EXPERIMENT4: IMAGE-TRANSFORMATIONS

Aim

To perform image transformation such as Translation, Scaling, Shearing, Reflection, Rotation and Cropping using OpenCV and Python.

Software Required:

Anaconda - Python 3.7

Algorithm:

Step 1:

Import necessary libraries (cv2, numpy, matplotlib) and load the source image.

Step 2:

Create transformation matrices for translation, rotation, scaling, and shearing using functions like cv2.getRotationMatrix2D().

Step 3:

Apply the geometric transformations using cv2.warpAffine() for affine transformations and cv2.flip() for reflection.

Step 4:

Crop the image by selecting a specific rectangular region using NumPy array slicing.

Step 5:

Display the original and all transformed images with appropriate titles using matplotlib.pyplot.

Program:

Developed By: VIRUMAA HARISH M Register Number: 212223230246

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Step 1: Load the image
image = cv2.imread('me.jpeg')  # Load the image from file
# Display the original image
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for correct
plt.title("Original Image")
plt.axis('off')
(np.float64(-0.5), np.float64(1279.5), np.float64(812.5), np.float64(-0.5))

i)Image Translation

tx, ty = 100, 50  # Translation factors (shift by 100 pixels horizontally and 50 vert
M_translation = np.float32([[1, 0, tx], [0, 1, ty]])  # Translation matrix:
# [1, 0, tx] - Horizontal shift by tx
# [0, 1, ty] - Vertical shift by ty
translated_image = cv2.warpAffine(image, M_translation, (image.shape[1], image.shape[
plt.imshow(cv2.cvtColor(translated_image, cv2.COLOR_BGR2RGB))  # Display the translat
plt.title("Translated Image")
plt.axis('off')
(np.float64(-0.5), np.float64(1279.5), np.float64(812.5), np.float64(-0.5))

ii) Image Scaling

fx, fy = 5.0, 2.0  # Scaling factors (1.5x scaling for both width and height)
scaled_image = cv2.resize(image, None, fx=fx, fy=fy, interpolation=cv2.INTER_LINEAR)
# resize: Resize the image by scaling factors fx, fy
# INTER_LINEAR: Uses bilinear interpolation for resizing
plt.imshow(cv2.cvtColor(scaled_image, cv2.COLOR_BGR2RGB))  # Display the scaled image
plt.title("Scaled Image")  # Set title
plt.axis('off')
(np.float64(-0.5), np.float64(6399.5), np.float64(1625.5), np.float64(-0.5))

iii)Image shearing

shear_matrix = np.float32([[1, 0.5, 0], [0.5, 1, 0]])  # Shearing matrix
# The matrix shears the image by a factor of 0.5 in both x and y directions
# [1, 0.5, 0] - Shear along the x-axis (horizontal)
# [0.5, 1, 0] - Shear along the y-axis (vertical)
sheared_image = cv2.warpAffine(image, shear_matrix, (image.shape[1], image.shape[0]))
plt.imshow(cv2.cvtColor(sheared_image, cv2.COLOR_BGR2RGB))  # Display the sheared ima
plt.title("Sheared Image")  # Set title
plt.axis('off')
(np.float64(-0.5), np.float64(1279.5), np.float64(812.5), np.float64(-0.5))
```

iv)Image Reflection

```
reflected_image = cv2.flip(image, 2)  # Flip the image horizontally (1 means horizont
# flip: 1 means horizontal flip, 0 would be vertical flip, -1 would flip both axes
plt.imshow(cv2.cvtColor(reflected_image, cv2.COLOR_BGR2RGB))  # Display the reflected
plt.title("Reflected Image")  # Set title
plt.axis('off')
(np.float64(-0.5), np.float64(1279.5), np.float64(812.5), np.float64(-0.5))
```

v)Image Rotation

```
(height, width) = image.shape[:2]  # Get the image height and width
angle = 45  # Rotation angle in degrees (rotate by 45 degrees)
center = (width // 2, height // 2)  # Set the center of rotation to the image center
M_rotation = cv2.getRotationMatrix2D(center, angle, 1)  # Get the rotation matrix
# getRotationMatrix2D: Takes the center of rotation, angle, and scale factor (1 means
rotated_image = cv2.warpAffine(image, M_rotation, (width, height))  # Apply rotation
plt.imshow(cv2.cvtColor(rotated_image, cv2.COLOR_BGR2RGB))  # Display the rotated ima
plt.title("Rotated Image")  # Set title
plt.axis('off')
(np.float64(-0.5), np.float64(1279.5), np.float64(812.5), np.float64(-0.5))
```

vi)Image Cropping

```
x, y, w, h = 100, 100, 200, 150  # Define the top-left corner (x, y) and the width (w
# Cropping the image from coordinates (x, y) to (x+w, y+h)
cropped_image = image[y:y+h, x:x+w]
# The crop is performed by slicing the image array in the y and x directions
plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))  # Display the cropped ima
plt.title("Cropped Image")  # Set title
plt.axis('off')
(np.float64(-0.5), np.float64(199.5), np.float64(149.5), np.float64(-0.5))
```

Output:

i)Image Translation

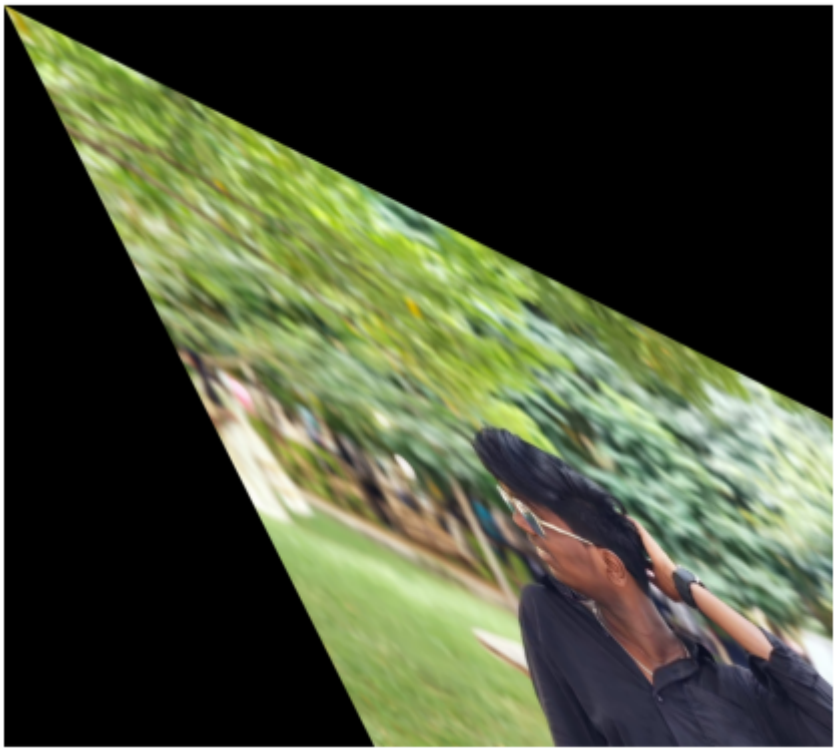
Translated Image

ii) Image Scaling

Scaled Image

iii)Image shearing

Sheared Image

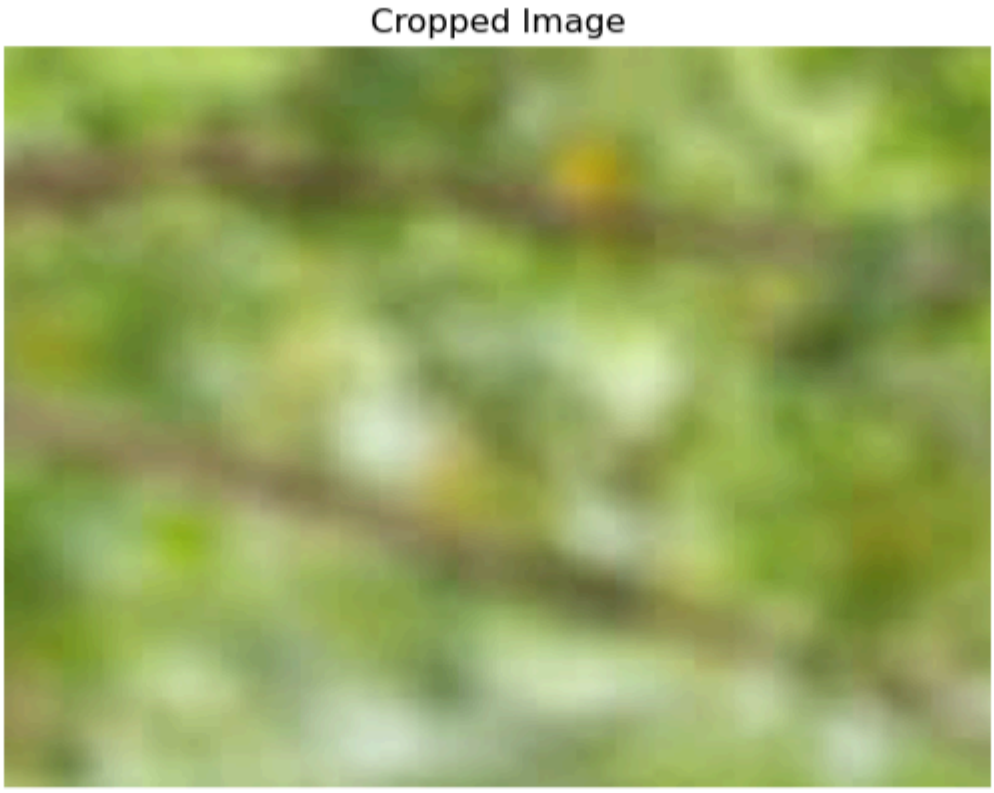


iv)Image Reflection

Reflected Image

v)Image Rotation

vi)Image Cropping



Result:

Thus the different image transformations such as Translation, Scaling, Shearing, Reflection, Rotation and Cropping are done using OpenCV and python programming.