Name : Aum M. Dhabalia **Date :** 28/07/2024

Roll No. : 21BEC027

Experiment…2 – Geometric Transformations

Objective : The objective of this lab is to introduce Geometric Transformation and apply it to images.

Import necessary libraries…

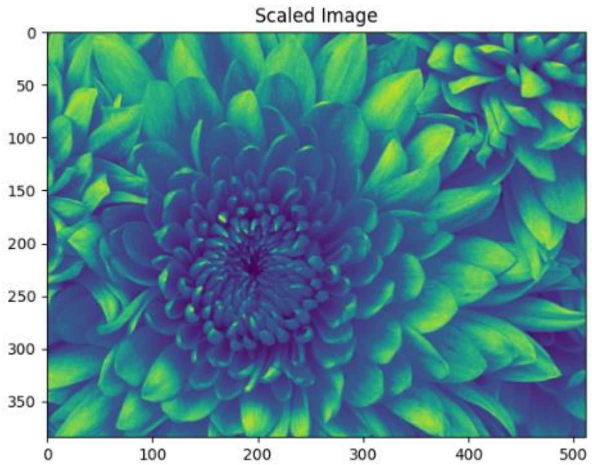
```
'''
Created on 28 July 2024 Mon 2:30 pm...
'''
import numpy as np
import cv2
import matplotlib.pyplot as plt

I = cv2.imread(r"D:\Nirma Files\Computer Vision\SampleImage.jpg")
Ig = cv2.cvtColor(cv2.cvtColor(I,cv2.COLOR_BGR2RGB),cv2.COLOR_RGB2GRAY)
x,y = Ig.shape
```

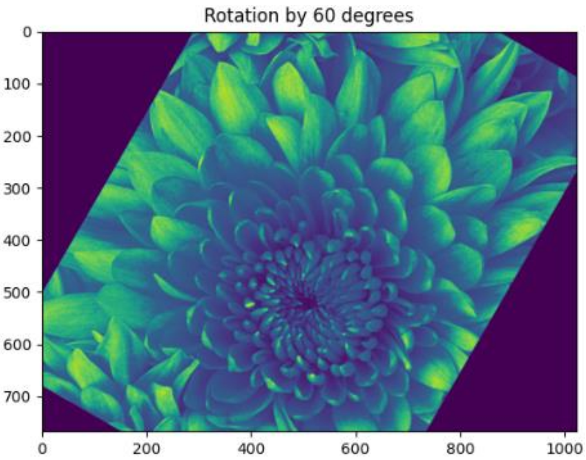
Task 1. Perform scaling, rotation and shifting operations on image using OpenCV.

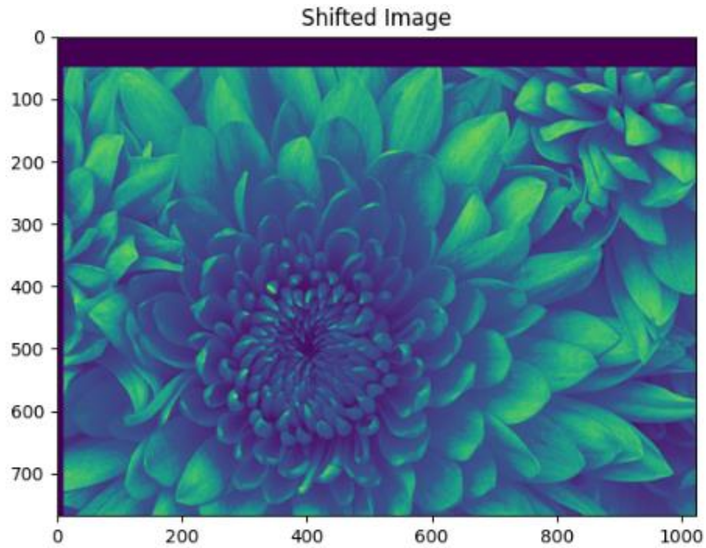
```
#Task 1
print("Image Scaling")
Iscale = cv2.resize(Ig,(0,0),fx = 0.5, fy = 0.5,interpolation = cv2.INTER_AREA)
plt.imshow(Iscale)
plt.title("Scaled Image")
plt.waitforbuttonpress()

print("Image rotation with warp")
R = cv2.getRotationMatrix2D((Ig.shape[1]/2,Ig.shape[0]/2),60,1)
Irotate = cv2.warpAffine(Ig,R,(Ig.shape[1],Ig.shape[0]));
plt.imshow(Irotate)
plt.title("Rotation by 60 degrees")
plt.waitforbuttonpress()

print("Image Shifting")
T = np.float32([[1,0,10],[0,1,50]]);
```

```
Ishift = cv2.warpAffine(Ig,T,(Ig.shape[1],Ig.shape[0]));
plt.imshow(Ishift)
plt.title("Shifted Image");
plt.waitforbuttonpress()
```

