

```

1 import cv2
2 import numpy as np
3
4 def Forward_transform(img, angle): # forward transformation
5     height, width = img.shape
6     result = np.zeros((height, width), np.uint8) # result image
7
8     affine = np.array([[np.cos(np.radians(angle)), -np.sin(np.radians(angle)), 0],
9                        [np.sin(np.radians(angle)), np.cos(np.radians(angle)), 0],
10                       [0, 0, 1]]) # Affine transformation matrix
11
12     for x in range(width):
13         for y in range(height):
14             p = affine.dot(np.array([x, y, 1])) #rotate x,y
15
16             xp = int(p[0])
17             yp = int(p[1])
18
19             if 0 <= yp < height and 0 <= xp < width:
20                 result[yp, xp] = img[y, x]
21     return result
22
23 def Reverse_transform(img, angle): # forward transformation
24     height, width = img.shape
25     result = np.zeros((height, width), np.uint8) # result image
26
27     affine = np.array([[np.cos(np.radians(angle)), np.sin(np.radians(angle)), 0],
28                        [-np.sin(np.radians(angle)), np.cos(np.radians(angle)), 0],
29                       [0, 0, 1]]) # Affine transformation matrix
30
31     for x in range(width):
32         for y in range(height):
33             p = affine.dot(np.array([x, y, 1])) #rotate x,y
34
35             #
36             xp = int(p[0])
37             yp = int(p[1])
38
39             if 0 <= yp < height and 0 <= xp < width:
40                 result[y, x] = img[yp, xp]
41     return result
42
43
44 in_image = cv2.imread('dgu_gray.png', 0) # img2numpy
45
46 out_image1 = Forward_transform(in_image, 20)
47 out_image2 = Reverse_transform(in_image, 20)
48
49 cv2.imshow('Input Image', in_image)
50 cv2.imshow('Forward Image', out_image1)
51 cv2.imshow('Reverse Image', out_image2)
52
53 cv2.imwrite('dgu_gray_rotate1.png', out_image1) # save result img
54 cv2.imwrite('dgu_gray_rotate2.png', out_image2) # save result img
55 cv2.waitKey()
56

```

