| **EXERCISE NO:5**  **DATE:13/02/2025** | **IMAGE TRANSFORMATION** |
| --- | --- |

**BILLIONAIRE IMAGE:**

**ALGORITHM:**

1. Import OpenCV and necessary libraries
2. Read Image → cv2.imread()
3. Convert BGR to RGB → cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
4. Display Original Image

#### Image Transformations

1. Translation → Shift image using pixel offsets (bx, by)
2. Rotation → Rotate image by a defined angle using a transformation matrix
3. Scaling → Resize image using scale factors (sx, sy)
4. Shearing → Apply shear transformation along x and y axes
5. Reflection → Compute horizontal, vertical, and diagonal flips
6. Display Transformed Images using matplotlib

**CODE:**

import cv2

import matplotlib.pyplot as plt

import numpy as np

image = cv2.imread("billionaire1.jpg")

image\_rgb = cv2.cvtColor(image,cv2.COLOR\_BGR2RGB)

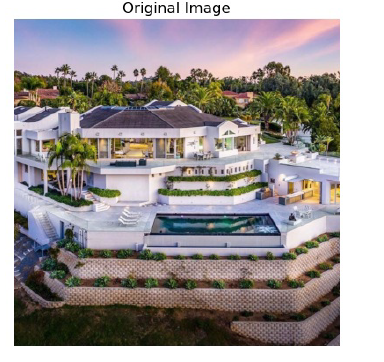
plt.imshow(image\_rgb)

plt.title('Original Image')

plt.axis('off')

plt.show()

**OUTPUT:**



**IMAGE TRANSLATION**

**CODE:**

rows,cols,channels= image\_rgb.shape

bx=47

by=47

translated\_image=np.zeros\_like(image)

for y in range(cols):

for x in range(rows):

new\_x=x+bx

new\_y=y+bx

if 0<=new\_x<cols and 0<=new\_y<rows:

translated\_image[new\_y,new\_x]=image\_rgb[y,x]

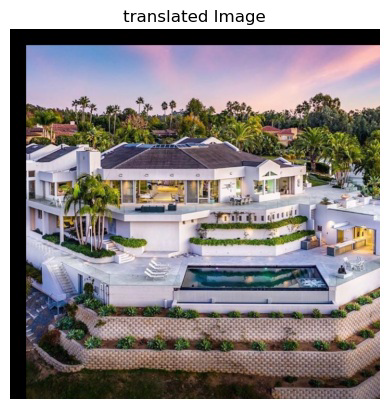
plt.imshow(translated\_image)

plt.axis("off")

plt.title("translated Image")

plt.show()

**OUTPUT:**



**IMAGE ROTATION:**

**CODE:**

rows,cols,channels=image\_rgb.shape

rotated\_image=np.zeros\_like(image\_rgb)

angle=10

angle\_rad=np.radians(angle)

rotated\_matrix=np.array([[np.cos(angle\_rad),-np.sin(angle\_rad)],[np.sin(angle\_rad),np.cos(angle\_rad)]])

cx,cy=cols//2,rows//2

for y in range(rows):

for x in range(cols):

tx,ty=x-cx,y-cy

new\_tx,new\_ty=np.dot(rotated\_matrix ,np.array([tx,ty]))

new\_x,new\_y = int(new\_tx + cx),int(new\_ty + cy)

if 0<=new\_x<cols and 0<=new\_y<rows:

rotated\_image[new\_y,new\_x] = image\_rgb[y,x]

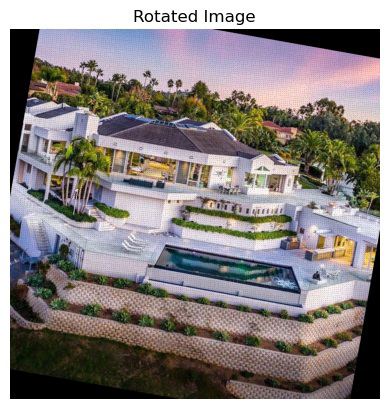
plt.imshow(rotated\_image)

plt.axis('off')

plt.title('Rotated Image')

plt.show()

**OUTPUT:**

****

**IMAGE SCALING:**

**CODE:**

sx ,sy = 2.5,1.0

scaled\_image = np.zeros\_like(image\_rgb)

cx ,cy = cols//2,rows//2

for y in range(rows):

for x in range(cols):

tx , ty = x - cx , y - cy

new\_tx,new\_ty = tx \* sx,ty \* sy

new\_x,new\_y = int(new\_tx + cx),int(new\_ty + cy)

if 0<=new\_x<cols and 0<=new\_y<rows:

scaled\_image[new\_y,new\_x] = image\_rgb[y,x]

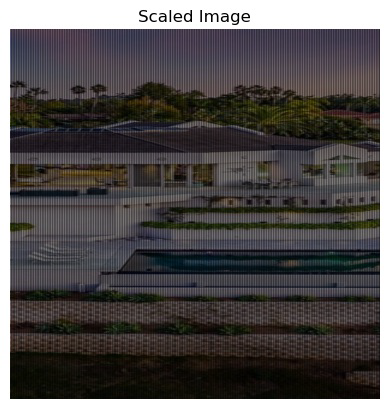
plt.imshow(scaled\_image)

plt.axis('off')

plt.title('Scaled Image')

plt.show()

**OUTPUT:**

****

**IMAGE SHEAR:**

**CODE:**

shear\_x, shear\_y = 0.25, 0.25 # Shear in the x direction

sheared\_image = np.zeros\_like(image\_rgb)

# Find the center of the image

cx, cy = cols // 2, rows // 2

for y in range(rows):

for x in range(cols):

tx, ty = x - cx, y - cy

new\_tx = tx + shear\_x \* ty # Apply shear in x direction

new\_ty = ty + shear\_y \* tx # Apply shear in y direction

new\_x, new\_y = int(new\_tx + cx), int(new\_ty + cy)

# Ensure the new coordinates are within bounds

if 0 <= new\_x < cols and 0 <= new\_y < rows:

sheared\_image[new\_y, new\_x] = image\_rgb[y, x]

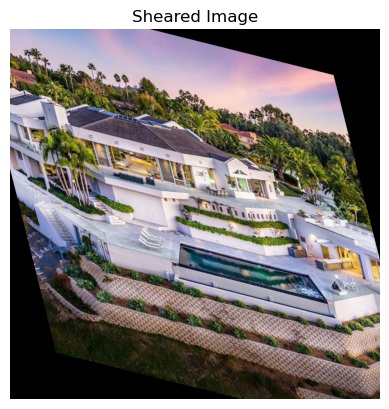
plt.imshow(sheared\_image)

plt.axis('off')

plt.title('Sheared Image')

plt.show()

**OUTPUT:**

****

**IMAGE REFLECTION:**

**CODE:**

horizontal\_reflection = np.zeros\_like(image\_rgb)

vertical\_reflection = np.zeros\_like(image\_rgb)

diagonal\_reflection = np.zeros\_like(image\_rgb)

# Perform Manual Reflection

for y in range(rows):

for x in range(cols):

# Horizontal Flip (Mirror Left to Right)

horizontal\_reflection[y, x] = image\_rgb[y, cols - x - 1]

# Vertical Flip (Mirror Top to Bottom)

vertical\_reflection[y, x] = image\_rgb[rows - y - 1, x]

# Diagonal Flip (Both Axes)

diagonal\_reflection[y, x] = image\_rgb[rows - y - 1, cols - x - 1]

plt.figure(figsize=(10, 5))

plt.subplot(1, 4, 1)

plt.imshow(image\_rgb)

plt.axis('off')

plt.title("Original Image")

plt.subplot(1, 4, 2)

plt.imshow(horizontal\_reflection)

plt.axis('off')

plt.title("Horizontal Reflection")

plt.subplot(1, 4, 3)

plt.imshow(vertical\_reflection)

plt.axis('off')

plt.title("Vertical Reflection")

plt.subplot(1, 4, 4)

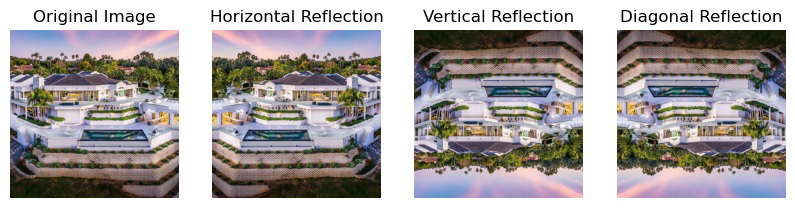
plt.imshow(diagonal\_reflection)

plt.axis('off')

plt.title("Diagonal Reflection")

plt.show()

**OUTPUT:**

****

**RESULT:**

**We have successfully executed the code…**