EE5175 -Image Signal Processing

Lab-1

Geometric Transforms

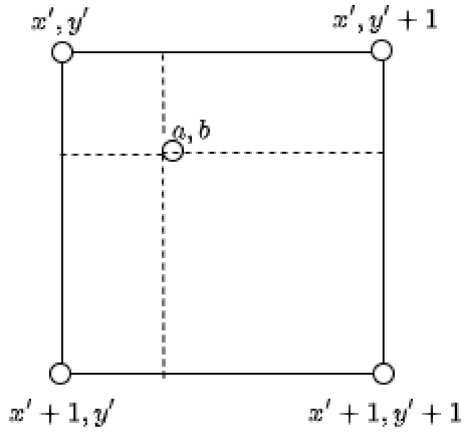
Import Libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread
import math as m
```

In [2]:
```
img_lena = imread('lena_translate.png')
img_pisa = imread('pisa_rotate.png')
img_cell = imread('cells_scale.png')
```

Bilinear Interpolation

$I_t = (1-a) * (1-b)\, I_s(x', y') + (1-a) * (b)\, I_s(x', y'+1) + (a) * (1-b)\, I_s(x'+1, y') + (a) * (b)\, I_s(x'+1, y'+1)$



In [3]:
```
def bilinear_interploation(source , x , y ):
    '''
    input params : source =  source image
                   x =  x coordinate
                   y = y coordinate

        output : returns a target image
    '''

    xx = source.shape[0]   # shape of the input image
    yy = source.shape[1]
```

```
    x_dash = m.floor(x)    #  floor value of the coordinate (x ' and y' as mentione
    y_dash = m.floor(y)

    a = x - x_dash       # finding a and b i.e difference between actual and the fl
    b = y - y_dash

    if x_dash >= 0 and y_dash>=0 and x_dash <= xx-2 and y_dash <= yy-2:  ##formula
        target_img =(1-a)*(1-b)*source[x_dash,y_dash]+ (1-a)*b*source[x_dash,y_dash

    else:
        target_img = 0     # without using this condition , zeros wouldn't be creat

    return target_img
```

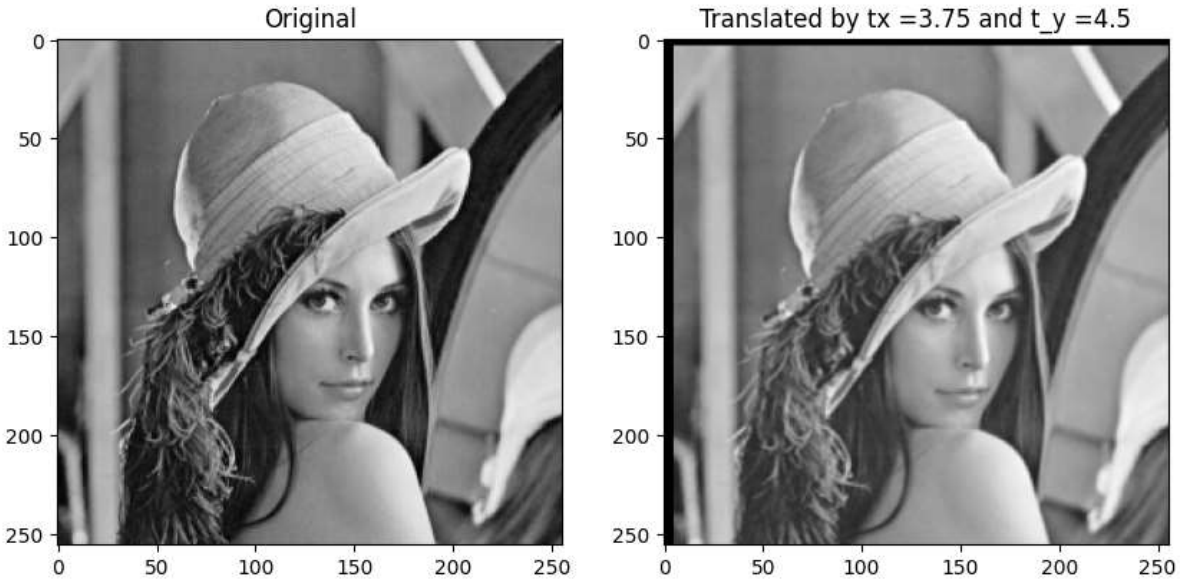
Translation

In [4]:
```
tx = 3.75  #given values
ty = 4.5

target_ = np.zeros_like(img_lena)   #create a image of zeros of the same dimension
x_lena = img_lena.shape[0]
y_lena = img_lena.shape[1]

for x in range(x_lena):
    for y in range(y_lena):
        xs = x - tx  # translation matrix equations
        ys = y - ty
#         print(xs,ys)
        target_[x,y] = bilinear_interploation(img_lena , xs , ys)
```

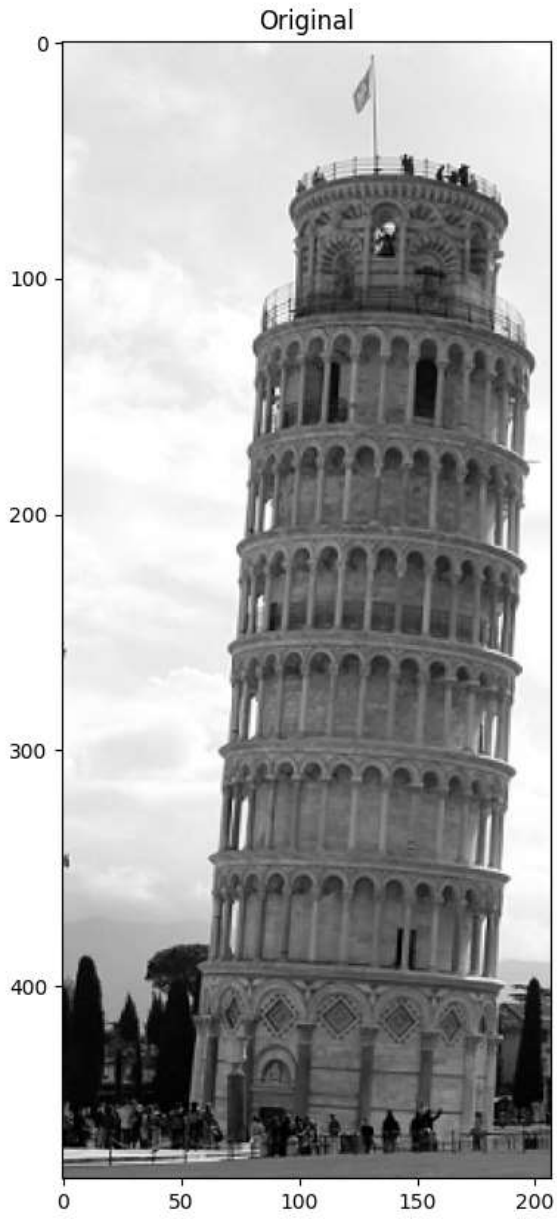
In [17]:
```
plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.imshow(img_lena , 'gray')
plt.title('Original')
plt.subplot(1,2,2)
plt.imshow(target_ , 'gray')
plt.title("Translated by tx =3.75 and t_y =4.5")
plt.show()
```

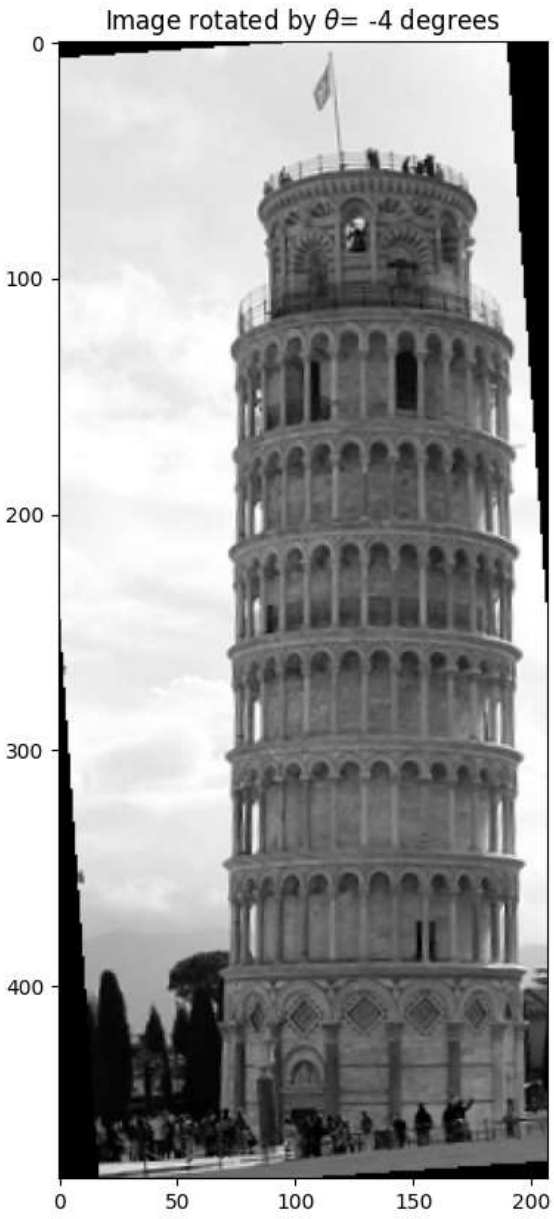


Rotation

In [32]:
```
target_ = np.zeros_like(img_pisa) # target image zeros creation
xx = img_pisa.shape[0]
yy = img_pisa.shape[1]
# print(xx)
x_center , y_center = xx//2 ,yy//2
# print(x_center , y_center)
theta = -4*np.pi/180
for x in range(xx):
    for y in range(yy):
        x_c, y_c = x-x_center, y-y_center
