TRANSFORMATION

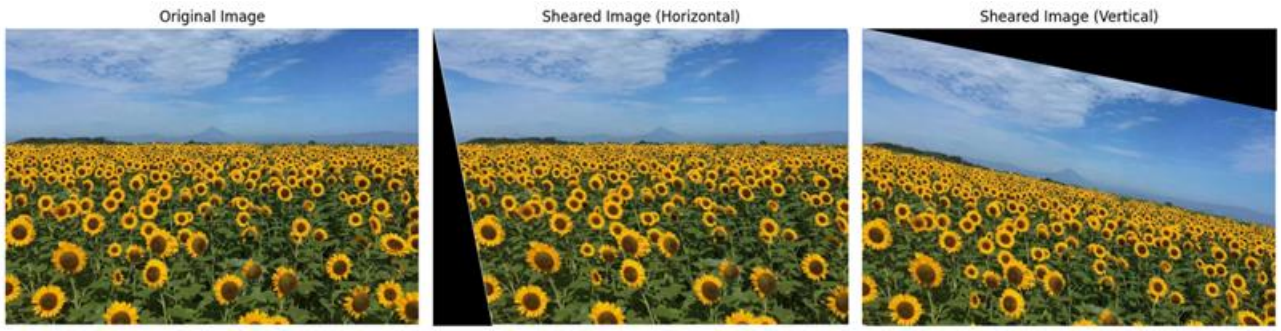
Krithika 22011101046

I. SHEARING

CODE:

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

# Load the image
image = cv2.imread(r'/content/image.png')

# Convert from BGR to RGB (OpenCV loads images in BGR by default)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Define shear matrix for the x-axis (horizontal shear)
shear_factor_x = 0.2
M_x = np.float32([[1, shear_factor_x, 0],
[0, 1, 0]])

# Define shear matrix for the y-axis (vertical shear)
shear_factor_y = 0.2
M_y = np.float32([[1, 0, 0],
[shear_factor_y, 1, 0]])

# Apply shear to the image (horizontal shear)
sheared_image_x = cv2.warpAffine(image, M_x, (image.shape[1], image.shape[0]))

# Apply shear to the image (vertical shear)
sheared_image_y = cv2.warpAffine(image, M_y, (image.shape[1], image.shape[0]))

# Show original and sheared images
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(image)
axes[0].set_title('Original Image')
axes[0].axis('off')

axes[1].imshow(sheared_image_x)
axes[1].set_title('Sheared Image (Horizontal)')
axes[1].axis('off')

axes[2].imshow(sheared_image_y)
axes[2].set_title('Sheared Image (Vertical)')
axes[2].axis('off')
```

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



II. EUCLIDEAN TRANSFORMATION (ROTATION AND TRANSLATION)

CODE:

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

# Load an image
image = cv2.imread(r'/content/image.png')

# Get the image's dimensions (height, width)
height, width = image.shape[:2]

# Define the center of the image for rotation
center = (width / 2, height / 2)

# Define the rotation angle
angle = 45 # Rotate 45 degrees

# Create the rotation matrix (without scaling)
rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1) # No scaling (scale=1)

# Apply the rotation to the image
rotated_image = cv2.warpAffine(image, rotation_matrix, (width, height))

# Add translation (shifting the image along X and Y axes)
tx, ty = 100, 50 # Translate 100 pixels along X and 50 pixels along Y
translation_matrix = np.float32([[1, 0, tx], [0, 1, ty]])

# Apply the translation to the rotated image
final_image = cv2.warpAffine(rotated_image, translation_matrix, (width, height))

# Display the original and transformed images using cv2_imshow
```

```
cv2_imshow(image) # Display the original image
cv2_imshow(final_image) # Display the rotated and translated image
```

III. PROJECTIVE TRANSFORMATION

CODE:

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

# Load the image
image = cv2.imread(r'/content/image.png')

source_points = np.float32([[50, 50], [200, 50], [50, 200], [200, 200]]) # Example points in the source image
destination_points = np.float32([[10, 100], [250, 50], [100, 250], [250, 250]]) # Example points in the destination image

# Calculate the Homography matrix
H, status = cv2.findHomography(source_points, destination_points)
```

```
# Get the shape of the image
height, width, channels = image.shape

# Apply the projective transformation to the image
warped_image = cv2.warpPerspective(image, H, (width, height))

# Show the original and transformed images
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)) # Original image
plt.title("Original Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(warped_image, cv2.COLOR_BGR2RGB)) # Transformed image
plt.title("Warped Image")
plt.axis('off')

plt.show()

# Save the warped image
cv2.imwrite('warped_image.jpg', warped_image)
```