

riruz0o8m

December 3, 2024

0.1 Lab Exercise 5: Geometric Transformations and Affine Transformations

- **Objective:** Apply geometric transformations and affine transformations to images.
- **Task:** Perform image scaling, rotation, translation, and affine transformations. Visualize the effect of each transformation on an image.

0.2 Image Scaling

```
[1]: import numpy as np
import cv2
from google.colab.patches import cv2_imshow
```

```
[2]: # Reading the original image before resizing
img = cv2.imread('photography.jpg')
resized_img = cv2.resize(img, (300, 300))
# original dimensions of the image
height, width = img.shape[:2]
height,width
```

```
[2]: (1500, 1000)
```

```
[3]: # Displaying the orginial Image
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
[4]: # Resizing the image
resized_img = cv2.resize(img, (300, 300))
# Dimensions after resizing
# original dimensions of the image
height, width = resized_img.shape[:2]
height,width
```

[4]: (300, 300)

```
[5]: # Save the resized image
cv2.imwrite("Photograph Resized.jpg", resized_img)
img_resized = cv2.imread('Photograph Resized.jpg',0)
```

```
[6]: # displaying the image
cv2_imshow(img_resized)
```



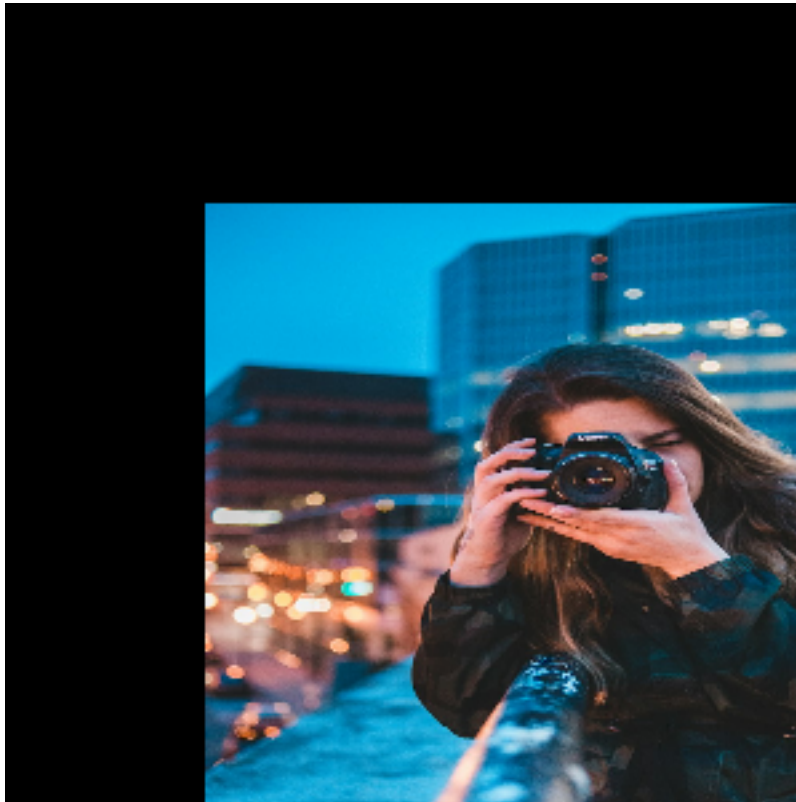
0.3 Image Translation

```
[7]: # get tx and ty values for translation
# you can specify any value of your choice
tx, ty = width / 4, height / 4

# create the translation matrix using tx and ty, it is a NumPy array
translation_matrix = np.array([
    [1, 0, tx],
    [0, 1, ty]
], dtype=np.float32)

[8]: # apply the translation to the image
translated_image = cv2.warpAffine(src=resized_img, M=translation_matrix,
    ↳dsize=(width, height))

[9]: # display the original and the Translated images
cv2.imshow( translated_image)
cv2.waitKey(0)
# save the translated image to disk
cv2.imwrite('translated_image.jpg', translated_image)
```




```
[9]: True
```

0.4 Image Rotation

```
[10]: # Reading the image  
resized_img = cv2.imread('Photograph Resized.jpg')
```

```
[11]: # Displaying the image  
cv2_imshow(resized_img)
```



```
[12]: # Dividing height and width by 2 to get the center of the image  
height, width = resized_img.shape[:2]  
center = (width/2, height/2)
```

```
[13]: # the above center is the center of rotation axis  
# use cv2.getRotationMatrix2D() to get the rotation matrix  
rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=45, scale=1)
```

```
[14]: # Rotate the image using cv2.warpAffine  
rotated_image = cv2.warpAffine(src=resized_img, M=rotate_matrix, dsize=(width, height))
```

```
[15]: # visualize the original and the rotated image  
cv2.imshow(rotated_image)
```



```
[16]: # write the output, the rotated image to disk  
cv2.imwrite('rotated_image.jpg', rotated_image)
```

```
[16]: True
```

```
[16]:
```