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:

Step1:

Import necessary libraries such as OpenCV, NumPy, and Matplotlib for image processing and visualization.

Step2:

Read the input image using cv2.imread() and store it in a variable for further processing.

Step3:

Apply various transformations like translation, scaling, shearing, reflection, rotation, and cropping by defining corresponding functions:

1.Translation moves the image along the x or y-axis. 2.Scaling resizes the image by scaling factors. 3.Shearing distorts the image along one axis. 4.Reflection flips the image horizontally or vertically. 5.Rotation rotates the image by a given angle.

Step4:

Display the transformed images using Matplotlib for visualization. Convert the BGR image to RGB format to ensure proper color representation.

Step5:

Save or display the final transformed images for analysis and use plt.show() to display them inline in Jupyter or compatible environments.

Program:

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

# Function to display image using Matplotlib
def display_image(image, title):
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB for proper colo
    plt.imshow(image_rgb)
    plt.title(title)
    plt.axis('off')
    plt.show()

# Load an image
image = cv2.imread('tree.jpg')
display_image(image, 'Original Image')


# i) Image Translation
def translate(img, x, y):
    M = np.float32([[1, 0, x], [0, 1, y]])
    translated = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
    return translated

translated_image = translate(image, 100, 50)
display_image(translated_image, 'Translated Image')

# ii) Image Scaling
def scale(img, scale_x, scale_y):
    scaled = cv2.resize(img, None, fx=scale_x, fy=scale_y, interpolation=cv2.INTER_LINEAR)
    return scaled

scaled_image = scale(image, 1.5, 1.5)
display_image(scaled_image, 'Scaled Image')

# iii) Image Shearing
def shear(img, shear_factor):
    rows, cols, _ = img.shape
    M = np.float32([[1, shear_factor, 0], [0, 1, 0]])
    sheared = cv2.warpAffine(img, M, (cols, rows))
    return sheared

sheared_image = shear(image, 0.5)
display_image(sheared_image, 'Sheared Image')

# iv) Image Reflection
def reflect(img):
    reflected = cv2.flip(img, 1)  # 1 for horizontal flip
    return reflected

reflected_image = reflect(image)
display_image(reflected_image, 'Reflected Image')
```

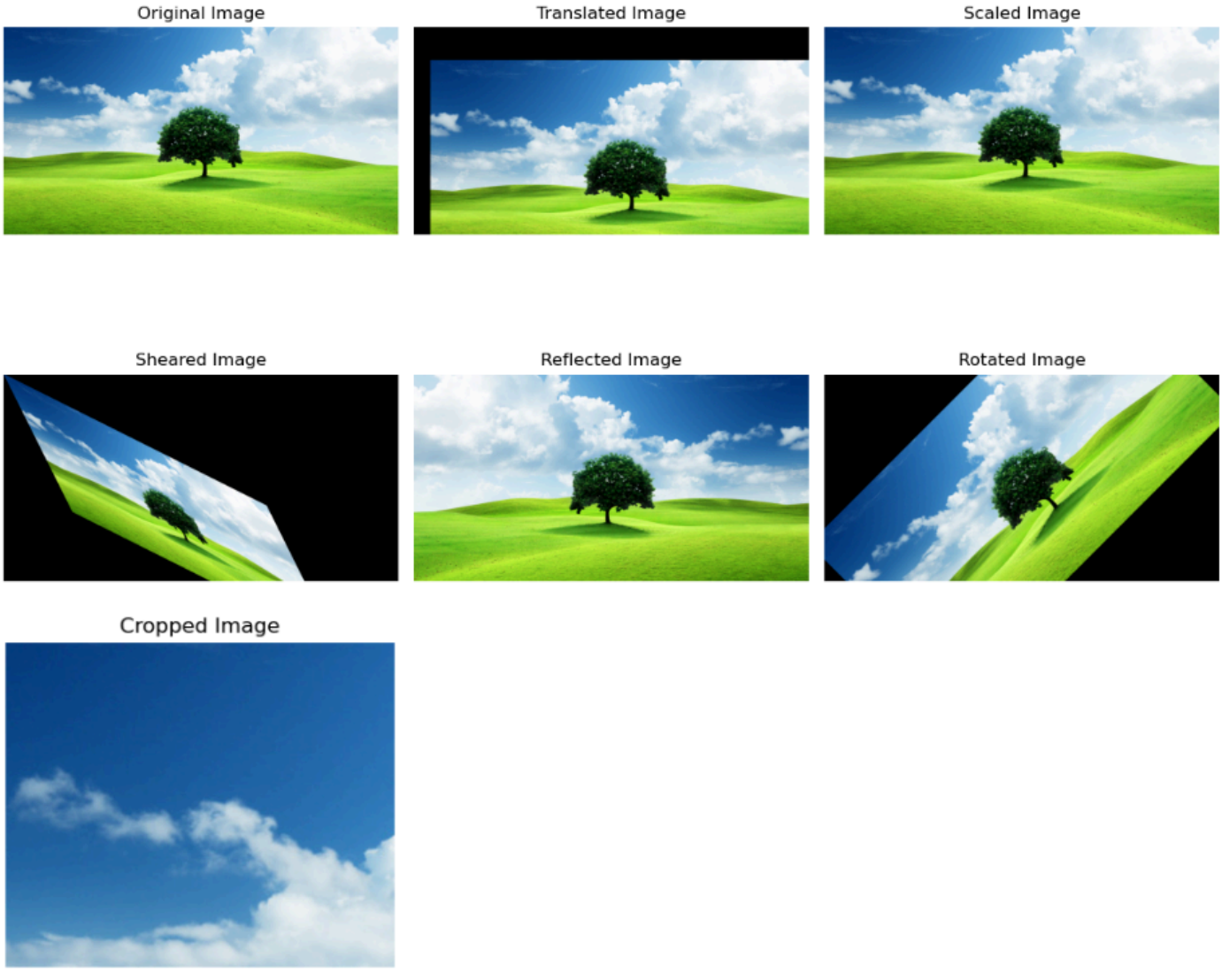
```
# v) Image Rotation
def rotate(img, angle):
    (h, w) = img.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(img, M, (w, h))
    return rotated

rotated_image = rotate(image, 45)
display_image(rotated_image, 'Rotated Image')

# vi) Image Cropping
def crop(img, start_row, start_col, end_row, end_col):
    cropped = img[start_row:end_row, start_col:end_col]
    return cropped

cropped_image = crop(image, 50, 50, 200, 200)
display_image(cropped_image, 'Cropped Image')
```

Output:



Result:

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