# CS 682: COMPUTER VISION

## Abhishek Bodas

## G01160204

## HW1

**Website:** http://mason.gmu.edu/~abodas/vision/

Username: CS682

Password: abodas682

## 1) Installing opencv2

i) Installing python 3.8.1 successfully.
ii) Installing editors and plugins: Atom and eclipse with pydev.
iii) Configuring opencv2
iv) Writing sample program with images to makes sure everything is proper
v) Coding Assignment programs

## 2) Grayscale image

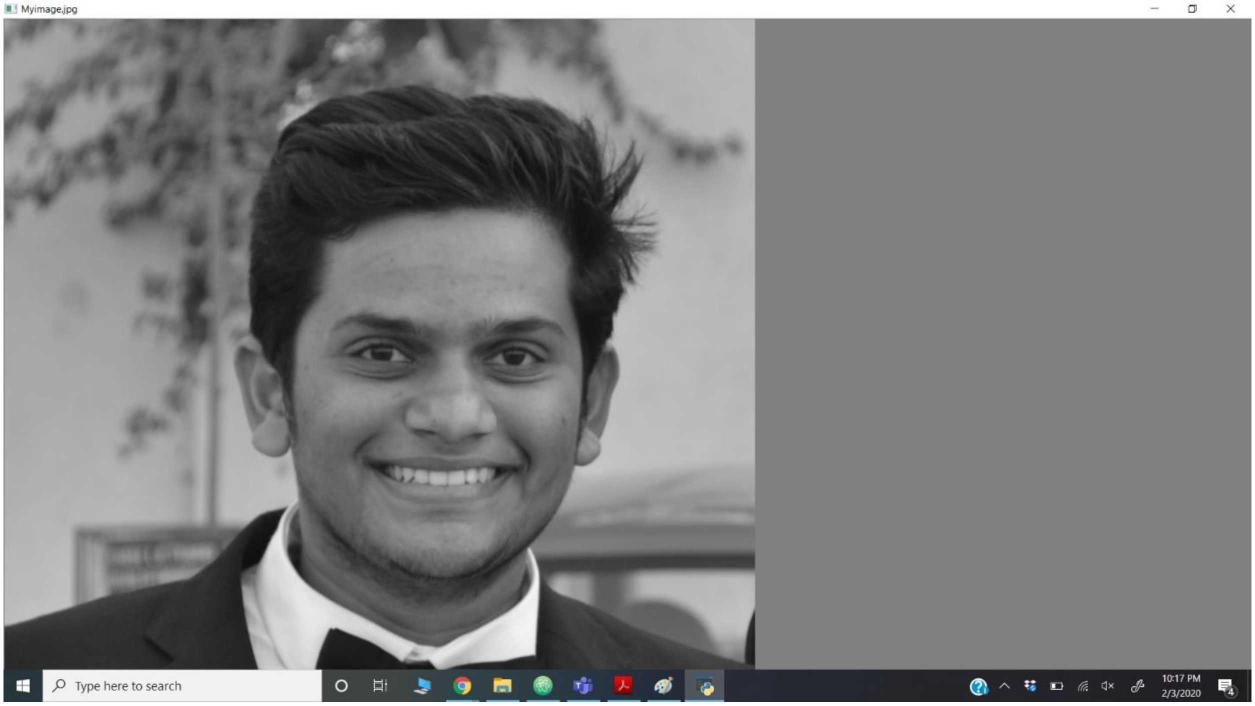
Converting a colored image to grayscale

Code:

```
#Script to cause dilation in a gray scale image

import numpy as np #To import numpy library

import cv2 #To import cv2 module

image = cv2.imread('myimage.jpg',0) #opening gray scale image

#To return matrix of ones

matrix = np.ones((10,10),np.int8) #matrix of size 10 and data type int8

dilation = cv2.dilate(image,matrix,iterations = 3)#To dilate the image by iterating thrice

cv2.imshow('myimage.jpg',dilation) #To display the image

key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

   cv2.destroyAllWindows()
