Lab Assignment 1

Problem Statement

Write a Python program to perform different 2D and 3D transformation techniques on a given image.

Description

You will implement translations, rotations, scaling, and perspective transformations in 2D, and simulate a basic 3D transformation (e.g., perspective skew) to gain a foundational understanding of spatial transformations in computer vision.

Tools and Libraries

- **Python Libraries**: Use OpenCV for image processing, NumPy for matrix manipulations, and Matplotlib for visualization.
- **Image Requirements**: Choose any image to apply these transformations. Make sure to test each transformation separately for a better understanding of its effect on the image.

2D Transformations to Perform

- Translation: Shift the image along the x and y axes.
- Rotation: Rotate the image by a given angle around its center.
- Scaling: Resize the image by a specific scale factor.
- Perspective Transformation: Change the perspective by mapping points from one plane to another.

3D Transformation to Perform:

- Simulate 3D Rotation: Approximate a 3D effect using a rotation or perspective skew to simulate depth.

Approach (Pseudocode)

- Load the image using OpenCV.
- Define the transformation matrices for each of the transformations:
  - Translation Matrix: Create a matrix to shift the image.
  - Rotation Matrix: Define a rotation matrix with the center of rotation and angle.
  - Scaling Matrix: Use OpenCV to resize the image based on scaling factors.
  - Perspective Matrix: Identify four points on the original image and map them to a new perspective.

- Apply each transformation using OpenCV's functions and visualize using Matplotlib.
- Display the transformed images in a grid for comparison.

Hints:

- OpenCV Functions:
  - Use `cv2.imread()` to load the image.
  - Use `cv2.warpAffine()` for transformations like translation and rotation.
  - Use `cv2.getRotationMatrix2D()` to generate a rotation matrix.
  - Use `cv2.resize()` for scaling transformations.
  - Use `cv2.getPerspectiveTransform()` and `cv2.warpPerspective()` for perspective transformation.
- Matplotlib for Visualization:
  - Use `plt.imshow()` to display images.
  - Organize the output images in a grid using `plt.subplot()` to easily compare different transformations.

