**Lab-1-Report**

**Geometric Transforms**

****

**COURSE CODE: EE5175 (ISP)**

**Department of Electrical and Electronics**

**Engineering**

**INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**CHENNAI, TAMILNADU**

**Under the guidance of**

**Prof. Rajagopalan AN**

By

**EE21S048**

Ms – 2nd Semester

Date of submission: 06-02-2022

The following codes are executed and explained in google collab notebook (python) and link is <https://colab.research.google.com/drive/1kzXKh5r6b7D1Jpqs50fWIaZOWjBahycs?usp=sharing>

* **OpenCV** is a huge open-source library for computer vision, machine learning, and image processing.
* OpenCV library will allow to perform operations on Images. In our codes, we used opencv to read and display images.

**Q1. Translate the given image (lena translate.png) by (tx = 3.75, ty = 4.3) pixels.**

* Given Translation Parameters are: tx = 3.75, ty = 4.3
* Let (xs,ys) and (xt,yt) be source image co-ordinates and target image co-ordinates respectively.
* We have xt=xs+tx and yt=ys+ty translation.

**Bilinear Interpolation:**

Bilinear Interpolation considers four nearest neighbours in the source image and does distance weighted average to get a new pixel intensity value for target co-ordinates. The intensity of the four neighbours is weighted and averaged to the intensity value and is assigned to the target.

* img\_padded: The zero rows and columns are padded at the beginning and ending because for the first and last rows in performing bilinear transformation there should be intensity values for 4 nearest neighbours.
* The number of rows padded depends on n nearest neighbours interpolation.
* Here we chose bilinear transformation so number of nearest neighbours are 4.

**Python Code:**

from numpy import \*

import sys

import math

import cv2

import matplotlib.pyplot as plt

img1= cv2.imread("lena\_translate.png",0)#imread reads the images

img2= cv2.imread("pisa\_rotate.png",0)

img3= cv2.imread("cells\_scale.png",0)

tx=3.75

ty=4.3

width, height= img1.shape

print(width, height)

img1\_t=zeros((width, height))

img\_padded = zeros((width+2, height+2))

img\_padded[1:-1, 1:-1] = img1

for xt in range(width):

  for yt in range(height):

    xs=xt-tx

    ys=yt-ty

    #intensity=img1[xs,ys]

    x=xs+1

    y=ys+1

    xs1=math.floor(x)

    ys1=math.floor(y)

    a=x-xs1

    b=y-ys1

    if xs1>=0 and xs1<=width and ys1>=0 and ys1<=height:

      img1\_t[xt,yt]=(1-a)\*(1-b)\*img\_padded[xs1,ys1]+(1-a)\*b\*img\_padded[xs1,ys1+1]+a\*(1-b)\*img\_padded[xs1+1,ys1]+a\*b\*img\_padded[xs1+1,ys1+1]

    else:

      img1\_t[xt,yt]=125

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9,12), constrained\_layout=True)

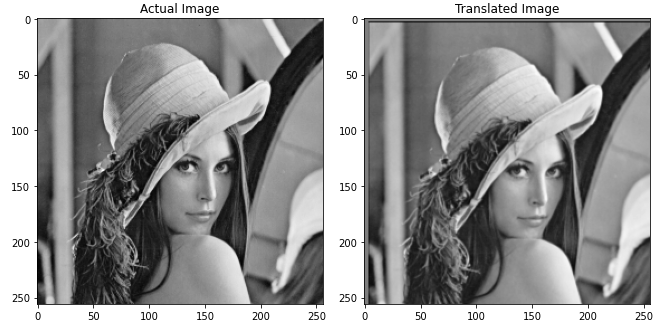
ax1.imshow(img1,'gray')

ax1.title.set\_text("Actual Image")

ax2.imshow(img1\_t,'gray')

ax2.title.set\_text("Translated Image")

**Result:**



**Conclusion:**

* We can observe grey borders at the edge of image because intensity value 125 is assigned to the values where (xs,ys) doesn’t exist.
* We can see that translated image is not clear compared to actual image because of averaging of intensity values in bilinear interpolation.
* Thus the Lena image(img1) is translated by tx=3.75 and ty=4.3.

**Q2. Rotate the given image (pisa rotate.png) about the image centre, so as to straighten the Pisa tower.**

* Given parameters are:  , by trail and error lets check for its value.
* Let (xs,ys) and (xt,yt) be source image co-ordinates and target image co-ordinates respectively.
* We have and rotation.
* We have to rotate the image about the image center (width2/2, height2/2).
* To rotate around a point, we have to translate to that point and then apply rotation and then translate back

**Python Code:**

#let theta=10',5',4'

theta=-5

theta\_r=pi\*theta/180

width2, height2= img2.shape

print(width2, height2)

img\_padded2 = zeros((width2+2, height2+2))

img\_padded2[1:-1, 1:-1] = img2

img2\_r=zeros((width2, height2))

for xt in range(width2):

  for yt in range(height2):

      xc, yc = xt-width2/2, yt-height2/2

      xs = cos(theta\_r)\*xc - sin(theta\_r)\*yc + width2/2

      ys = cos(theta\_r)\*yc + sin(theta\_r)\*xc + height2/2

      x=xs+1

      y=ys+1

      xs1=math.floor(x)

      ys1=math.floor(y)

      a=x-xs1

      b=y-ys1

      if xs1>=0 and xs1<=width2 and ys1>=0 and ys1<=height2:

        img2\_r[xt,yt]=(1-a)\*(1-b)\*img\_padded2[xs1,ys1]+(1-a)\*b\*img\_padded2[xs1,ys1+1]+a\*(1-b)\*img\_padded2[xs1+1,ys1]+a\*b\*img\_padded2[xs1+1,ys1+1]

      else:

        img2\_r[xt,yt]=125

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9,12), constrained\_layout=True)

