DIGITAL_IMAGE_PROCESSING_LAB_WORKSHEET

OPERATIONS ON IMAGE

1.DRAW A LINE

In [1]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)
```

In [2]:

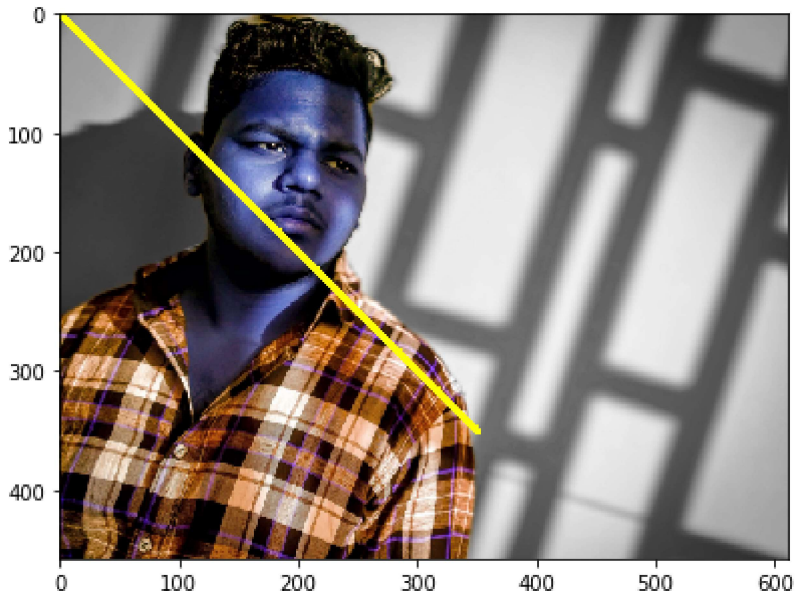
```
# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [3]:

```
# drawing a line on an Image usinig opencv module
cv.line(img_res,(0,0), (350,350), (458, 611, 3), 5)
# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x21829a7a308>
```



2. DRAW RECTANGLE BOX

In [4]:

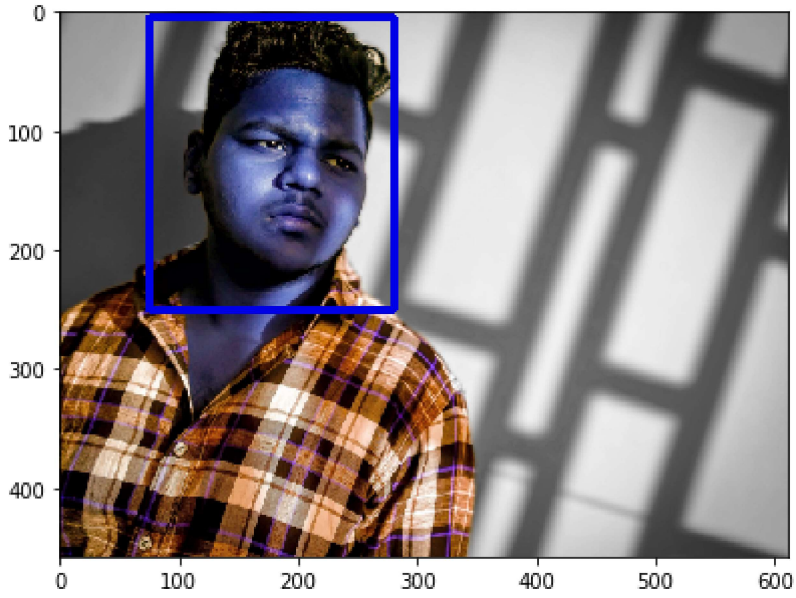
```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)

# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [5]:

```
# to draw the rectange on the plot
cv.rectangle(img_res, (75,5), (280,250), (0,0,229), 5)

# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x2182dd48488>
```



2. DRAW A CIRLCE

In [6]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)

# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [7]:

```
# to draw the circle on the plot
cv.circle(img_res, (350,150), 50, (0,214,0), -6)

# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x2182ddc0688>
```

3.WRITING TEXT ON IMAGE

In [8]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)

# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [9]:

```
# to write on am image
font = cv.FONT_HERSHEY_SIMPLEX
img_res = cv.putText(img_res, 'YADAV TECHNICAL CORP', (300,250), font, 0.8, (225,0,0), 4, c
```

In [10]:

```
# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x2182de22948>
```

In [ ]:

```

```

4.WORKING ON PIXELS

In [11]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)

# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [12]:

```
# Get the pixel of specific
px = img_res[55,5]
print(px)
```

```
[156 156 156]
```

In [13]:

```
# To modify that pixcel
img_res[55,5]= [180,110,150]
print(px)
```

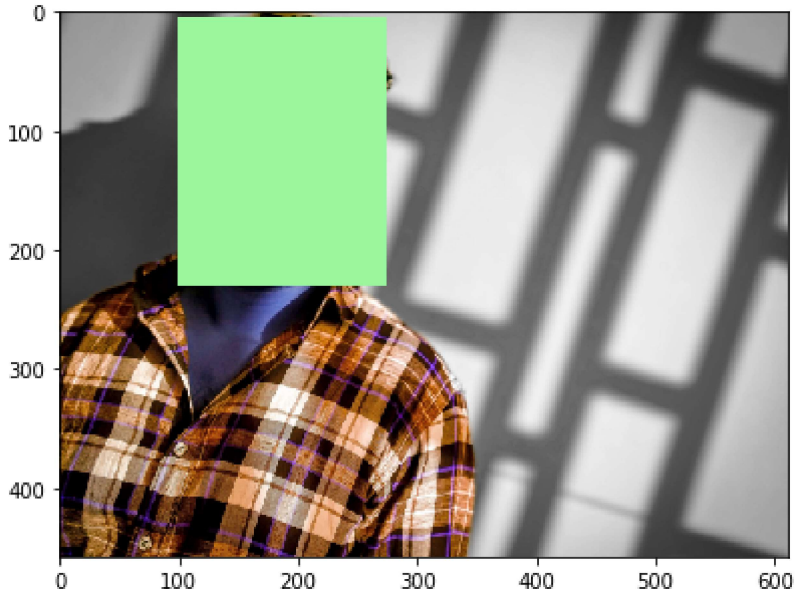
```
[180 110 150]
```

In [14]:

```
# Region of an Image
img_res[5:230, 100:275] = [155,247,157]
# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[14]:

```
<matplotlib.image.AxesImage at 0x2182de8bb88>
```



5.COPY AND PASTE AN IMAGE

In [15]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg', cv.IMREAD_COLOR)

# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

In [16]:

```
# To Copy and past an Image
copy_image = img_res[137:211, 209:294]
img_res[0:74, 0:85] = copy_image

# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x2182dee84c8>
```

6.CONVERSION OF IMAGE INTO DIFFERENT COLOR MODE

In [17]:

```
# Importing required modules
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
# Defining and reading an Image usinig opencv module
img = cv.imread('./COLOR_POP.jpg',1)
# RESIZE IMAGE TO FIT IN SCREEN
scale_percent = 20 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
# resize image
img_res = cv.resize(img, dim, interpolation = cv.INTER_AREA)
```

6.1 PLOT INTO GBR MODE

In [18]:

```
# Ploting and Displaying the image
plt.figure(figsize=(20,5))
plt.imshow(img_res)
```

Out[18]:

```
<matplotlib.image.AxesImage at 0x2182f173c08>
```

6.2.BGR TO RGB MODE

In [19]:

```
# Convert GBR color mode to RGB color mode
RGBimg = cv.cvtColor(img_res, cv.COLOR_BGR2RGB)
# Ploting and Displaying the image
plt.figure(figsize=(20,8))
plt.imshow(RGBimg)
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x2182ef9ab88>
```