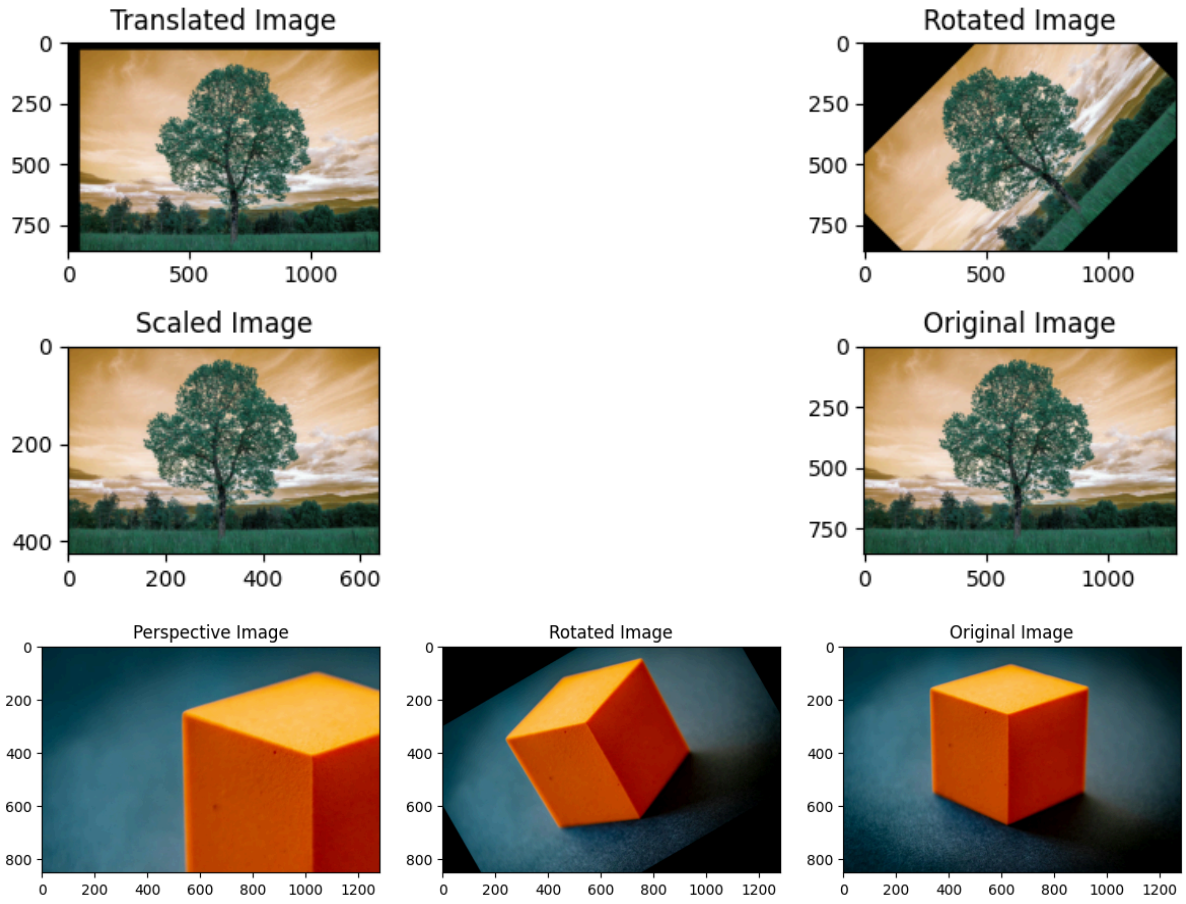
In [1]:
```
# Download assignment files
!wget https://github.com/buntyke/vnr_dlcv2024_labs/releases/download/DLCVLab1/cube_
!wget https://github.com/buntyke/vnr_dlcv2024_labs/releases/download/DLCVLab1/tree_
```

```
--2024-12-08 06:45:40--  https://github.com/buntyke/vnr_dlcv2024_labs/releases/dow
nload/DLCVLab1/cube_image.png
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e
65be/878811324/bfa9df01-3843-4e29-974f-911800768af8?X-Amz-Algorithm=AWS4-HMAC-SHA2
56&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Fs3%2Faws4_requ
est&X-Amz-Date=20241208T064540Z&X-Amz-Expires=300&X-Amz-Signature=6ca4c5d67fd2dbb2
2abefcccebfa28575775ce1eec477fa9cdbb61cba4257583&X-Amz-SignedHeaders=host&response
-content-disposition=attachment%3B%20filename%3Dcube_image.png&response-content-ty
pe=application%2Foctet-stream [following]
--2024-12-08 06:45:40--  https://objects.githubusercontent.com/github-production-r
elease-asset-2e65be/878811324/bfa9df01-3843-4e29-974f-911800768af8?X-Amz-Algorithm
=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2
Fs3%2Faws4_request&X-Amz-Date=20241208T064540Z&X-Amz-Expires=300&X-Amz-Signature=6
ca4c5d67fd2dbb22abefcccebfa28575775ce1eec477fa9cdbb61cba4257583&X-Amz-SignedHeader
s=host&response-content-disposition=attachment%3B%20filename%3Dcube_image.png&resp
onse-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.19
9.108.133, 185.199.111.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1294089 (1.2M) [application/octet-stream]
Saving to: ‘cube_image.png’

cube_image.png      100%[===================>]   1.23M  --.-KB/s    in 0.04s

2024-12-08 06:45:40 (29.9 MB/s) - ‘cube_image.png’ saved [1294089/1294089]

--2024-12-08 06:45:40--  https://github.com/buntyke/vnr_dlcv2024_labs/releases/dow
nload/DLCVLab1/tree_image.png
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e
65be/878811324/5f654664-e675-40c3-aca7-b770953d114f?X-Amz-Algorithm=AWS4-HMAC-SHA2
56&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Fs3%2Faws4_requ
est&X-Amz-Date=20241208T064541Z&X-Amz-Expires=300&X-Amz-Signature=0ad360ad5a808ff8
5d162f9ec13d5b9b48d24404e45808305d1f96722964f9a8&X-Amz-SignedHeaders=host&response
-content-disposition=attachment%3B%20filename%3Dtree_image.png&response-content-ty
pe=application%2Foctet-stream [following]
--2024-12-08 06:45:41--  https://objects.githubusercontent.com/github-production-r
elease-asset-2e65be/878811324/5f654664-e675-40c3-aca7-b770953d114f?X-Amz-Algorithm
=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2
Fs3%2Faws4_request&X-Amz-Date=20241208T064541Z&X-Amz-Expires=300&X-Amz-Signature=0
ad360ad5a808ff85d162f9ec13d5b9b48d24404e45808305d1f96722964f9a8&X-Amz-SignedHeader
s=host&response-content-disposition=attachment%3B%20filename%3Dtree_image.png&resp
onse-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2098806 (2.0M) [application/octet-stream]
Saving to: ‘tree_image.png’

tree_image.png      100%[===================>]   2.00M  --.-KB/s    in 0.05s

2024-12-08 06:45:41 (37.0 MB/s) - ‘tree_image.png’ saved [2098806/2098806]
```

