**D. J. Sanghvi College of Engineering**
**Department of Electronics & Telecommunication Engineering**

**Sem VI** **TE** **DECL- 604 Image Processing and Machine Vision Laboratory**

**Mini Project Report**

**ON**

# ‘Invisible Cloak Using Python’

Submitted by

| Student Name | SAP ID |
|---|---|
| Sahil Mangaonkar | 60002180090 |
| Mrinmayi Kadam | 60002198006 |
| Vatshal Shah | 60002198011 |
| Jash Varu | 60002198012 |

**DEPARTMENT**

**OF**

**ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

**Academic Year: 2020-2021**

# INDEX

## 1. INTRODUCTION:

Have you ever seen Harry Potter's Invisible Cloak? Was it wonderful? Have you ever wanted to wear that cloak? Here we have build the same cloak which Harry Potter uses to become invisible. Yes, we are not building it in a real way but it is all about graphics trickery. A cloak of invisibility is a fictional theme. In folklore, mythology and fairy tales, a cloak of invisibility appears either as a magical item used by duplicitous characters or an item worn by a hero to fulfill a quest. It is a common theme in Welsh and Germanic folklore, and may originate with the cap of invisibility seen in ancient Greek myths.

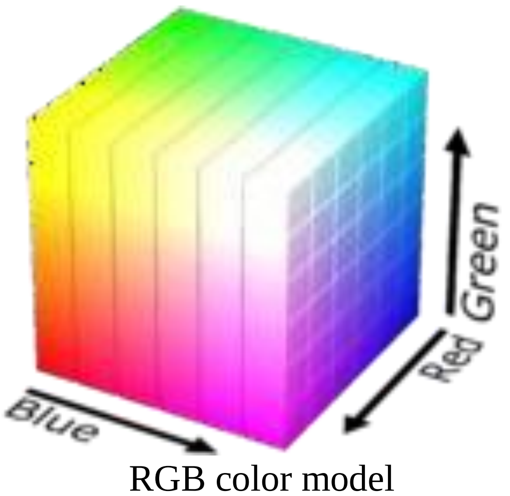
## 2. THEORY:

Here, we will create this magical experience using an image processing technique called Color detection and segmentation. You must have a cloth of same color and no other color should be visible into that cloth. We are taking the Blue cloth. If you are taking some other cloth, the code will remain the same but with minute changes.

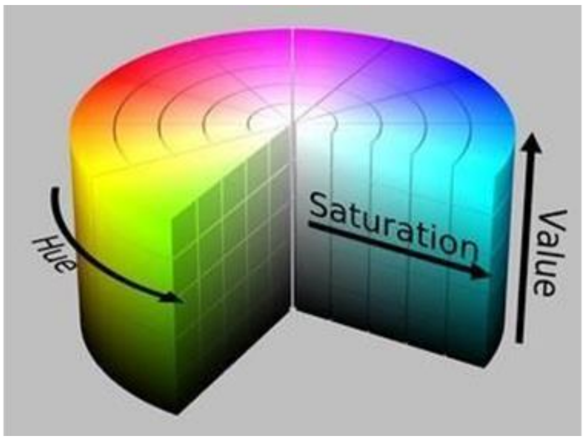
- **Color models and color spaces:**
  To understand the nature of something, it can be helpful to create a visual representation of the subject. In fact, humans tend to do this quite often, from scribbling notes in lectures, to drawing charts and maps to explain specific datasets. A color model is a visualization that depicts the color spectrum as a multidimensional model.

1. RGB color model:

RGB is a color model with three dimensions – red, green, and blue – that are mixed to produce a specific color. When defining colors in these dimensions, one has to know the sequence of colors in the color spectrum, e.g. that a mix of 100% red and green produces yellow.



RGB color model



HSV color model

2. HSV color model:

HSV is a cylindrical color model that remaps the RGB primary colors into dimensions that are easier for humans to understand. Like the Munsell Color System, these dimensions are hue, saturation, and value.