```

```
    elif key==ord('q'): #Press 'q' key to quit the image window
       cv2.destroyAllWindows()
    elif key==ord('e'):#Press 'e' key to exit the image window
       cv2.destroyAllWindows()
    elif key==ord('x'):#Press 'x' key to cancel the image window
       cv2.destroyAllWindows()
    cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



# 3) Image transformations

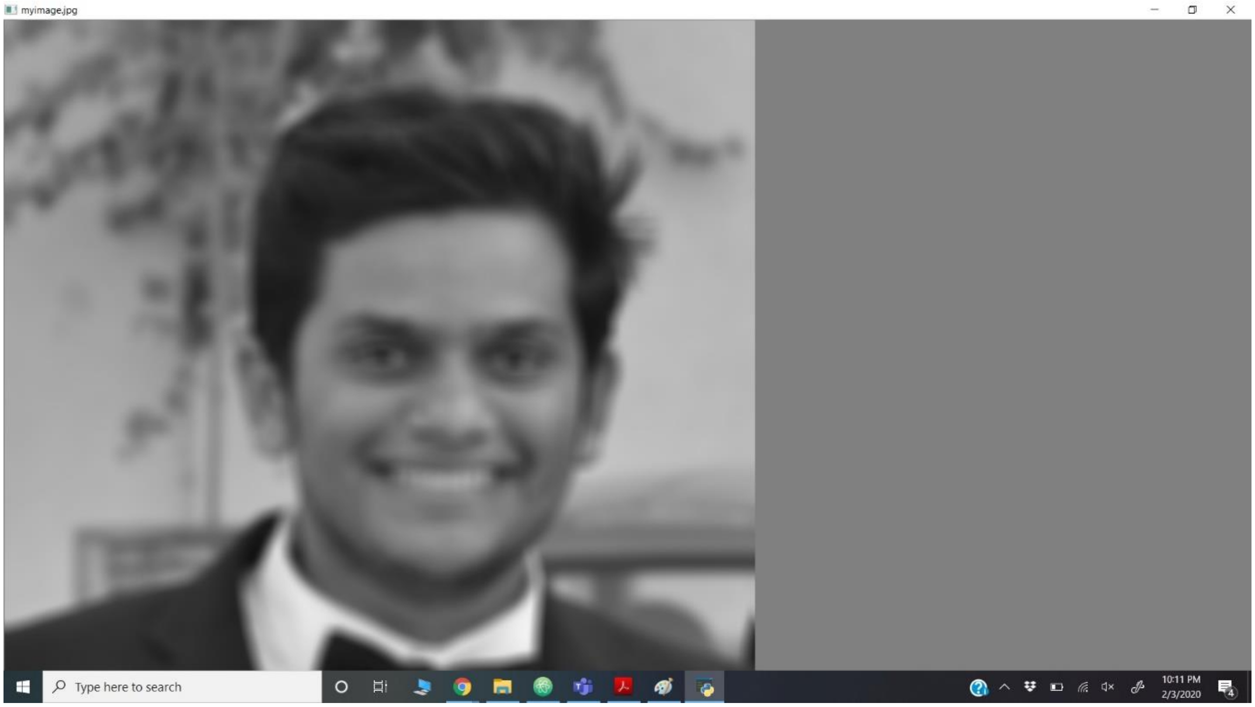
**i) Blurring: to blur the original image**

code:

```
#Script to blur a image in grayscale
import cv2 #To import cv2 module
```

```
image = cv2.imread('myimage.jpg',0) #opening image in gray scale

blur = cv2.blur(image,(20,20)) #To blur the image

cv2.imshow('myimage.jpg',blur)

key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

   cv2.destroyAllWindows()

elif key==ord('q'): #Press 'q' key to quit the image window

   cv2.destroyAllWindows()

elif key==ord('e'):#Press 'e' key to exit the image window

   cv2.destroyAllWindows()

elif key==ord('x'):#Press 'x' key to cancel the image window

   cv2.destroyAllWindows()

cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



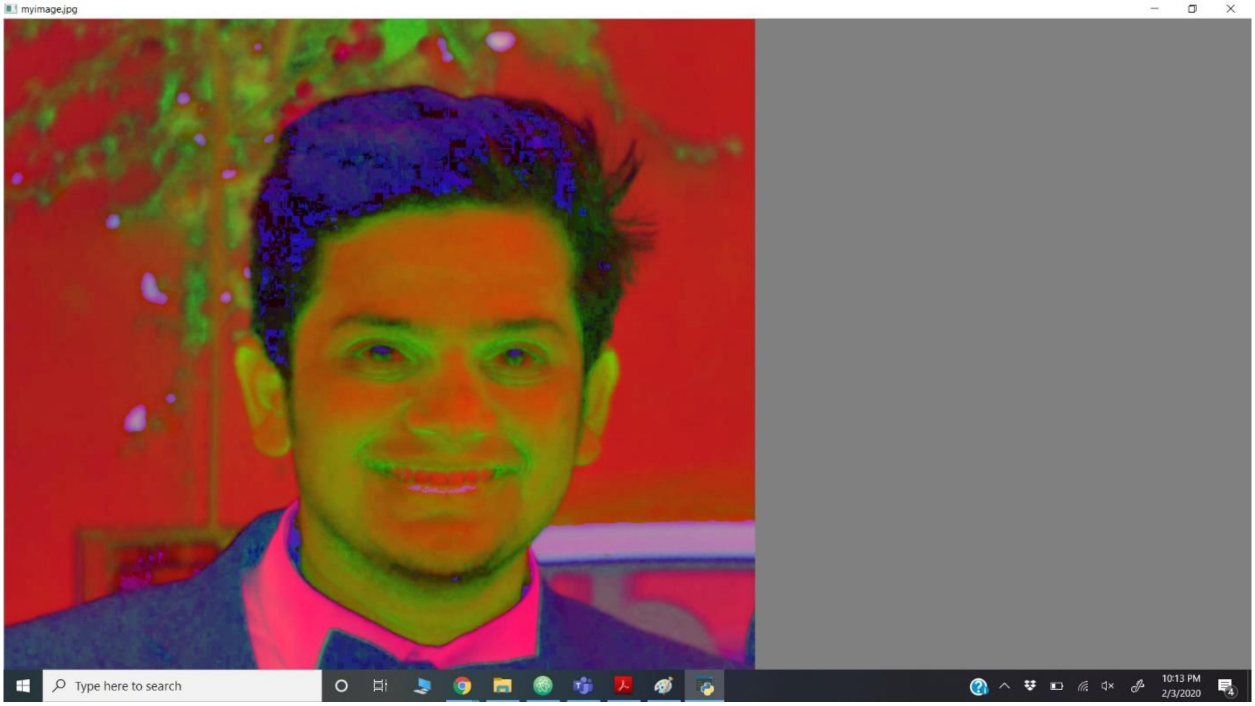
**ii) changing colorspaces**

converting image to HSV(Hue saturation and value)

