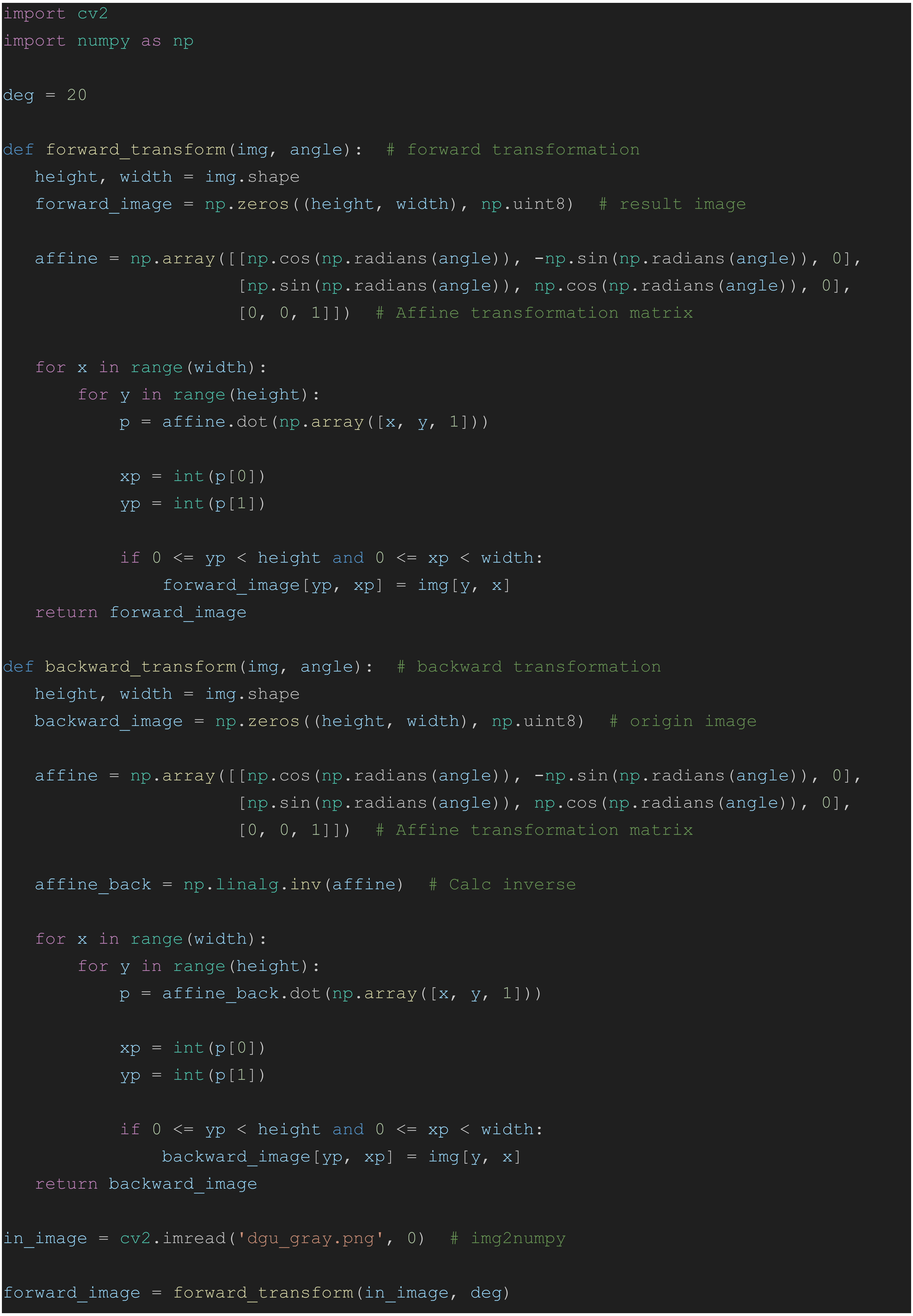

```python
import cv2
import numpy as np

deg = 20

def forward_transform(img, angle):  # forward transformation
    height, width = img.shape
    forward_image = 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]))

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

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

def backward_transform(img, angle):  # backward transformation
    height, width = img.shape
    backward_image = np.zeros((height, width), np.uint8)  # origin 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

    affine_back = np.linalg.inv(affine)  # Calc inverse

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

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

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

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

forward_image = forward_transform(in_image, deg)
```

```
backward_image = backward_transform(forward_image, deg)

stack_image = np.vstack([forward_image,backward_image])
cv2.imshow('Stack Image', stack_image)

cv2.imwrite('dgu_gray_stack.png', stack_image)

cv2.waitKey()
```