- Hue specifies the angle of the color on the RGB color circle. A 0° hue results in red, 120° results in green, and 240° results in blue.
- Saturation controls the amount of color used. A color with 100% saturation will be the purest color possible, while 0% saturation yields grayscale.
- Value controls the brightness of the color. A color with 0% brightness is pure black while a color with 100% brightness has no black mixed into the color. Because this dimension is often referred to as brightness, the HSV color model is sometimes called HSB.

- **Binary Image:**

A binary image is one that consists of pixels that can have one of exactly two colors, usually black and white. Binary images are also called bi-level or two- level. Binary images often arise in digital image processing as masks or thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bilevel computer displays, can only handle bilevel images. A binary image can be stored in memory as a bitmap, a packed array of bits.

Use:

- Image segmentation:

Binary images are produced from color images by segmentation. Segmentation is the process of assigning each pixel in the source image to two or more classes. If there are more than two classes then the usual result is several binary images. The simplest form of segmentation is probably Otsu's method which assigns pixels to foreground or background based on grayscale intensity. Edge detection also often creates a binary image with some pixels assigned to edge pixels.

## 3. IMPLEMENTATION:

The following project was implemented using Python. About the Software:

OpenCV (Open Source Computer Vision Library): OpenCV is an open source computer vision and machine learning software library. It is designed to solve computer vision problems and programming functions of OpenCV mainly aims at real-time computer vision. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception.

### OpenCV Functions:

Some of the OpenCV functions used are explained below:

#### I. cv2.cvtColor(src, code[, dst[, dstCn]]) -

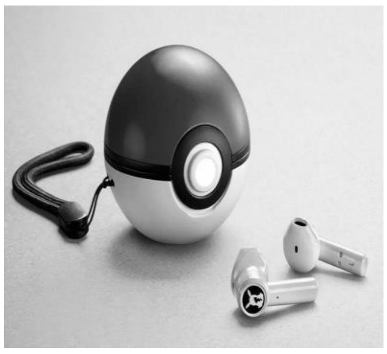
cv2.cvtColor() method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV. We will use some of color space conversion codes below. It can convert images from one color-space to another, like BGR Gray, BGR HSV etc.

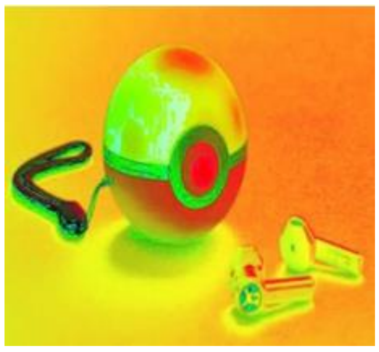
E.g.

BGR ↔ HSV = cv2.cvtColor(Original Image, cv2.COLOR_BGR2HSV)

BGR ↔ Gray = cv2.cvtColor(Original Image,cv2.COLOR_BGR2GRAY)

Original Image

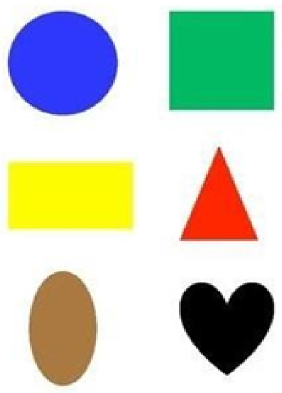

Gray Image


HSV Image

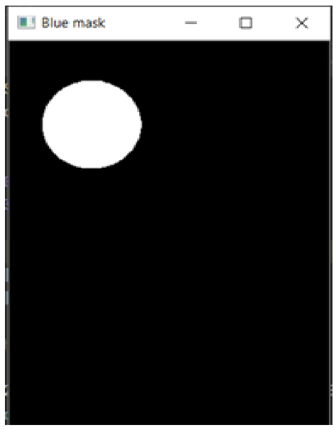
#### II. cv2.inRange(src, lowerb, upperb[, dst]) -

The cv2.inRange function expects three arguments: the first is the image were we are going to perform color detection, the second is the lower limit of the color you want to detect, and the third argument is the upper limit of the color you want to detect. After calling cv2.inRange, a binary mask is returned, where white pixels (255) represent pixels that fall into the upper and lower limit range and black pixels (0) do not.