code:

```
#Script to convert an colored image to Hue, Saturation and Value(HSV)
import cv2 #To import cv2 module
image = cv2.imread('myimage.jpg',1) #To read the image
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)#To convert image into HSV
cv2.imshow('myimage.jpg',hsv)
key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems
if key==27: #Press Escape key to close the image window
   cv2.destroyAllWindows()
elif key==ord('q'): #Press 'q' key to quit the image window
   cv2.destroyAllWindows()
elif key==ord('e'):#Press 'e' key to exit the image window
   cv2.destroyAllWindows()
elif key==ord('x'):#Press 'x' key to cancel the image window
   cv2.destroyAllWindows()
cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



### iii) Erosion

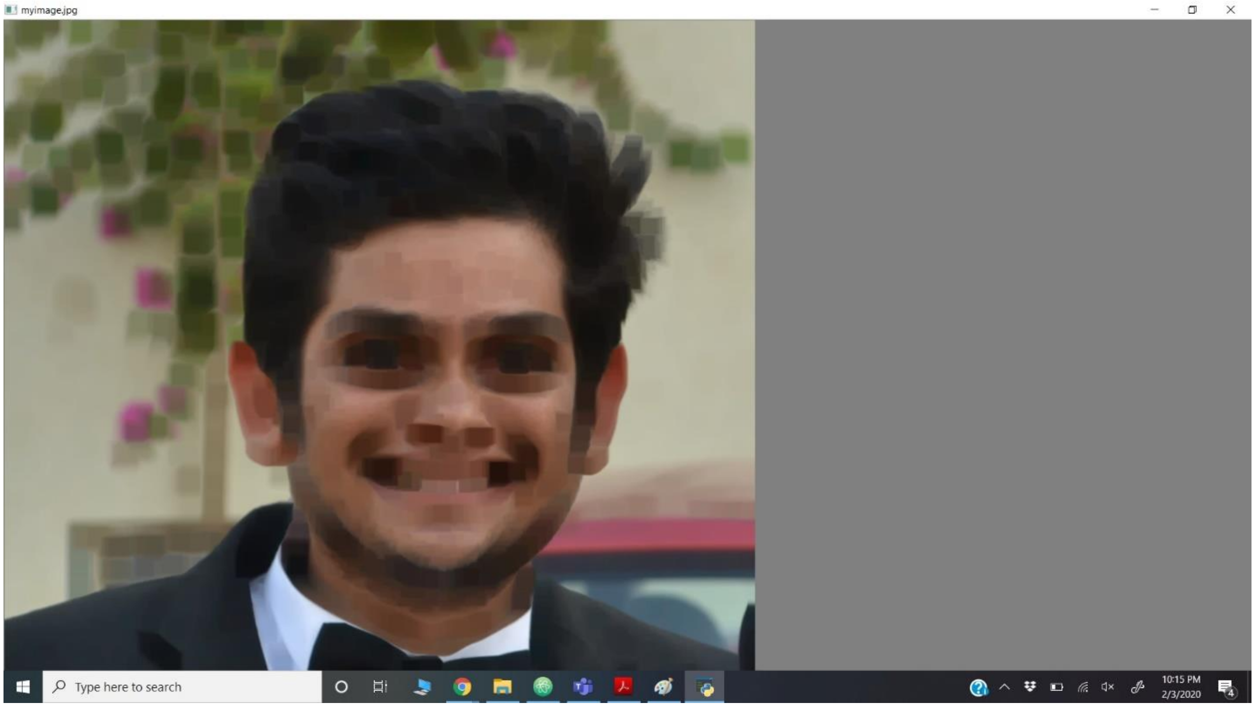
to erode the image

code:

```
#Script to cause erosion in a colored image

import numpy as np #To import numpy library

import cv2 #To import cv2 module

image = cv2.imread('myimage.jpg',1) #opening colored image

#To return matrix of ones

matrix = np.ones((10,10),np.int8) #matrix of size 10 and data type int8

erosion = cv2.erode(image,matrix,iterations = 2)#To erode the image by 2 iterating twice

cv2.imshow('myimage.jpg',erosion) #To display the image

key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

   cv2.destroyAllWindows()

elif key==ord('q'): #Press 'q' key to quit the image window
```

```
    cv2.destroyAllWindows()
elif key==ord('e'):#Press 'e' key to exit the image window
    cv2.destroyAllWindows()
elif key==ord('x'):#Press 'x' key to cancel the image window
    cv2.destroyAllWindows()
cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



