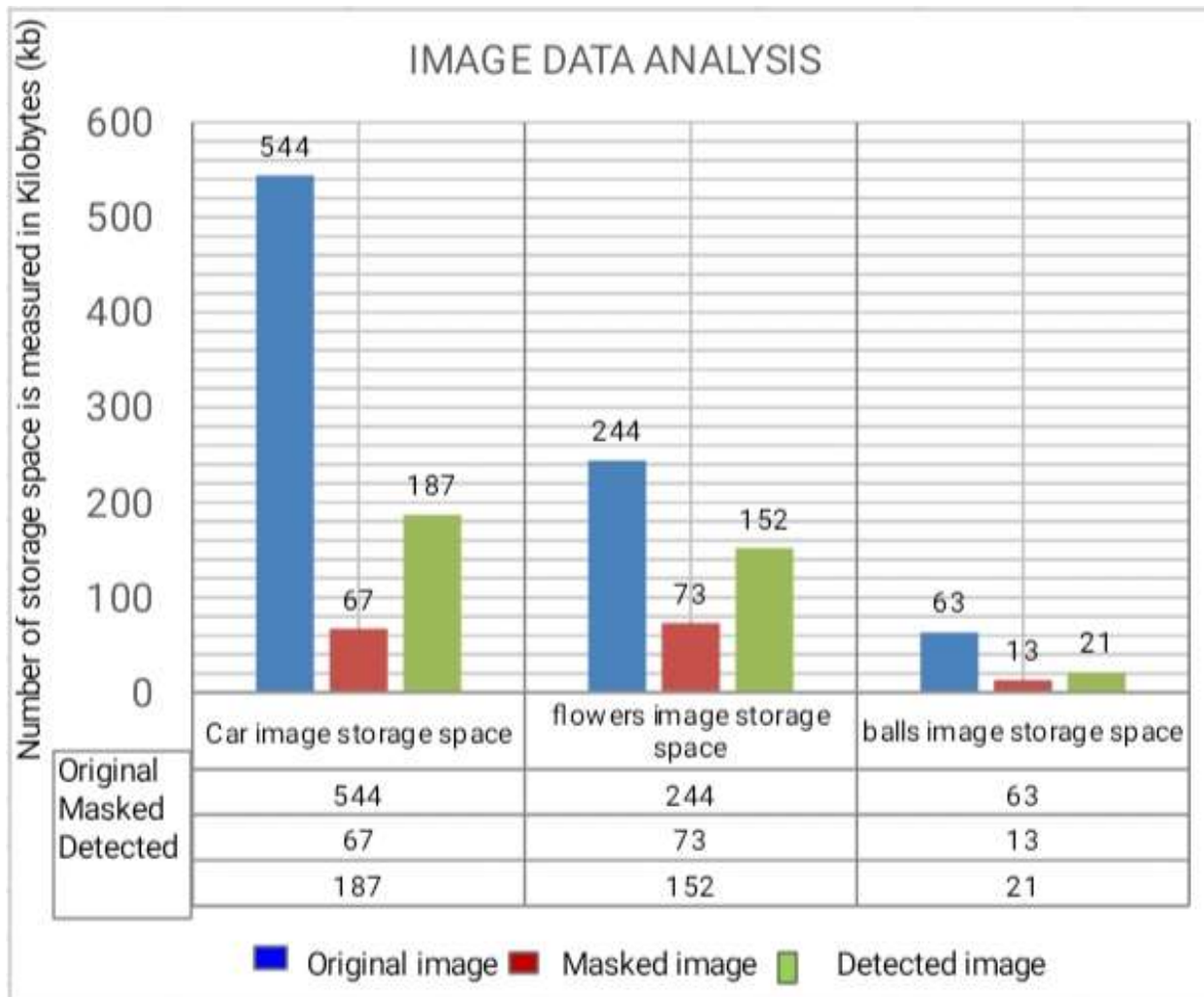
$O \Rightarrow$ Original Balls image storage space

$D \Rightarrow$ Detected Balls image storage space

$M \Rightarrow$ Masked Balls image storage space

Above, mentioned condition is for image's storage space occupied.

C.4.4. IMAGE DATA ANALYSIS:



EXPLANATION:

This experimental analysis is belongs to the Overall analysis of the object detection using HSV color space with machine learning libraries in python. This bar chart gives the details about, the different image storage space and it is measured in Kilobytes (kb) and it is vary from each other. The x-axis is name of the image used namely Original Car image, masked Car image, and detected Car image, Original Flowers image, masked Flowers image, and detected Flowers image, Original Balls image, masked Balls image, and detected Balls image storage space. The y-axis is the number of storage space and it is measured in Kilobytes (kb). The name mention in the left bottom corner is the name of the images used Original image, Masked image , Detected image and it has ".png" extension (Car.png, Flowers.png, Balls.png). Now, take a look at chart

there are three sections. First section belongs to the Car.png image's storage space. It contains three bar, first bar in the first section is Original Car image storage space of 544 Kilobytes (kb). Second bar in the first section is Masked Car image storage space of 67 Kilobytes(kb). Third bar in the first section is Detected Car image storage space of 187 Kilobytes (kb). So,we can conclude that Detected Car image storage space is less than Original Car image storage space and greater than Masked Car image storage space. Because, the detected image has only the object that have been detected by the trackbar and it displayed as a colored detected image. So, it has greater storage space than Masked Car image(Monochrome image or black and white image) and less storage space than Original Car image storage space.

The second section belongs to the Flowers.png image's storage space. It contains three bar, first bar in the first section is Flowers image storage space of 244 Kilobytes (kb). Second bar in the first section is Masked Flowers image storage space of 73 Kilobytes(kb). Third bar in the first section is Detected Flowers image storage space of 152 Kilobytes (kb). So,we can conclude that Detected Flowers image storage space is less than Original Flowers image storage space and greater than Masked Flowers image storage space. Because, the detected image has only the object that have been detected by the trackbar and it displayed as a colored detected image. So, it has greater storage space than Masked Flowers image(Monochrome image or black and white image) and less storage space than Original Flowers image storage space.

Third section belongs to the Balls.png image's storage space. It contains three bar, first bar in the first section is Original Balls image storage space of 63 Kilobytes (kb). Second bar in the first section is Masked Balls image storage space of 13 Kilobytes(kb). Third bar in the first section is Detected Balls image storage space of 21 Kilobytes (kb). So,we can conclude that Detected Balls image storage space is less than Original Balls image storage space and greater than Masked Balls image storage space. Because, the detected image has only the object that have been detected by the trackbar and it displayed as a colored detected image. So, it has greater storage space than Masked Balls image(Monochrome image or black and white image) and less storage space than Original Balls image storage space. The object detection using HSV color space with machine learning libraries in python, should satisfy the below conditions.

$D < O \ \& \ D > M$ (i.e., Detected image storage space is less than Original image storage space and Detected image storage space is greater than Masked image storage space).

$O \neq D \ \& \ M$ (i.e., Original image storage space is not equal to Detected image storage space and Masked image storage space).

$D \neq O \ \& \ M$ (i.e., Detected image storage space is not equal to Original image storage space and Masked image storage space).

$M \neq O \ \& \ D$ (i.e., Masked image storage space is not equal to Original image storage space and Detected image storage space).

NOTE:

O => Original image storage space

D => Detected image storage space

M => Masked image storage space

Above, mentioned condition is for image's storage space occupied.

D. SAMPLE CODING:

```
# Import OpenCV package

# Import numpy package

import cv2

import numpy as np


# Define the call back function with Pass statement
def nothing(x):

    pass


# Creating an window with name for tracking bar window
cv2.namedWindow("Tracking")

# Create an trackbar bar window for Lower Hue
cv2.createTrackbar("LH","Tracking",0,255,nothing)

# Create an trackbar bar window for Lower Saturation
cv2.createTrackbar("LS","Tracking",0,255,nothing)

# Create an trackbar bar window for Lower Value
cv2.createTrackbar("LV","Tracking",0,255,nothing)

# Create an trackbar bar window for Upper Hue
cv2.createTrackbar("UH","Tracking",255,255,nothing)

# Create an trackbar bar window for Upper Saturation
cv2.createTrackbar("US","Tracking",255,255,nothing)

# Create an trackbar bar window for Upper Value
cv2.createTrackbar("UV","Tracking",255,255,nothing)
```

```

# Create an infinite while loop
while True:

    # Loading an colored image data into an frame window
    frame=cv2.imread('C://Users//sanjivi//Pictures//car.png')

    # Convert the colored image into HSV image
    hsv=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Getting an trackbar position for Lower Hue
    l_h=cv2.getTrackbarPos("LH","Tracking")

    # Getting an trackbar position for Lower Saturation
    l_s=cv2.getTrackbarPos("LS","Tracking")

    # Getting an trackbar position for Lower Value
    l_v=cv2.getTrackbarPos("LV","Tracking")

    # Getting an trackbar position for Upper Hue
    u_h=cv2.getTrackbarPos("UH","Tracking")

    # Getting an trackbar position for Upper Saturation
    u_s=cv2.getTrackbarPos("US","Tracking")

    # Getting an trackbar position for Upper Value
    u_v=cv2.getTrackbarPos("UV","Tracking")

    # Thresholding an lower boundation of HSV image
    l_b=np.array([l_h,l_s,l_v])

    # Thresholding an Upper boundation of HSV image
    u_b=np.array([u_h,u_s,u_v])

```

```
# Applying masking to an HSV image within a range of lower and upper boundation
mask=cv2.inRange(hsv,l_b,u_b)

# Apply bitwise and operation
res=cv2.bitwise_and(frame,frame,mask=mask)

# Display the result
cv2.imshow("frame",frame)
cv2.imshow("mask",mask)
cv2.imshow("res",res)

key=cv2.waitKey(1)
if key == 27:
    break

cv2.destroyAllWindows()
```