Output



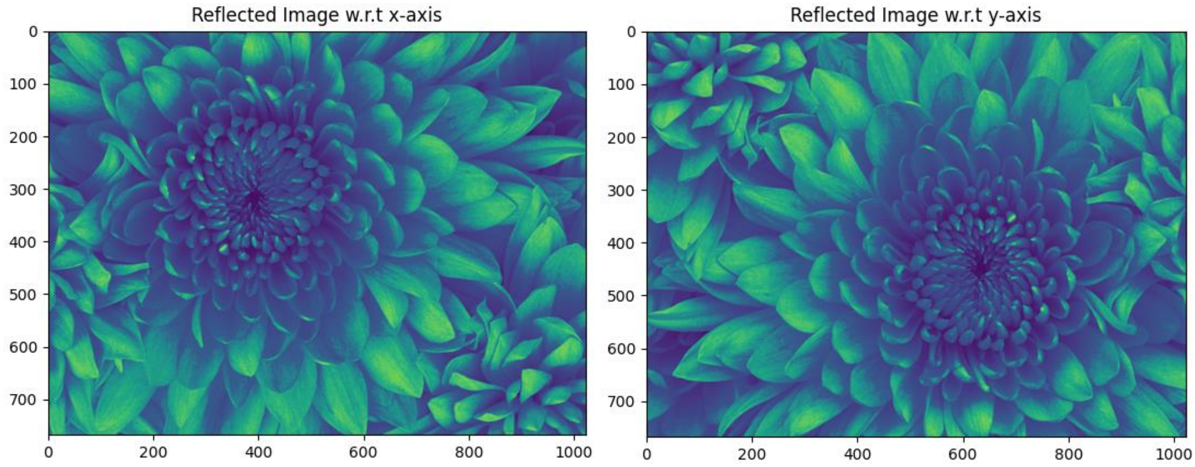




Task 2. Perform image reflection on an image using OpenCV.

```
#Task 2...
print("Reflection of image")
ref_matx = np.float32([[1,0,0],
                       [0,-1,x],
                       [0,0,1]])
reflectionbyX = cv2.warpPerspective(Ig,ref_matx,(int(y),int(x)))
plt.imshow(reflectionbyX)
plt.title("Reflected Image w.r.t x-axis")
plt.waitforbuttonpress()
ref_maty = np.float32([[-1,0,y],
                       [0,1,0],
                       [0,0,1]])
reflectionbyY = cv2.warpPerspective(Ig,ref_maty,(int(y),int(x)))
plt.imshow(reflectionbyY)
plt.title("Reflected Image w.r.t y-axis")
plt.waitforbuttonpress()
```