In [2]:

```
### WRITE CODE HERE ###
import io
import cv2
import numpy as np
from PIL import Image
from matplotlib import pyplot as plt

# Load the image
image = cv2.imread('tree_image.png')

# 2D translation
tx, ty = 50, 30 # Translation in x and y
translation_matrix = np.float32([[1, 0, tx], [0, 1, ty]])
translated_image = cv2.warpAffine(image, translation_matrix, (image.shape[1],
                                                              image.shape[0]))

# 2D rotation
angle = 45 # Rotation angle in degrees
rotation_matrix = cv2.getRotationMatrix2D((image.shape[1] / 2, image.shape[0] / 2),
                                          angle, 1)
rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1],
                                                        image.shape[0]))

# 2D scaling
scale_factor = 0.5
scaled_image = cv2.resize(image, None, fx=scale_factor, fy=scale_factor)

# Display the transformed images
plt.figure(figsize=(12, 4))
plt.subplot(221), plt.imshow(translated_image), plt.title('Translated Image')
plt.subplot(222), plt.imshow(rotated_image), plt.title('Rotated Image')
plt.subplot(223), plt.imshow(scaled_image), plt.title('Scaled Image')
plt.subplot(224), plt.imshow(image), plt.title('Original Image')
plt.tight_layout()
plt.show()

# Load the image
image = cv2.imread('cube_image.png')
pts1 = np.float32([[10, 10], [100, 10], [10, 100], [100, 100]])
pts2 = np.float32([[0, 0], [150, 0], [0, 150], [150, 150]])

# Calculate the perspective transformation matrix
perspective_matrix = cv2.getPerspectiveTransform(pts1, pts2)

# Apply the perspective transformation
perspective_image = cv2.warpPerspective(image, perspective_matrix, (image.shape[1],
                                                                    image.shape[0]))

# Simulate 3D rotation
angle = 30 # Rotation angle in degrees
rotation_matrix = cv2.getRotationMatrix2D((image.shape[1] / 2, image.shape[0] / 2),
                                          angle, 1)
rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1],
                                                        image.shape[0]))

# Display the transformed images
plt.figure(figsize=(12, 4))
plt.subplot(131), plt.imshow(cv2.cvtColor(perspective_image, cv2.COLOR_BGR2RGB)),
                                                  plt.title('Perspective Image')
plt.subplot(132), plt.imshow(cv2.cvtColor(rotated_image, cv2.COLOR_BGR2RGB)),
                                                    plt.title('Rotated Image')
plt.subplot(133), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)),
                                                plt.title('Original Image')
```

```
plt.tight_layout()
plt.show()
```



In [ ]: