**Lab 04:** Image Transformations and Warping using Homogeneous Coordinates

**Lab Outcomes:**

1. Understand homogeneous coordinates and their role in image transformations.
2. Implement affine transformations (translation, rotation, scaling, shearing) using Python.
3. Apply perspective transformations and image warping techniques.
4. Develop a deep understanding of how transformation matrices affect images.
5. Solve real-world problems involving image manipulation and geometric transformations.

**Introduction and Theoretical Concepts**

**Homogeneous Coordinates**

Homogeneous coordinates extend traditional 2D coordinates by adding an extra dimension, making it easier to represent transformations as matrix multiplications. A 2D point is represented in homogeneous coordinates as . Transformations such as translation, rotation, scaling, and shearing can be expressed as matrix operations:

where is a 3x3 transformation matrix.

**Affine and Perspective Transformations**

* **Affine Transformations:** Preserve lines and parallelism, including translation, rotation, scaling, and shearing.
* **Perspective Transformations:** Modify the perspective of an image, simulating a 3D viewpoint.

**Image Warping**

Image warping involves mapping an image from one shape to another using interpolation techniques. It is widely used in computer vision applications such as face alignment, panoramic stitching, and augmented reality.

## ****Solved Activities****

### ****Activity 1: Applying Translation to an Image****

**Instructions:**

1. Load an image using OpenCV.
2. Define a translation matrix and apply it.
3. Display the original and transformed image.

import cv2

import numpy as np

import matplotlib.pyplot as plt

image = cv2.imread('image.jpg')

h, w = image.shape[:2]

T = np.array([[1, 0, 50], [0, 1, 30], [0, 0, 1]], dtype=np.float32)

translated\_image = cv2.warpPerspective(image, T, (w, h))

plt.imshow(cv2.cvtColor(translated\_image, cv2.COLOR\_BGR2RGB))

plt.title("Translated Image")

plt.show()

### ****Activity 2: Rotation of an Image using Homogeneous Coordinates****

**Instructions:**

1. Define a rotational transformation matrix in homogeneous coordinates.
2. Apply it using OpenCV.

angle = np.radians(45)

cos\_theta, sin\_theta = np.cos(angle), np.sin(angle)

# Define the homogeneous transformation matrix for rotation

M = np.array([[cos\_theta, -sin\_theta, w//2 - (cos\_theta \* w//2 - sin\_theta \* h//2)],

[sin\_theta, cos\_theta, h//2 - (sin\_theta \* w//2 + cos\_theta \* h//2)],

[0, 0, 1]], dtype=np.float32)

rotated\_image = cv2.warpPerspective(image, M, (w, h))

plt.imshow(cv2.cvtColor(rotated\_image, cv2.COLOR\_BGR2RGB))

plt.title("Rotated Image using Homogeneous Coordinates")

plt.show()

**Activity 3: Scaling an Image**

scale\_x, scale\_y = 1.5, 1.5

M = np.array([[scale\_x, 0, 0], [0, scale\_y, 0]], dtype=np.float32)

scaled\_image = cv2.warpAffine(image, M, (int(w \* scale\_x), int(h \* scale\_y)))

plt.imshow(cv2.cvtColor(scaled\_image, cv2.COLOR\_BGR2RGB))

plt.title("Scaled Image")

plt.show()

**Activity 4: Shearing an Image**

shear\_x, shear\_y = 0.2, 0.3

M = np.array([[1, shear\_x, 0], [shear\_y, 1, 0]], dtype=np.float32)

sheared\_image = cv2.warpAffine(image, M, (w, h))

plt.imshow(cv2.cvtColor(sheared\_image, cv2.COLOR\_BGR2RGB))

plt.title("Sheared Image")

plt.show()

**Activity 5: Perspective Transformation**

pts1 = np.float32([[50, 50], [200, 50], [50, 200], [200, 200]])

pts2 = np.float32([[10, 100], [180, 50], [50, 250], [220, 220]])

M = cv2.getPerspectiveTransform(pts1, pts2)

perspective\_image = cv2.warpPerspective(image, M, (w, h))

plt.imshow(cv2.cvtColor(perspective\_image, cv2.COLOR\_BGR2RGB))

plt.title("Perspective Warped Image")

plt.show()

### ****Activity 6: Combining Rotation, Translation, and Scaling****

# Get image dimensions

h, w = image.shape[:2]  # Extract height and width

# Transformation parameters

angle = np.radians(30)

scale\_x, scale\_y = 1.2, 1.2

translate\_x, translate\_y = 40, 30

# Homogeneous transformation matrix

M = np.array([[scale\_x \* np.cos(angle), -np.sin(angle), translate\_x],

              [np.sin(angle), scale\_y \* np.cos(angle), translate\_y],

              [0, 0, 1]], dtype=np.float32)

# Apply transformation

transformed\_image = cv2.warpPerspective(image, M, (w, h))

# Display the result

plt.imshow(transformed\_image, cmap='gray')

plt.title("Rotated, Translated, and Scaled Image")

plt.axis("off")

plt.show()

**Graded Tasks**

**Task 1: Implement Translation and Rotation for a Given Image**

Scenario: A user wants to shift and rotate a scanned document image to align it properly.

**Task 2: Resize an Image While Maintaining Aspect Ratio**

Scenario: Resize a user-uploaded image to fit within a 500x500 px frame without distortion.

**Task 3: Implement Horizontal and Vertical Flipping**

Scenario: Mirror an image for creating reflection effects in a photo editing tool.

**Task 4: Create a Perspective Transformation for Augmented Reality**

Scenario: Transform an advertisement board in a street image to show a custom ad.

**Task 5: Design a Warp Transformation for Face Alignment**

Scenario: Align a face in an image for a facial recognition system.

**Task 6: Apply Shearing to Simulate a 3D Effect**

Scenario: Create a slanted text effect for an artistic poster.

**Task 7: Implement Image Warping for Panorama Stitching**

Scenario: Align two overlapping images to create a panorama.

**Task 8: Develop a Function to Apply Multiple Transformations Sequentially**

Scenario: Simulate a complex photo filter applying multiple transformations.

**Task 9: Implement Image Cropping and Rescaling**

Scenario: Crop a region of interest and resize it to a fixed dimension for an ID card.

**Task 10: Extract and Rectify a Region from a Document Image**

Scenario: Extract a table from a scanned document and transform it to a standard rectangular form.