Digital Image Processing
Spring 2025
Lab:3 Affine Transformation

Name	
Student ID	
Date	

Objectives

- Perform affine transformations on images such as translation, rotation, scaling, and shearing.
- Understand how affine transformations preserve the relationships between points, straight lines, and planes while allowing for changes in angles and distances.

Prerequisites

Make sure you have the following Python libraries installed:

- `opencv-python` (OpenCV)
- `matplotlib`
- `numpy`

You can install them using following command in command prompt or bash:

pip install opencv-python, matplotlib, numpy

Digital Image Processing
Spring 2025
Lab:3 Affine Transformation

Part 1: Affine Transformations

Affine transformations include linear operations like translation, rotation, scaling, and shearing. They preserve straight lines and parallelism in an image but allow changes in size and orientation.

Step 1: Translation (Shifting an Image)

Translation shifts an image by adjusting pixel coordinates. The transformation matrix for translation according to the textbook is:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Opencv uses a similar 2×3 matrix that is provided below

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read an image
image = cv2.imread('lena.png')
print(image.shape)
# Define a translation matrix (shift 100 pixels
right, 50 pixels down)
translation_matrix = np.float32([[1, 0, 100], [0,
1, 50]])

# Apply the translation matrix to the image
translated_image = cv2.warpAffine(image,
translation_matrix, (image.shape[1],
image.shape[0]))
print(translated_image.shape)
```

```
# Display the translated image
plt.imshow(cv2.cvtColor(translated_image,
cv2.COLOR_BGR2RGB))
plt.title('Translated Image')
plt.axis('off')
plt.show()
```

Task 1:

1. Experiment with different translations.
2. Try negative shifts (moving the image left or up).

Step 2: Rotation

Rotation rotates the image around a point (commonly the center). The rotation matrix is:

$$T = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ Sin\theta & cos\theta & 0 \end{bmatrix}$$

```
# rotation
(h, w) = image.shape[:2]
center = (w // 2, h // 2)

# Define a rotation matrix to rotate by 45 degrees
rotation_matrix = cv2.getRotationMatrix2D(center,
45, 1.0)

# Apply the rotation to the image
rotated_image = cv2.warpAffine(image,
rotation_matrix, (w, h))

# Display the rotated image
plt.imshow(cv2.cvtColor(rotated_image,
cv2.COLOR_BGR2RGB))
plt.title('Rotated Image (45 degrees)')
plt.axis('off')
plt.show()
```

Task 2:

1. Rotate the image by different angles (e.g., 90°, 180°).
2. Explore rotation with scaling (e.g., change the scale factor in the rotation matrix).

Step 3: Scaling

Scaling alters the size of an image by stretching or shrinking it. The transformation matrix for scaling is:

$$T = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \end{bmatrix}$$

where s_x and s_y are the scaling factors along the x and y axes.

```
# scalling
# Define a scaling matrix (double the size of the
image)
Factor = 2
scaling_matrix = cv2.getRotationMatrix2D((0, 0), 0,
Factor)

# Apply the scaling transformation
scaled_image = cv2.warpAffine(image,
scaling_matrix, (w*Factor, h*Factor))

# Display the scaled image
plt.imshow(cv2.cvtColor(scaled_image,
cv2.COLOR_BGR2RGB))
plt.title('Scaled Image (2x)')
plt.axis('off')
plt.show()
print(scaled_image.shape)
```

Task 3:

1. Scale the image by different factors (e.g., 0.5, 1.5).
2. Discuss the impact of extreme scaling on image quality (try using big scaling factors such as 5, 8, 10, 20, save the image to get the answer).

Step 4: Shearing

Shearing tilts the image in a specified direction. The transformation matrix for shearing along the x-axis is:

$$T = \begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Where m is the shearing factor

```
# Shear
m = 0.67
shearing_matrix = np.float32([[1, m, 0], [0, 1,
0]])

# Apply the shearing transformation
sheared_image = cv2.warpAffine(image,
shearing_matrix, (w, h))

# Display the sheared image
plt.imshow(cv2.cvtColor(sheared_image,
cv2.COLOR_BGR2RGB))
plt.title('Sheared Image (x-axis, factor ' + str(m)
+')')
plt.axis('off')
plt.show()
lt.show()
```

Task 4:

1. Try shearing the image along the x and y axes using different factors.
2. Discuss how shearing distorts the image and affects its proportions.

Submission

Submit the following:

1. Screenshots of the images after performing each transformation.
2. The code for each task.
3. A brief report explaining the purpose of affine transformations and what you learned from each operation.

Conclusion

In this lab, you explored affine transformations such as translation, rotation, scaling, and shearing. These transformations are fundamental for manipulating images geometrically and are key to understand some of the dataset augmentation techniques used in Deep Learning.