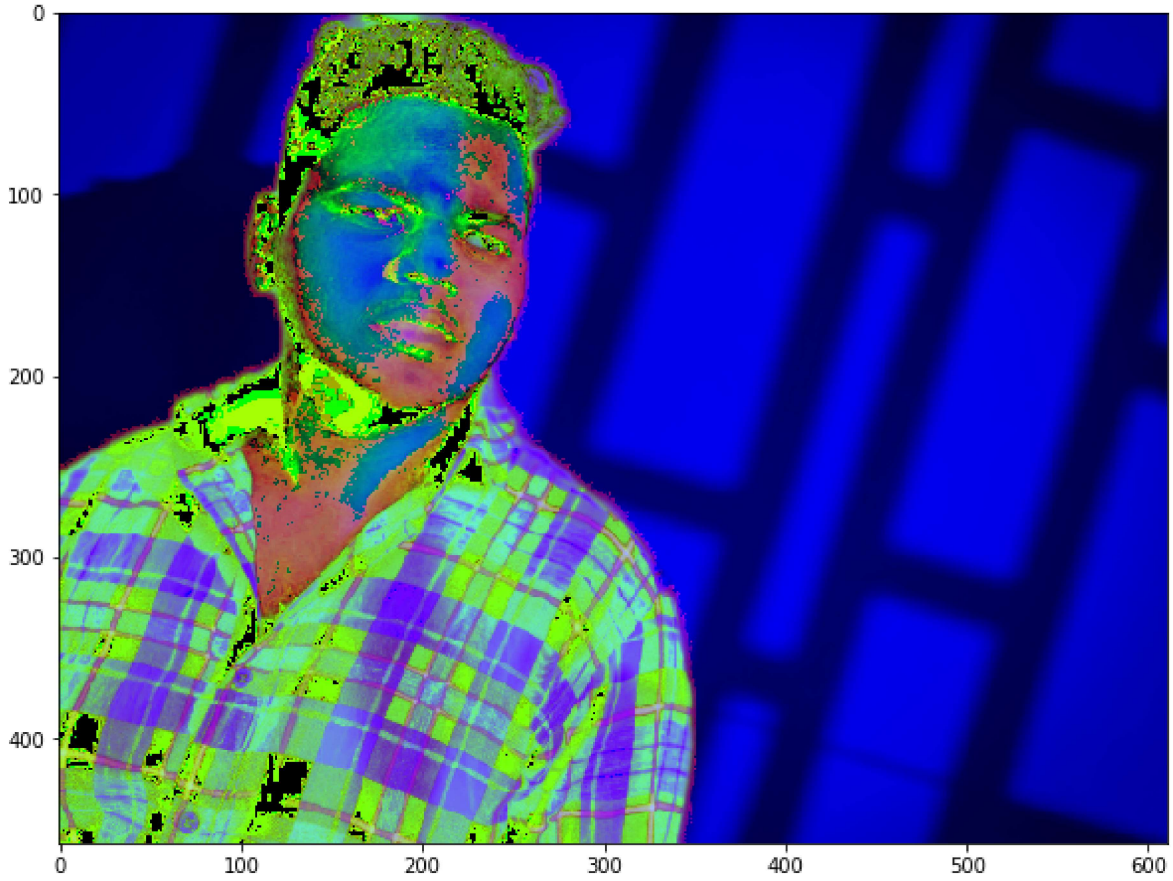
6.3 RGB TO HSV COLOR MODE

In [20]:

```
#Convert RGB color mode to HSV color mode
HSVimg = cv.cvtColor(RGBimg, cv.COLOR_RGB2HSV)
# Ploting and Displaying the image
plt.figure(figsize=(10,10))
plt.imshow(HSVimg)
```

Out[20]:

```
<matplotlib.image.AxesImage at 0x2182f002508>
```



In [ ]:

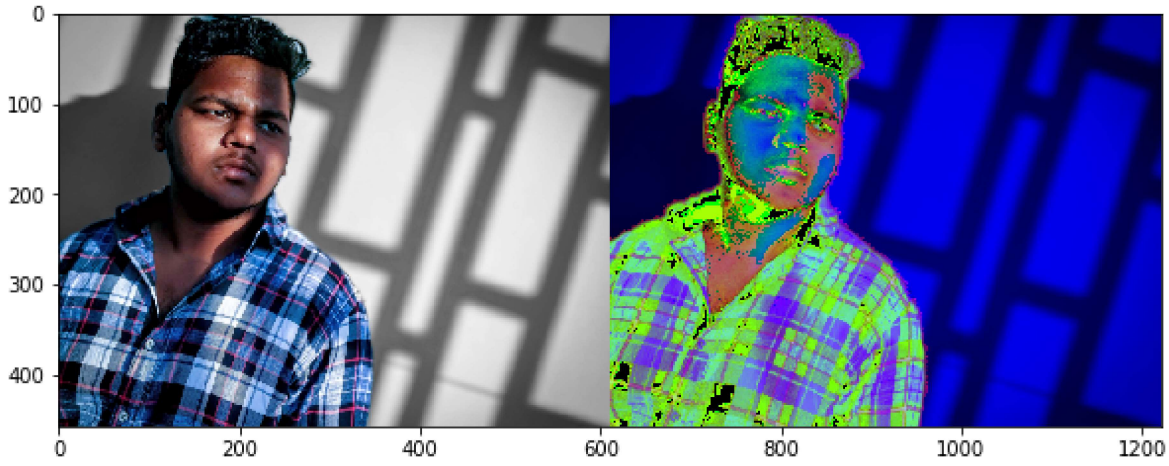
7.DETECTION OF COLOR PRESENT IN IMAGE

In [21]:

```
lower = np.array([150,150,350])
upper = np.array([205,255,355])
mask = cv.inRange(HSVimg, lower, upper)
output = cv.bitwise_and(HSVimg, RGBimg, mask = mask)

#diplay the image
plt.figure(figsize=(10,10))
plt.imshow(np.hstack([RGBimg, HSVimg]))
```

Out[21]:

```
<matplotlib.image.AxesImage at 0x2182f345c88>
```



In [22]:

```
# plot it through matplotlib
plt.figure(figsize=(10,10))
plt.imshow(np.hstack([HSVimg, RGBimg]))
```

Out[22]:

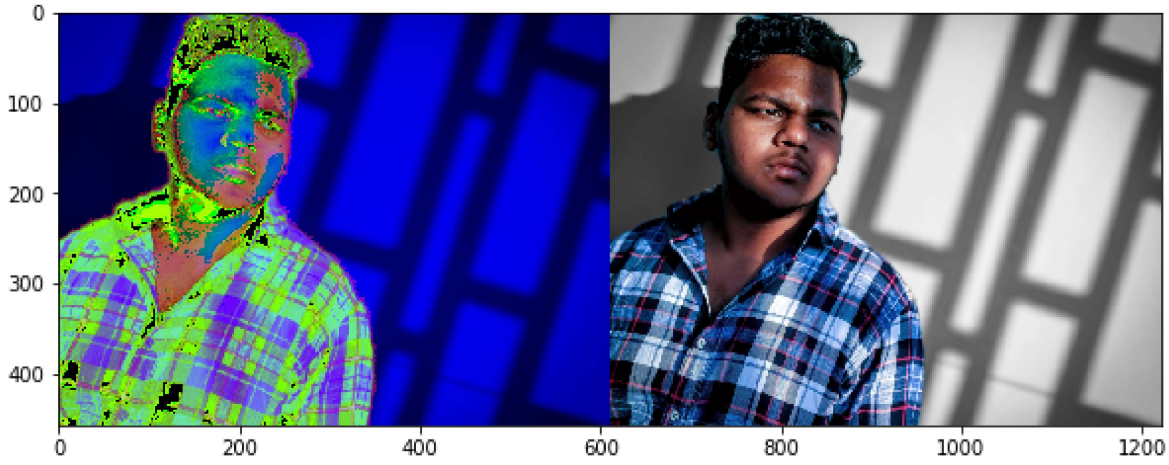
```
<matplotlib.image.AxesImage at 0x2182f1c9f88>
```

In [23]:

```
# plot it through matplotlib
plt.figure(figsize=(10,10))
plt.imshow(np.hstack([HSVimg, RGBimg]))
```

Out[23]:

```
<matplotlib.image.AxesImage at 0x2182f22c4c8>
```

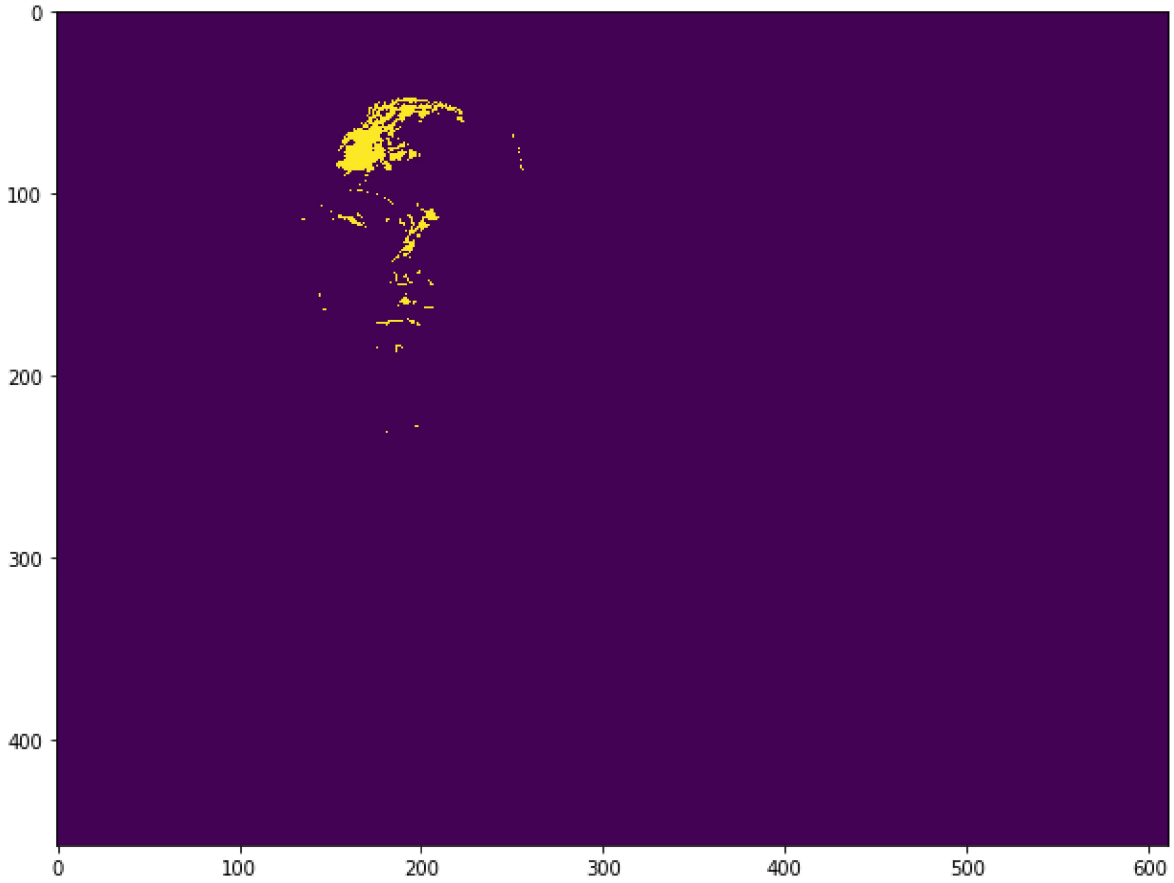


In [24]:

```
# changing the array location
lower = np.array([0,160,50])
upper = np.array([10,255,255])
# To detetic a specific color eg: yellow
mask = cv.inRange(HSVimg, lower, upper)
# Ploting and Displaying the image
plt.figure(figsize=(20,8))
plt.imshow(mask)
```

Out[24]:

```
<matplotlib.image.AxesImage at 0x2182f290708>
```



8.COLOR DETECTION IN LIVE CAPTURING IMAGE/VEDIO

8.1 RGB TO HSV

In [25]:

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('livecolor.mp4',fourcc, 20.0, (640,480))


# Loop untill the end of the video
while (cap.isOpened()):

    # Capture frame-by-frame
    ret, frame = cap.read()
    frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                        interpolation = cv2.INTER_AREA)


    hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
    lower_red = np.array([110,50,50])
    upper_red = np.array([130,255,255])
    # Here we are defining range of bluecolor in HSV
    # This creates a mask of blue coloured
    # objects found in the frame.
    mask = cv2.inRange(hsv, lower_red, upper_red)
    # The bitwise and of the frame and mask is done so
    # that only the blue coloured objects are highlighted
    # and stored in res
    res = cv.bitwise_and(frame,frame, mask= mask)
    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    # define q as the exit button
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
  # release the video capture object
cap.release()
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-25-9fdd1e138397> in <module>
     17     ret, frame = cap.read()
     18     frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
---> 19                        interpolation = cv2.INTER_AREA)
     20
     21

error: OpenCV(3.4.2) C:\projects\opencv-python\opencv\modules\imgproc\src\re
size.cpp:4044: error: (-215:Assertion failed) !ssize.empty() in function 'c
v::resize'
```

8.2 RGB TO BGR

In [27]:

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('livecolor.mp4',fourcc, 20.0, (640,480))


# Loop untill the end of the video
while (cap.isOpened()):

    # Capture frame-by-frame
    ret, frame = cap.read()
    frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                        interpolation = cv2.INTER_AREA)


    bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    lower_red = np.array([110,50,50])
    upper_red = np.array([130,255,255])
    # Here we are defining range of bluecolor in GBR
    # This creates a mask of blue coloured
    # objects found in the frame.
    mask = cv2.inRange(bgr, lower_red, upper_red)
    # The bitwise and of the frame and mask is done so
    # that only the blue coloured objects are highlighted
    # and stored in res
    res = cv.bitwise_and(frame,frame, mask= mask)
    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    # define q as the exit button
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
  # release the video capture object
cap.release()
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-27-f4010473d3d0> in <module>
     17     ret, frame = cap.read()
     18     frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
---> 19                        interpolation = cv2.INTER_CUBIC)
     20
     21

error: OpenCV(3.4.2) C:\projects\opencv-python\opencv\modules\imgproc\src\re
size.cpp:4044: error: (-215:Assertion failed) !ssize.empty() in function 'c
v::resize'
```

BGR TO HSV

In [ ]:

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('livecolor.mp4',fourcc, 20.0, (640,480))


# Loop untill the end of the video
while (cap.isOpened()):

    # Capture frame-by-frame
    ret, frame = cap.read()
    frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                        interpolation = cv2.INTER_AREA)


    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_red = np.array([110,50,50])
    upper_red = np.array([130,255,255])
    # Here we are defining range of bluecolor in GBR
    # This creates a mask of blue coloured
    # objects found in the frame.
    mask = cv2.inRange(hsv, lower_red, upper_red)
    # The bitwise and of the frame and mask is done so
    # that only the blue coloured objects are highlighted
    # and stored in res
    res = cv.bitwise_and(frame,frame, mask= mask)
    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    # define q as the exit button
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
  # release the video capture object
cap.release()
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```

THANKS FOR READING MY FILE THAT'S ALL WHAT I LEARN TILL NOW

In [ ]:

```
# importing the necessary libraries
import cv2
import numpy as np

# Creating a VideoCapture object to read the video
cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.mp4',fourcc, 20.0, (640,480))


# Loop untill the end of the video
while (cap.isOpened()):

    # Capture frame-by-frame
    ret, frame = cap.read()
    frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                        interpolation = cv2.INTER_CUBIC)


    hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
    lower_red = np.array([110,50,50])
    upper_red = np.array([130,255,255])
    # Here we are defining range of bluecolor in HSV
    # This creates a mask of blue coloured
    # objects found in the frame.
    mask = cv2.inRange(hsv, lower_red, upper_red)
    # The bitwise and of the frame and mask is done so
    # that only the blue coloured objects are highlighted
    # and stored in res
    res = cv.bitwise_and(frame,frame, mask= mask)
    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)
    # define q as the exit button
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
  # release the video capture object
cap.release()
# Closes all the windows currently opened.
cv2.destroyAllWindows()
```