        # Then we apply rotation as we would apply rotation around the origin
        # And then translate back
        xs = np.cos(theta)*x_c - np.sin(theta)*y_c + x_center
        ys = np.cos(theta)*y_c + np.sin(theta)*x_c + y_center
        target_[x,y] = bilinear_interploation(img_pisa, xs, ys)
```

In [33]:
```
plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.imshow(img_pisa , 'gray')
plt.title('Original')
plt.subplot(1,2,2)
plt.imshow(target_ , 'gray')
plt.title(fr"Image rotated by $\theta$= -4 degrees")
plt.show()
```





Scaling

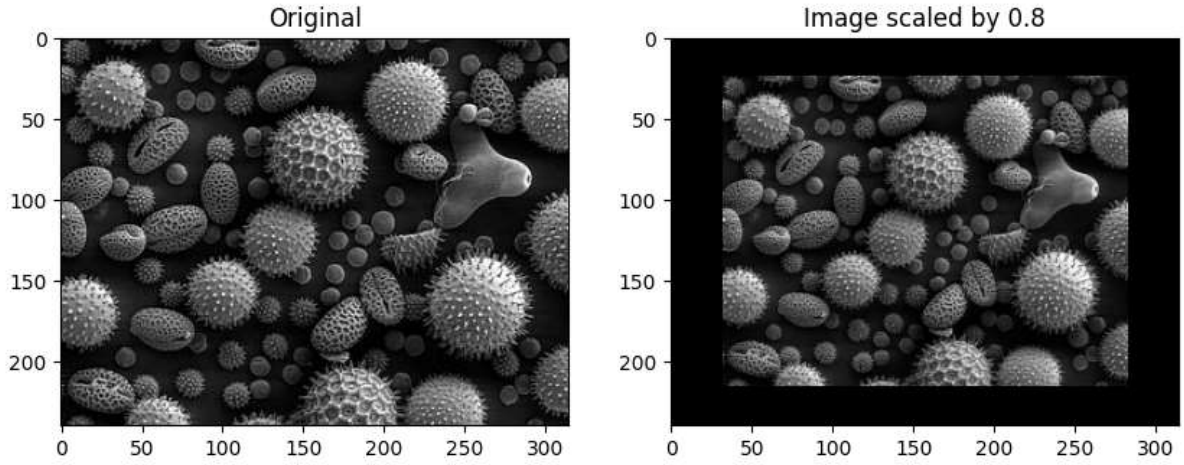
Zoom-out (scale factor less than one)

In [22]:
```
target_ = np.zeros_like(img_cell)
xx = img_cell.shape[0]
yy = img_cell.shape[1]
# print(xx)
x_center , y_center = xx//2 ,yy//2
print(x_center , y_center)
scale = 0.8
for x in range(xx):
    for y in range(yy):
        #subtracting from the center
        # subtract and translate back
        xs = (x-x_center)/scale+x_center
        ys = (y-y_center)/scale+y_center
```

