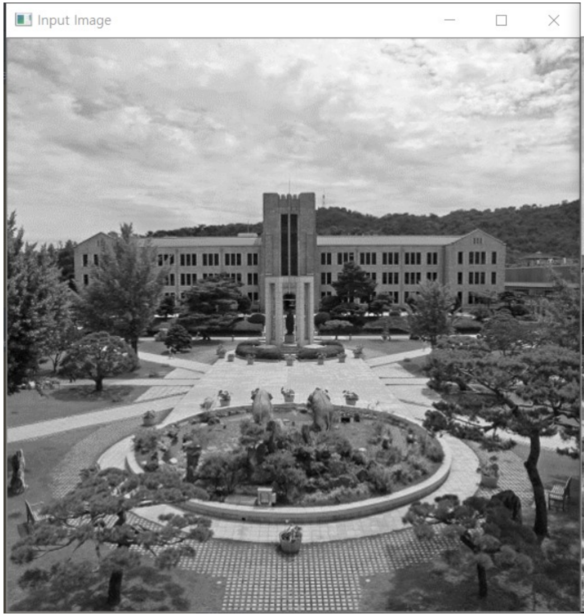
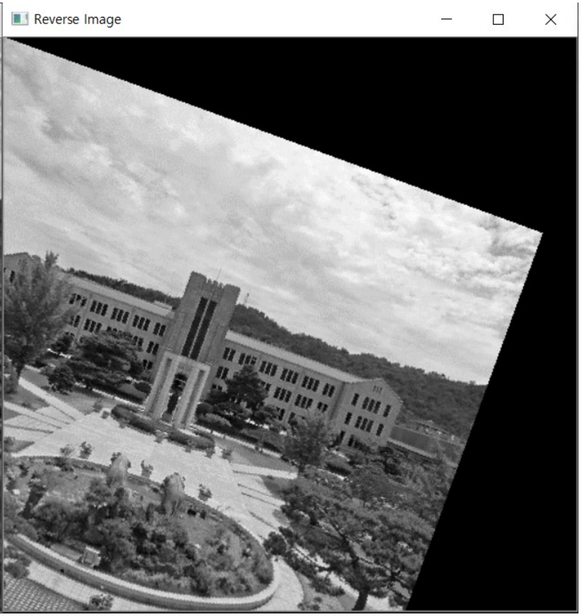
```python
import cv2
import numpy as np

def Forward_transform(img, angle):  # forward transformation
    height, width = img.shape
    result = np.zeros((height, width), np.uint8)  # result image

    affine = np.array([[np.cos(np.radians(angle)), -np.sin(np.radians(angle)), 0],
                       [np.sin(np.radians(angle)), np.cos(np.radians(angle)), 0],
                       [0, 0, 1]])  # Affine transformation matrix

    for x in range(width):
        for y in range(height):
            p = affine.dot(np.array([x, y, 1])) #rotate x,y

            xp = int(p[0])
            yp = int(p[1])

            if 0 <= yp < height and 0 <= xp < width:
                result[yp, xp] = img[y, x]
    return result

def Reverse_transform(img, angle):  # forward transformation
    height, width = img.shape
    result = np.zeros((height, width), np.uint8)  # result image

    affine = np.array([[np.cos(np.radians(angle)), np.sin(np.radians(angle)), 0],
                       [-np.sin(np.radians(angle)), np.cos(np.radians(angle)), 0],
                       [0, 0, 1]])  # Affine transformation matrix

    for x in range(width):
        for y in range(height):
            p = affine.dot(np.array([x, y, 1])) #rotate x,y

            # □□ □□□□ □□
            xp = int(p[0])
            yp = int(p[1])

            if 0 <= yp < height and 0 <= xp < width:
                result[y, x] = img[yp, xp]
    return result


in_image = cv2.imread('dgu_gray.png', 0)  # img2numpy

out_image1 = Forward_transform(in_image, 20)
out_image2 = Reverse_transform(in_image, 20)

cv2.imshow('Input Image', in_image)
cv2.imshow('Forward Image', out_image1)
cv2.imshow('Reverse Image', out_image2)

cv2.imwrite('dgu_gray_rotate1.png', out_image1)  # save result img
cv2.imwrite('dgu_gray_rotate2.png', out_image2)  # save result img
cv2.waitKey()

```


Input Image


Reverse Image