CS682 Homework 1 Report

Name: Rashika Koul
Webpage: http://mason.gmu.edu/~rkoul2/CS682/ **Username:** cs682 **Password:** 120196

1.1 OpenCV was installed successfully.

1.2 For converting the **color image into a gray scale image** the method I used was cvtColor from OpenCV library with cv2.COLOR_BGR2GRAY as the flag in the second argument.

```
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
```

1.3 I did 6 **transformations/changes** namely, blurring, binary thresholding, switching color planes from bgr to hsv, affine transformation, rotation and erosion.

1. Blurring: For blurring, the method I used was blur from OpenCV library with 5*5 low-pass filter kernel as the second argument. This was done on the original color image namely "face.jpg".

```
blur = cv2.blur(image,(5,5))
```

2. Binary thresholding: For binary thresholding, the method I used was threshold from the OpenCV library with 127 as the threshold value and cv2.THRESH_BINARY as the flag in the last argument.

```
ret,threshold = cv2.threshold(image,127,255,cv2.THRESH_BINARY)
```

3. BGR to HSV: For converting the BGR color image into a HSV color plane image the method I used was cvtColor from OpenCV library with cv2.COLOR_BGR2HSV as the flag in the second argument.

```
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

4. Affine Transform: In affine transform the parallel lines in the original image will remain parallel in the transformed image. I provided set of 3 points from the original image and a set of their desired corresponding locations in the output image to the method getAffineTransform from the OpenCV library.

```
M = cv2.getAffineTransform(pts1,pts2)
```

The value returned by it is passed to the warpAffine method from the OpenCV library along with the number of rows and columns in the original image.

```
transform = cv2.warpAffine(image,M,(cols,rows))
```

5. Erosion: In erosion, the boundaries of the bright colors will erode away. For performing erosion, I used the method erode from the OpenCV library with a 5*5 kernel as the second argument

```
kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(image,kernel,iterations = 1)
```

6. Rotation: To rotate the image without scaling, I used the getRotationMatrix2d method from the OpenCV library to provide the center and angle of rotation.

```
M2 = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
```

The value returned by it is passed to the warpAffine method from the OpenCV library along with the number of rows and columns in the original image.

```
rotation = cv2.warpAffine(image,M2,(cols,rows))
```

1.4 Gaussian Pyramid: For creating the gaussian images for various sizes I used the method pyrDown from the OpenCV library. I created a set of images, each one being about 1/4th the area of the previous image.

```
A = cv2.pyrDown(image)
```

Then to pack the images together into a horizontal pyramid, I created a new image of height same as the original image and width almost 3/2nd of the original image. The area/size of this image is almost 3/2nd of the original image. Then I iterated through the set of images and copy the pixel values in the new image. The space requirement of the pyramid is almost 4/3 times the original image.

1.5 Computer Vision Application link: Augmented Reality App Ikea Place