EXPERIMENT NO. 5

Title:

To perform Image Rotation, Translation, Scaling.

Objective:

- To understand basic geometric transformations in image processing.
- To implement image rotation, translation, and scaling using OpenCV.
- To visualize and analyze the effects of each transformation.

.

Brief Theory:

Geometric transformations are fundamental operations in image processing that modify the spatial relationship of pixels. They are used in a wide range of applications, including image alignment, object detection, and image augmentation.

1. **Image Rotation:**
    - Rotation moves an image around a central point (usually the center) by a specific angle.
    - Achieved using a rotation matrix and `cv2.warpAffine()` function in OpenCV.
2. **Image Translation:**
    - Translation shifts an image in the horizontal and/or vertical direction.
    - Defined using a 2x3 translation matrix that moves the image by given pixel values.
3. **Image Scaling:**
    - Scaling resizes an image by a given factor along the x and y axes.
    - Controlled using the `cv2.resize()` function with scaling factors `fx` and `fy`.

Steps: -

Step 1: Import Necessary Libraries

Use OpenCV, NumPy for image processing.

Code: -

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

Step 2: Load the Image

Code: -

```
# Load an image
image = cv2.imread('sample.jpg')
```

Step 3: Perform Image Rotation:

- Get image height and width.
- Define the rotation center and angle.
- Use `cv2.getRotationMatrix2D()` and `cv2.warpAffine()` to rotate the image.

Code: -

```
# Rotate Image
(h, w) = image.shape[:2]
center = (w // 2, h // 2)
matrix = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_img = cv2.warpAffine(image, matrix, (w, h))
```

Step 4: Perform Image Translation:

- Define the translation matrix with shift values.
- Use `cv2.warpAffine()` to apply the translation.

Code: -

```
# Translate Image
matrix = np.float32([[1, 0, 50], [0, 1, 30]])
translated_img = cv2.warpAffine(image, matrix, (w, h))
```

Step 5: Perform Image Scaling:

Use `cv2.resize()` with defined scale factors (`fx`, `fy`) and interpolation method.

Code: -

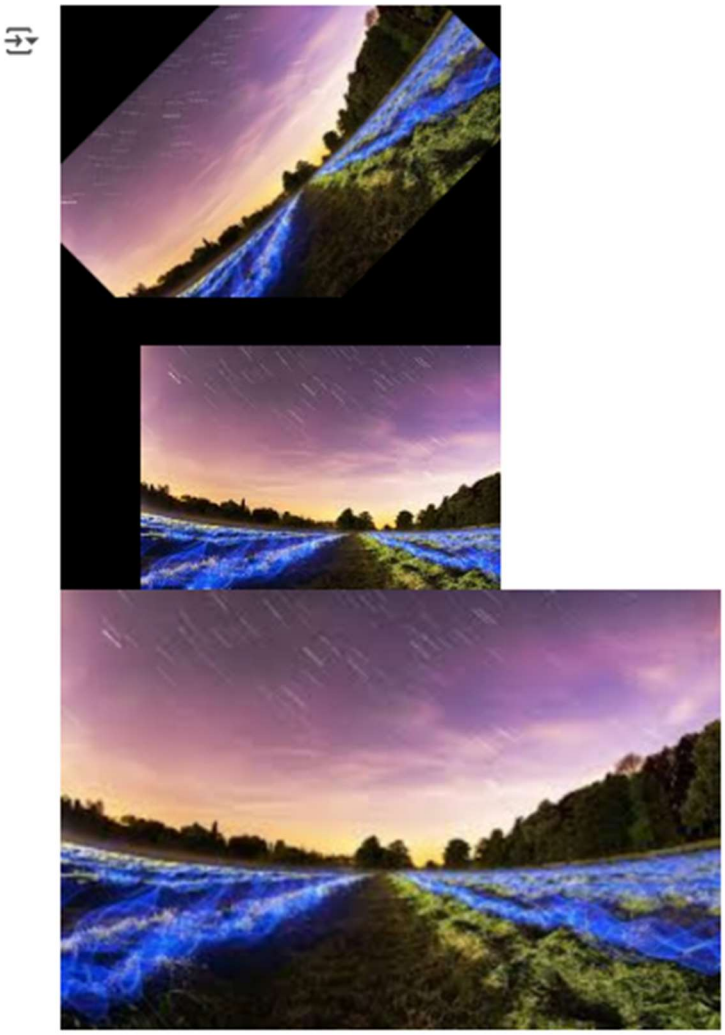
```
 # Scale Image
scaled_img = cv2.resize(image, None, fx=1.5, fy=1.5, interpolation=cv2.INTER_LINEAR)
```

Step 6: Display the Results:

Show the rotated, translated, and scaled images using `cv2_imshow()` or `cv2.imshow()` depending on the environment.

```
# Display results
cv2_imshow(rotated_img)
cv2_imshow(translated_img)
cv2_imshow(scaled_img)
```

Output:



Conclusion:

In this experiment, we successfully applied image rotation, translation, and scaling using OpenCV. Each transformation helps in manipulating the image's geometry and position. These are essential operations for tasks like image augmentation, pre-processing, and object tracking.

Practice Questions:

1. Rotate an image by 90 degrees around its center.
2. Translate an image to the left by 30 pixels and up by 50 pixels.
3. Scale an image to double its size using `cv2.resize()`.
4. Combine rotation and translation operations sequentially.
5. Scale down an image by 0.5 in both directions.
6. Perform rotation using a custom rotation center (not the image center).

Expected Oral Questions

1. What is the difference between image rotation and image translation?
2. How do you calculate the rotation matrix in OpenCV?
3. What is the function of `cv2.warpAffine()`?
4. Explain the purpose of scaling in image processing.
5. How can you preserve the image quality while scaling?
6. What interpolation methods are available in OpenCV?

FAQs in Interviews

1. **What is a transformation matrix and how is it used in OpenCV?**
   - A transformation matrix is a mathematical representation used to perform geometric transformations like rotation, translation, and scaling using `cv2.warpAffine()`.
2. **What are affine transformations?**
   - Affine transformations preserve points, straight lines, and planes. Rotation, translation, and scaling are all affine transformations.
3. **What is the difference between `cv2.resize()` and `cv2.warpAffine()`?**
   - `cv2.resize()` is specifically used for scaling, while `cv2.warpAffine()` is used for applying any affine transformation including rotation and translation.
4. **How do you choose the interpolation method in scaling?**
   - Based on whether you are enlarging (use `cv2.INTER_LINEAR` or `cv2.INTER_CUBIC`) or shrinking (use `cv2.INTER_AREA`) the image.
5. **What is the role of the rotation center in `cv2.getRotationMatrix2D()`?**
   - It determines the point around which the image will be rotated.
6. **Can geometric transformations be reversed?**
   - Yes, if the transformation parameters are known, inverse transformations can restore the original image.