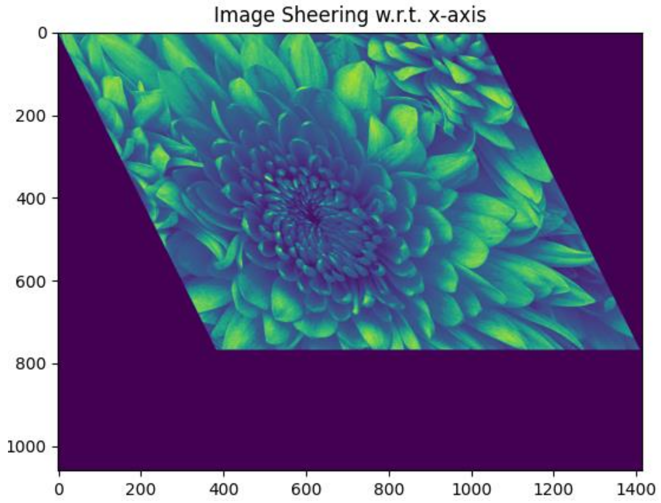
Output



Task 3. Perform image shearing on image using OpenCV.

```
print("Sheering of image")
sheer_mat = np.float32([[1,0.5,0],
                 [0,1,0],
                 [0,0,1]])
sheeredImage = cv2.warpPerspective(Ig,sheer_mat,(int(y*1.38),int(x*1.38)))
plt.imshow(sheeredImage)
plt.title("Image Sheering w.r.t. x-axis")
plt.waitforbuttonpress()
```

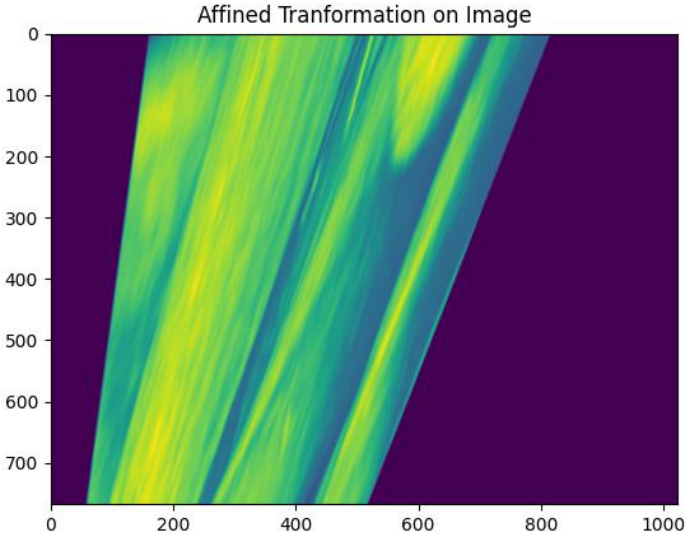
Output



Task 4. Apply affine transformation on an image iusing OpenCV.

```
print("Affined Transformation")
pts1 = np.float32([[90,90],[200,10],[100,20]])
pts2 = np.float32([[500,100],[200,10],[100,1050]])
affine_mat = cv2.getAffineTransform(pts1,pts2)
affineImage = cv2.warpAffine(Ig,affine_mat,(int(y),int(x)))
plt.imshow(affineImage)
plt.title("Affined Tranformation on Image")
plt.waitforbuttonpress()
```

Output



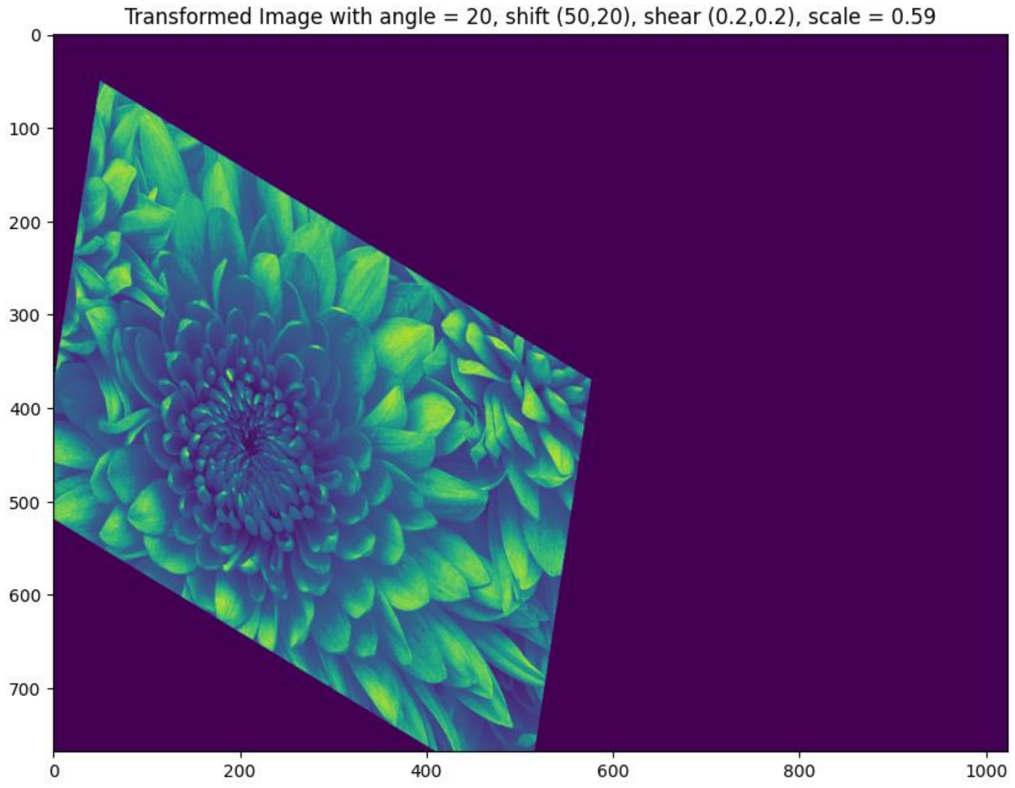
Task 5. Write a function for all geometric transformations and apply it to any image.

Task 6. Write affine transformation yourself

*****Task 5 and 6 are combined together as both has same code*****

```
def geometric_transform(img,theta=0,scale=1.0,trln=(0,0),shear=(0,0)):
    y,x= img.shape
    rotation = np.array([[np.cos(np.deg2rad(theta)), -np.sin(np.deg2rad(theta))],
                         [np.sin(np.deg2rad(theta)),  np.cos(np.deg2rad(theta))]])
    rotation *= scale
    shear_mat = np.array([[1,shear[0]],[shear[1],1]])
    affine_mat = np.hstack([np.dot(rotation,shear_mat),np.array([[trln[0]],[trln[1]]])])
    imageTransformed = cv2.warpAffine(img,affine_mat,(x,y))
    return imageTransformed


Igtransformed = geometric_transform(Ig,theta = 0,scale = 1,trln = (50,30),shear=(0.02,0.01))
plt.imshow(Igtransformed)
plt.title("Transformed Image")
plt.waitforbuttonpress()
```

Output



Observation:-

Different changes between original and processed images were observed.

- When scaling is applied on an image the size of the image is manipulated making image either big or small.
- When rotation matrix is applied on an image, it is observed that the image is rotated by an angle with respect to central pixel location of the image.
- When translation is applied, the image is shifted by certain pixels in either horizontal, vertical or both directions.
- When sheering is applied, it is observed that the opposites sides of the image maintain parallelism.
- A common transformation matrix called affine transformation matrix is created by combining rotation, scaling, sheeing and translation matrices. This transformation can transform image in combined rotation, scaling, sheering and translation.

Conclusion:-

As the experiment performed,

- the concept of geometric transformation was applied on image using OpenCV library in Python language.
- different transformation matrices like translation matrix, rotation matrix, shear matrix, scaling matrix, etc. were created for performing translation, rotation, sheering and scaling operations on an image.
- a universal transformation matrix called as affine transformation matrix was created by combining all four matrices i.e. translation, scaling, sheering and rotation.

Libraries and functions used are matplotlib, OpenCV, numpy, getAffineTransform(), getPerspectiveTransform(), warpAffine(), warpPerspective(), getRotationMatrix2D() etc.