**iv)Dilation**

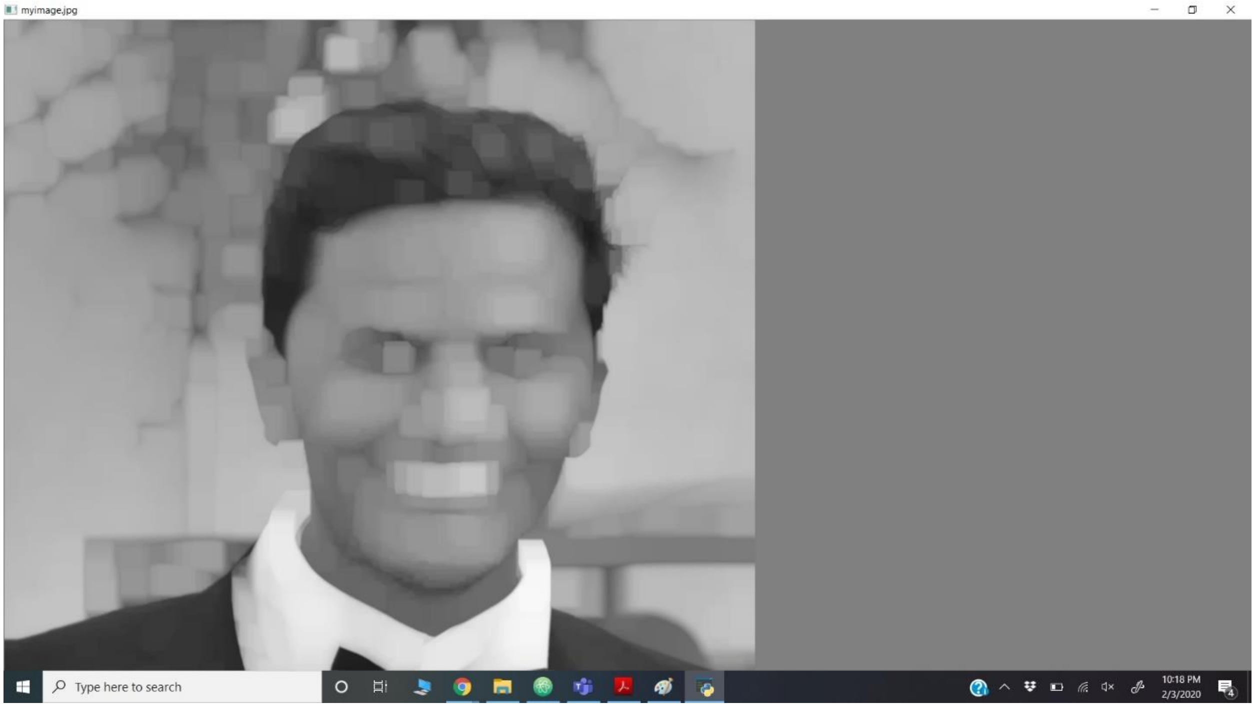
To dilate the image

Code:

```
#Script to cause dilation in a gray scale image
import numpy as np #To import numpy library
import cv2 #To import cv2 module
image = cv2.imread('myimage.jpg',0) #opening gray scale image
#To return matrix of ones
```

```
matrix = np.ones((10,10),np.int8) #matrix of size 10 and data type int8
dilation = cv2.dilate(image,matrix,iterations = 3)#To dilate the image by iterating thrice
cv2.imshow('myimage.jpg',dilation) #To display the image