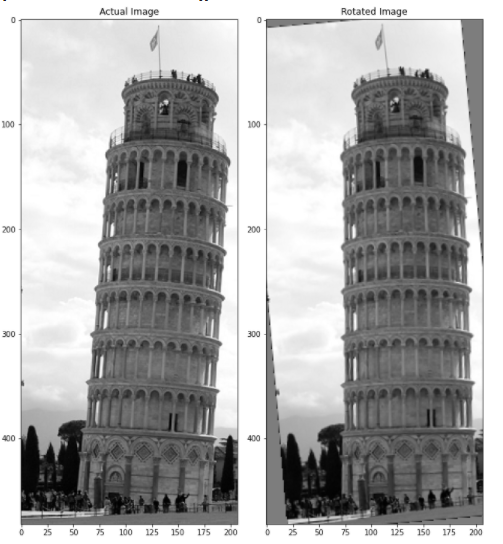
ax1.imshow(img2,'gray')

ax1.title.set\_text("Actual Image")

ax2.imshow(img2\_r,'gray')

ax1.title.set\_text("Rotated Image")

**Result:**



**Conclusion:**

* By the trail and error values for rotation angle(θ ) we can see straight tower at an angle of -5’.
* For the clock wise rotation, the angle should be positive as pisa tower has to be rotated counter clk-wise, the angle supplied is negative.
* The Pisa tower image is rotated in anti-clock wise direction by θ = -5 degrees.

**Q3. Scale the given image (cells scale.png) by 0.8 and 1.3 factors.**

* Given scaling parameters are: , .
* Let (xs,ys) and (xt,yt) be source image co-ordinates and target image co-ordinates respectively.
* We have and scaling.
* We have to scale the image about the image center (width2/2, height2/2).
* To scale around a point, we have to translate to that point and then apply scaling and then translate back.

**Python Code:**

a1=0.8

#a=1.3

width3, height3= img3.shape

print(width3, height3)

img\_padded3 = zeros((width3+2, height3+2))

img\_padded3[1:-1, 1:-1] = img3

img3\_s=zeros((width3, height3))

xc=width3/2

yc=height3/2

for xs in range(width3):

  for ys in range(height3):

    xt = (xs-xc)/a1 + xc

    yt = (ys-yc)/a1+yc

    x=xt+1

    y=yt+1

    xs1=math.floor(x)

    ys1=math.floor(y)

    a=x-xs1

    b=y-ys1

    if xs1>=0 and xs1<=width3 and ys1>=0 and ys1<=height3:

      img3\_s[xt,yt]=(1-a)\*(1-b)\*img\_padded3[xs1,ys1]+(1-a)\*b\*img\_padded3[xs1,ys1+1]+a\*(1-b)\*img\_padded3[xs1+1,ys1]+a\*b\*img\_padded3[xs1+1,ys1+1]

    else:

      img3\_s[xt,yt]=0

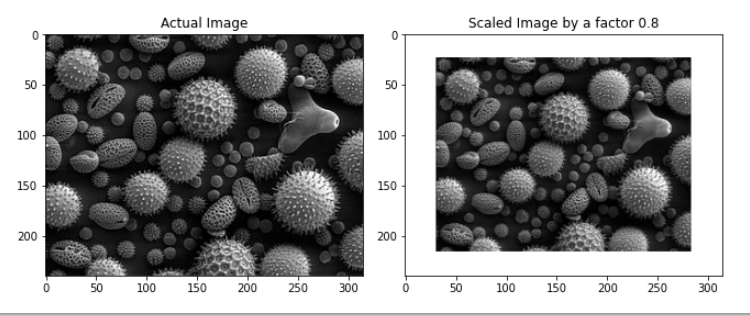
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9,12), constrained\_layout=True)

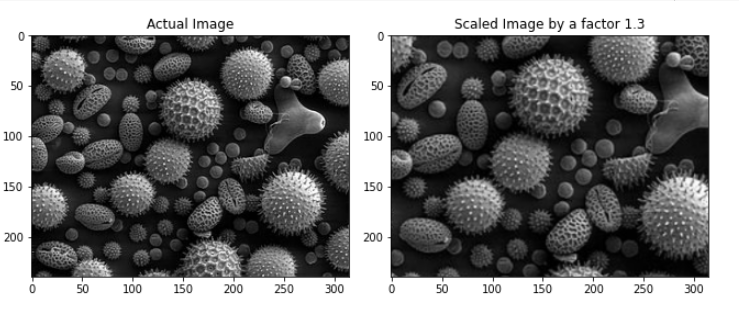
ax1.imshow(img3,'gray')

ax1.title.set\_text("Actual Image")

ax2.imshow(img3\_s,'gray')

ax2.title.set\_text("Scaled Image by a factor 0.8")





**Conclusion:**

* From the result, we can infer that a=0.8 implies zoom out operation and a=1.3 implies zoom in operation.
* In scaling operation there will be translation along z axis(optical axis).
* we have a=(1+tx/d) for the zoom in operation d decreases and as a result we have a>1 and for the zoom out operation d increases and a<1.
* Therefore we applied uniform scaling on the cells image with scaling factors 0.8,1.3.