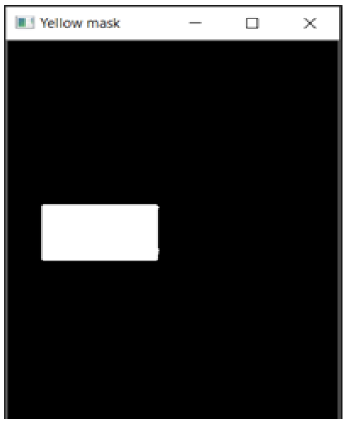
Original image



Mask for blue circle



Mask for yellow rectangle



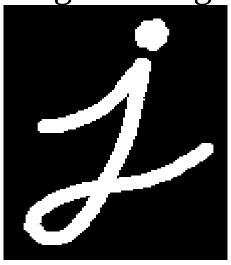
- For blue(in HSV color space)
  Lower range = (80,100,100)
  Upper range = (130,55,255)
- For Yellow(in HSV color space)
  Lower range = (25,100,100)
  Upper range = (40, 255, 255)

### III. cv2.morphologyEx(src, op, kernel, dst, anchor, iterations) -

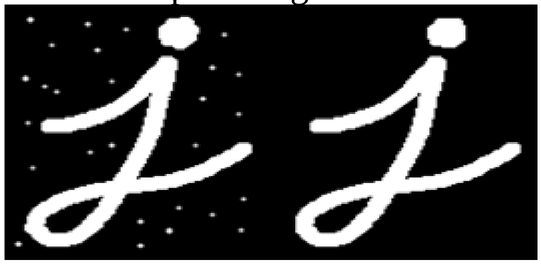
Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, and second one is called structuring element or kernel which decides the nature of operation.

- Open - Opening is erosion followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won’t come back, but our object area increases. It is also useful in joining broken parts of an object.
  E.g. Cv2.morphologyEx(src, op, kernel = cv2.MORPH_OPEN, dst, anchor, iterations)

Original image



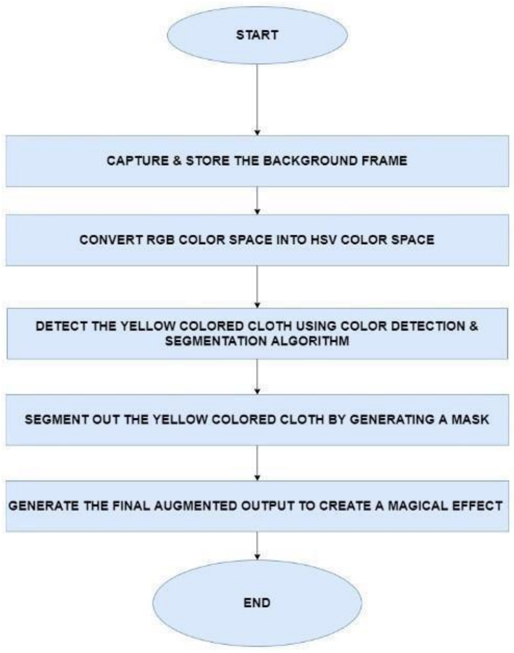
Open image



### IV. cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]):

The function addWeighted calculates the weighted sum of two arrays as follows:
dst(I)=saturate(src1(I)∗alpha+src2(I)∗beta+gamma)
It is use to overlap two images. E.g. The OpenCV logo is over lapped on the original image.

## Flow chart:


START
CAPTURE & STORE THE BACKGROUND FRAME
CONVERT RGB COLOR SPACE INTO HSV COLOR SPACE
DETECT THE YELLOW COLORED CLOTH USING COLOR DETECTION & SEGMENTATION ALGORITHM
SEGMENT OUT THE YELLOW COLORED CLOTH BY GENERATING A MASK
GENERATE THE FINAL AUGMENTED OUTPUT TO CREATE A MAGICAL EFFECT
END

In this project, we basically replaced the area covered by a particular cloth (blue cloth in our case) with the same area of pre-recorded background. This created an illusion that the part of body covered with the blue cloth is invisible.

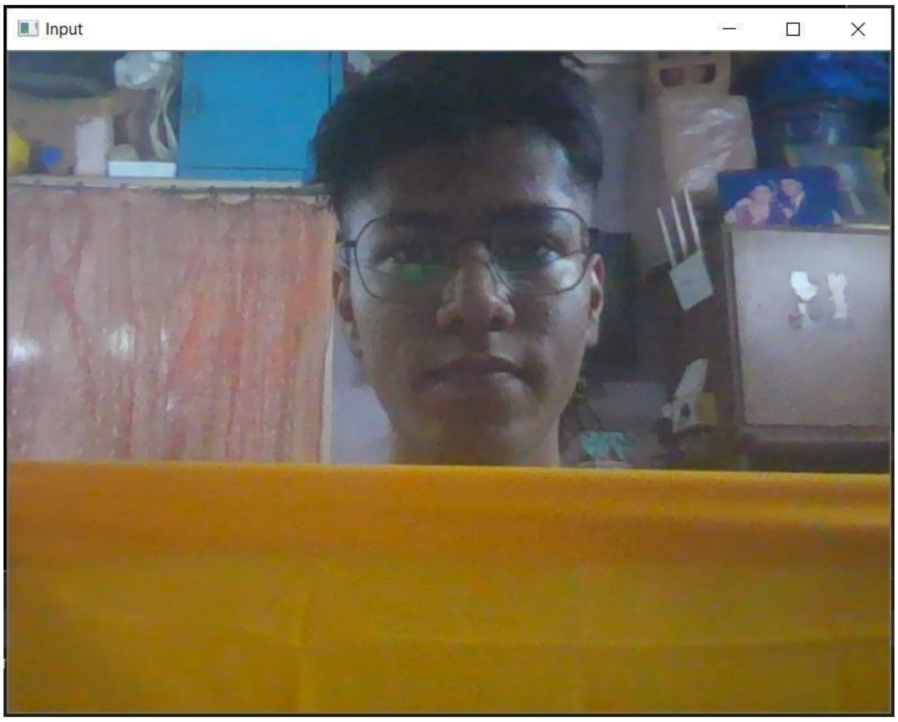
1. We captured capture and store the background frame using cap.read() function for some seconds. Flipped the background image over x-axis.
2. We captured the live frames & flipped it over x-axis using flip (img, 1) function.
3. We converted each live frame from BGR to HSV color space using cvtColor (img, cv2.COLOR_BGR2HSV) function.
4. We defined upper and lower range of blue color using numpy array.
5. We performed thresholding on the converted live image using upper and lower range of blue color with the help of inRange(HSV_Frame, lower_blue, upper_blue) function. This will create a mask with cloth covered part in white color & everything else will be in black color.
6. We reduced the noise present in mask1 using opening (morphological operation) with the help of cv2.morphologyEx() function.
7. We created another mask where cloth covered part will be in black color over white background by using bitwise_not(mask1) function.
8. We processed the live frame with mask2 to create an image where cloth covered part of the live frame will be replaced with black color. We used bitwise_and(img, img, mask=mask2) operation for this.

9. We processed the prerecorded background frame with mask1 to create an image where every part of the prerecorded background frame other than cloth covered part will be replaced with black color. We used bitwise_and(background, background, mask=mask1) operation for this.
10. We merged the results to create final output where the area of the live frame covered by a particular cloth (blue cloth in our case) is replaced by the same area of pre-recorded background to create an illusion that the part of body covered with the blue cloth is invisible. We used add Weighted(res1, 1, res2, 1, 0) function.

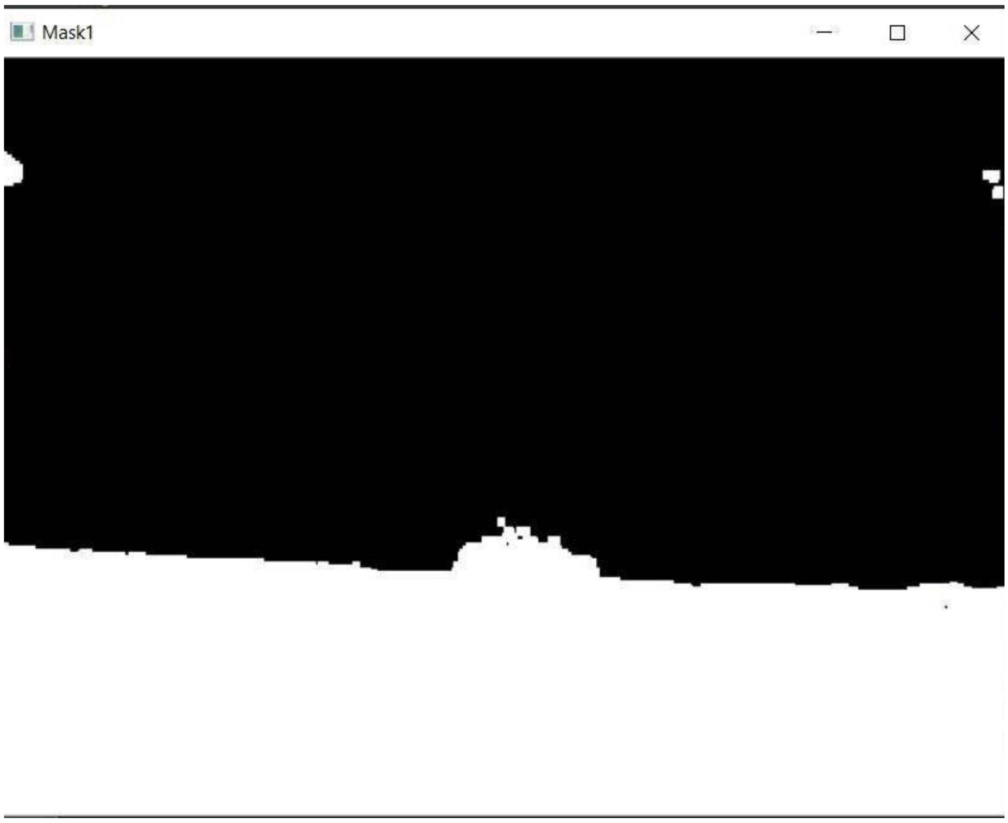
## 3. <u>CODE:</u>

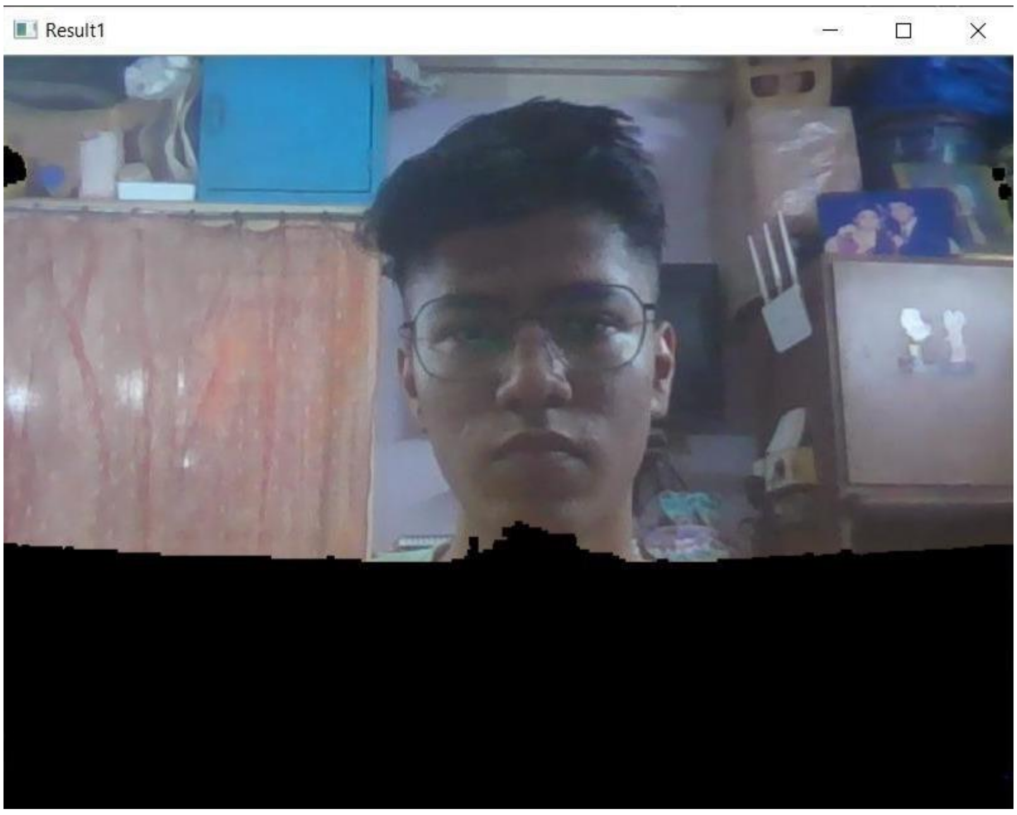
```
import cv2
import numpy as np

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

for i in range(30):
    ret, background = cap.read()

background = cv2.flip(background,1)

while True:
    #capturing the live frame
    ret, img = cap.read()
    img = cv2.flip(img, 1) #flipping the img over xaxis

    #coverting each frame from BGR to HSV color space
    HSV_Frame = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    #range
    lower_blue = np.array([70, 50, 50])
    upper_blue = np.array([179,255,255])
    mask1 = cv2.inRange(HSV_Frame, lower_blue, upper_blue)

    #To reduce the noise in the img
    mask1 = cv2.morphologyEx(mask1, cv2.MORPH_OPEN, np.ones((3, 3),
np.uint8))

    #creating a mask to display only blue cloth over black background
    mask2 = cv2.bitwise_not(mask1)
    res1 = cv2.bitwise_and(img, img, mask=mask2)
    #Taking the still background image and showing part of img where mask is
present
    res2 = cv2.bitwise_and(background, background, mask=mask1)
    #overlaping the two imgs


    final_output = cv2.addWeighted(res1, 1, res2, 1, 0)

    cv2.imshow("Mask1", mask1) #blue cloth will shown as white pixel over black
background
    cv2.imshow("Mask2", mask2)
    cv2.imshow("Result1", res1)
    cv2.imshow("Result2", res2)
    cv2.imshow("final_output", final_output)
    key = cv2.waitKey(1)
    if key == ord('s'):
        break

cv2.destroyAllWindows()
```

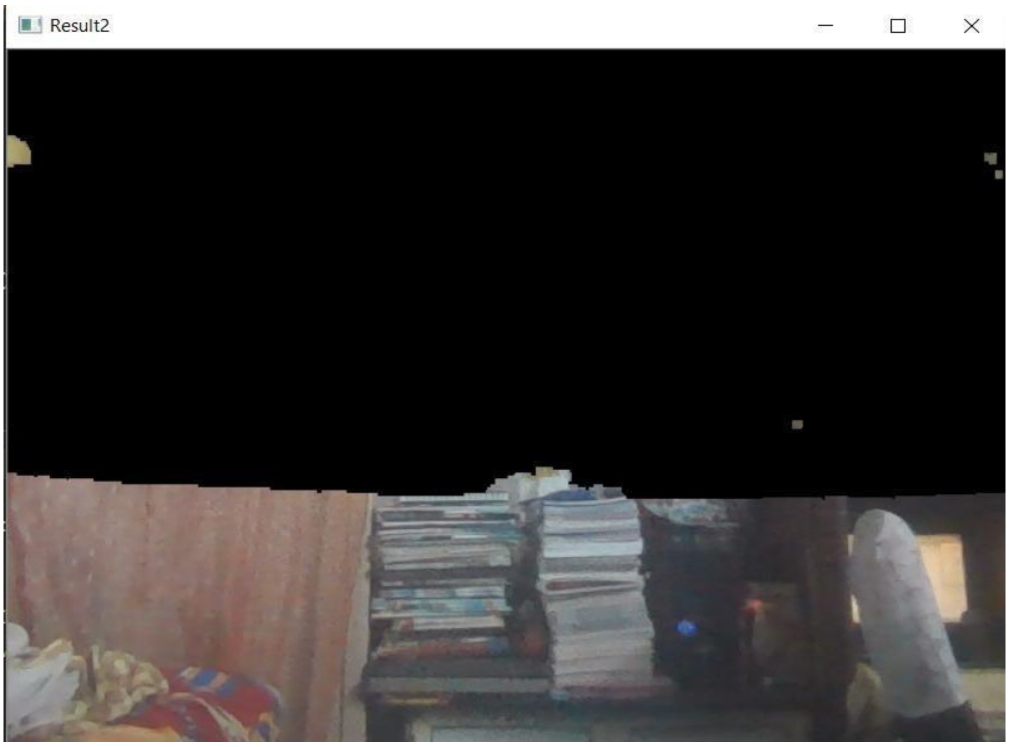
**INPUT:**

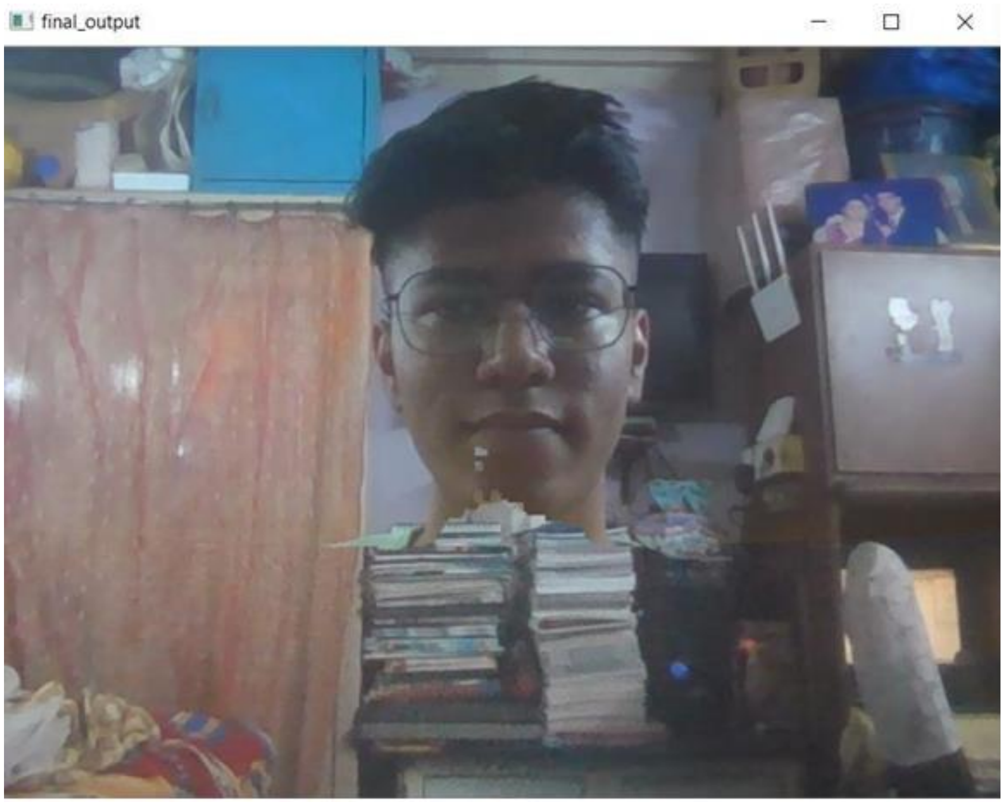


## 4. OUTPUT:









## 5. CONCLUSION:

Thus, with the help of color detection and segmentation in Image processing with is implemented in python using OpenCV library a yellow colored cloth is detected and the background is overlapped on it to give the magical effect of invisibility cloak.