key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems
if key==27: #Press Escape key to close the image window
   cv2.destroyAllWindows()
elif key==ord('q'): #Press 'q' key to quit the image window
   cv2.destroyAllWindows()
elif key==ord('e'):#Press 'e' key to exit the image window
   cv2.destroyAllWindows()
elif key==ord('x'):#Press 'x' key to cancel the image window
   cv2.destroyAllWindows()
cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



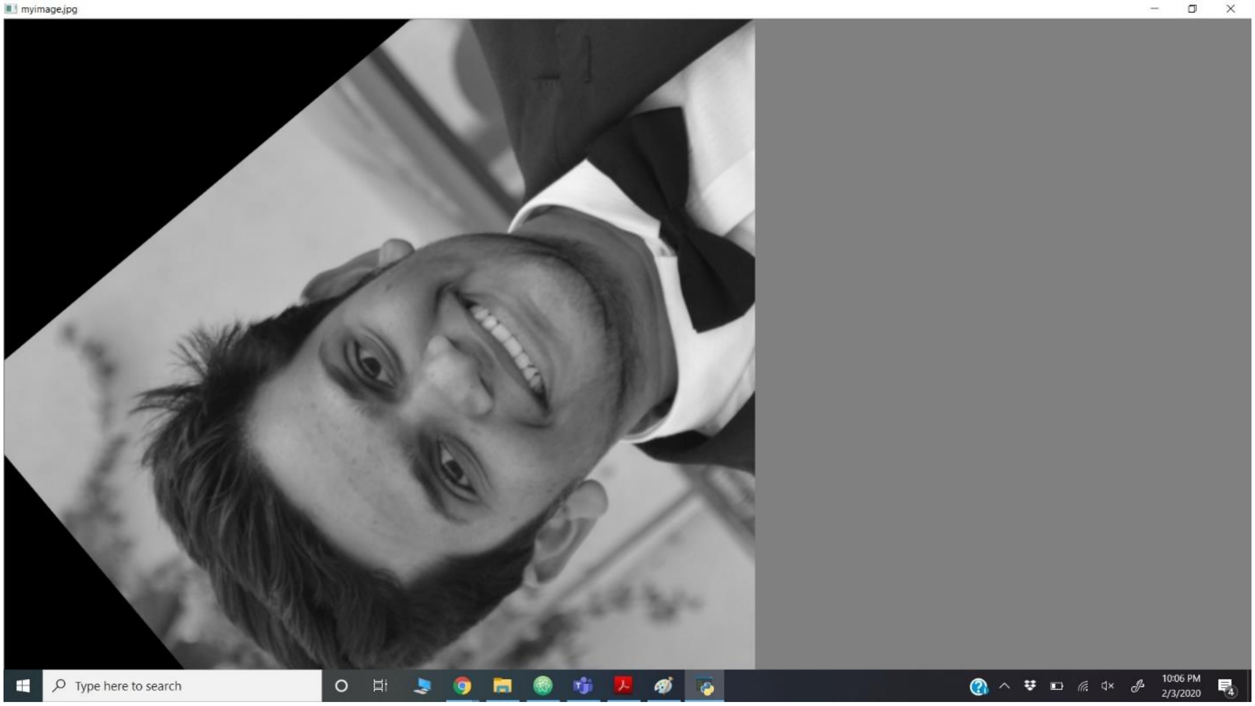
**v)Rotate**

To rotate the image

Code:

```
#Script to rotate a gray scale image

import cv2 #To import cv2 module

image = cv2.imread('myimage.jpg',0) #opening image in gray scale

row,column=image.shape #Returns a tuple of width and height

print(row,column) #To get the height and width of the image

rotate=cv2.getRotationMatrix2D((542,462),130,1) #Rotating the matrix and specifying center coordinates,angle and scale factor

out = cv2.warpAffine(image,rotate,(column,row)) #size of output image

cv2.imshow('myimage.jpg',out) #To display the image

key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems

if key==27: #Press Escape key to close the image window

   cv2.destroyAllWindows()

elif key==ord('q'): #Press 'q' key to quit the image window

   cv2.destroyAllWindows()

elif key==ord('e'):#Press 'e' key to exit the image window

   cv2.destroyAllWindows()

elif key==ord('x'):#Press 'x' key to cancel the image window

   cv2.destroyAllWindows()

cv2.destroyAllWindows() #To destroy windows anyway
```

output:



# 4) Gaussian pyramid

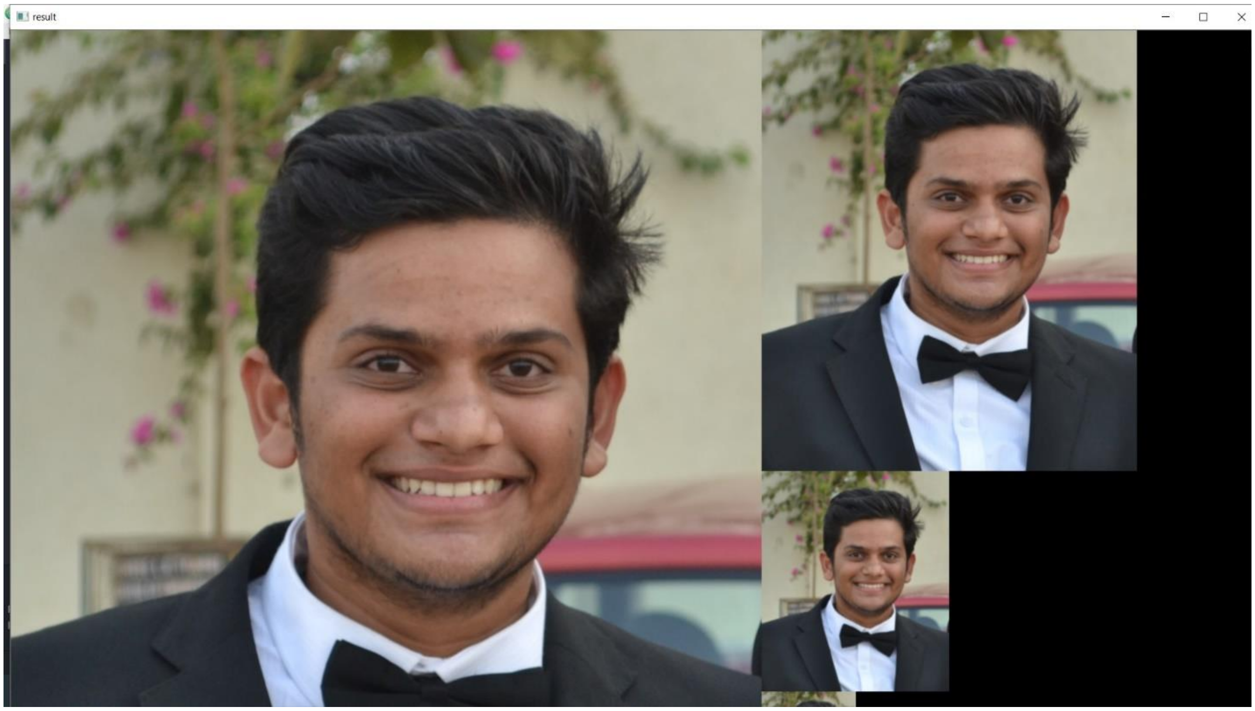
To create a gaussian pyramid of an image

Code:

```
#Script to display gaussian pyramid of an image

import cv2

import numpy as np

image = cv2.imread('myimage.jpg')

copy = image.copy()

space=int(image.shape[0])*int(image.shape[1])

#Taking image dimensions

row=copy.shape[0]

column=copy.shape[1]

channel=copy.shape[2]
```

```
newimage = np.zeros((row,column,channel), dtype=np.uint8) #zero array image
merge = np.concatenate((copy,newimage), axis=1) #merging the original image with a darken image of same size
height = 0
width = image.shape[1]
for i in range(0,5):

    lowerimage = cv2.pyrDown(image) #lowering image resolution
    space += int(lowerimage.shape[0])*int(lowerimage.shape[1])
    merge[height: height + lowerimage.shape[0], width: width + lowerimage.shape[1]]=lowerimage #appending size
    height += lowerimage.shape[0]
    image = lowerimage
    i+=1
print("space requirement for the pyramid is:",space)
cv2.imshow("result", merge)
size=int(merge.size/3)
print("size of smallest rectangular image is:",size)
key=cv2.waitKey(10000) & 0xFF #Display duration = 10 seconds & Mask for 64-bit systems
if key==27: #Press Escape key to close the image window
    cv2.destroyAllWindows()
elif key==ord('q'): #Press 'q' key to quit the image window
    cv2.destroyAllWindows()
elif key==ord('e'):#Press 'e' key to exit the image window
    cv2.destroyAllWindows()
elif key==ord('x'):#Press 'x' key to cancel the image window
    cv2.destroyAllWindows()
cv2.destroyAllWindows() #To destroy windows anyway
```

**output:**



**Space requirement for the pyramid is: 1334403 pixels**

**Size of smallest rectangular image is:2001384 pixels**

**5) Application**

Application of Computer Vision:

https://www.cc.gatech.edu/~thad/p/032_20_ARVR/stochastic_ISWC97.pdf

**"Stochasticks": Augmenting the Billiards Experience with Probabilistic Vision and Wearable Computers**

Description:

Wearable Augmented Reality (AR) application of Computer Vision to play the game of pool/billiards. It implements an autonomous probabilistic vision algorithm to function. Some Vision processing techniques used here are Color Feature Detection, Contour Computation, Symmetry Detection, Color Model Classification, and Edge Detection. Basically, it assists the player in planning and aiming the ball in the pockets by maintaining visual sensing. The system displays a graphical output that helps in shot suggestion and assist targeting.