```
        target_[x,y] = bilinear_interploation(img_cell, xs, ys)


plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.imshow(img_cell , 'gray')
plt.title('Original')
plt.subplot(1,2,2)
plt.imshow(target_ , 'gray')
plt.title(fr"Image scaled by 0.8")
plt.show()
```

```
120 157
```

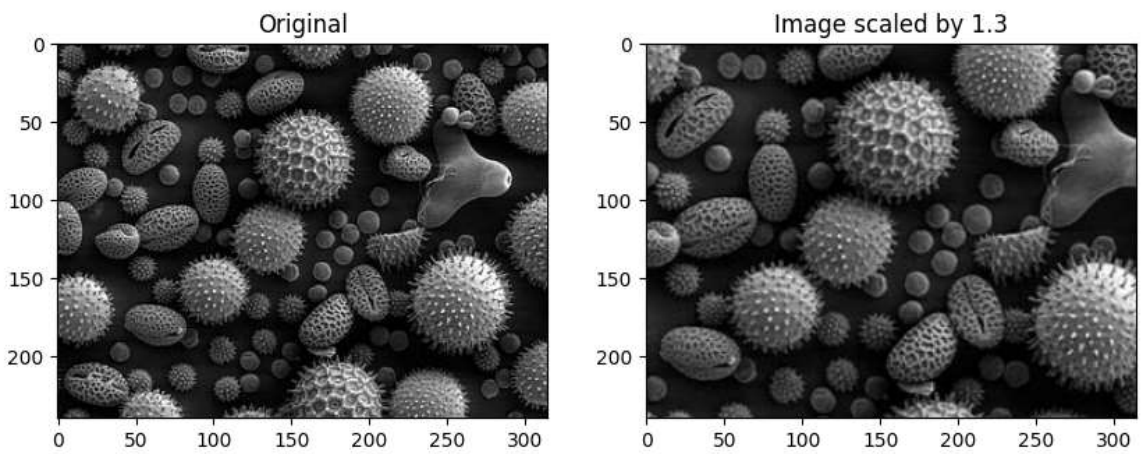


Zoom-In (scale factor greater than one)

In [23]:
```
target_ = np.zeros_like(img_cell)
xx = img_cell.shape[0]
yy = img_cell.shape[1]
# print(xx)
x_center , y_center = xx//2 ,yy//2
print(x_center , y_center)
scale = 1.3
for x in range(xx):
    for y in range(yy):
        x_c, y_c = x-x_center, y-y_center
        xs = (x-x_center)/scale+x_center
        ys = (y-y_center)/scale+y_center
        target_[x,y] = bilinear_interploation(img_cell, xs, ys)


plt.figure(figsize=(10,10))
plt.subplot(1,2,1)
plt.imshow(img_cell , 'gray')
plt.title('Original')
plt.subplot(1,2,2)
plt.imshow(target_ , 'gray')
plt.title("Image scaled by 1.3")
plt.show()
```

```
120 157
```



Observations

- Due to the interpolation the quality of the image is lost
- No Holes found in the image due to Bilinear Interpolation
- Rotation and Translation performed from the center of the image