# Install required libraries if not already installed

!pip install opencv-python pillow matplotlib

# Import necessary libraries

import cv2

import numpy as np

from google.colab import files

from PIL import Image

import matplotlib.pyplot as plt

# Step 1: Upload an image

uploaded = files.upload()

# Open the image using PIL and convert it to a format OpenCV understands

image\_path = list(uploaded.keys())[0]

image = Image.open(image\_path)

image\_cv = np.array(image)

# Convert from RGB to BGR (since OpenCV uses BGR by default)

image\_cv = cv2.cvtColor(image\_cv, cv2.COLOR\_RGB2BGR)

# Get the image dimensions (height, width)

height, width = image\_cv.shape[:2]

# Step 2: Define the points for Affine Transformation

# These are the 3 points from the original image (source points)

pts1 = np.float32([[50, 50], [200, 50], [50, 200]])

# Define the destination points to map the source points to

pts2 = np.float32([[10, 100], [200, 50], [100, 250]])

# Step 3: Compute the Affine Transformation Matrix

affine\_matrix = cv2.getAffineTransform(pts1, pts2)

# Step 4: Apply the Affine Transformation

transformed\_image = cv2.warpAffine(image\_cv, affine\_matrix, (width, height))

# Step 5: Convert the transformed image back to RGB for displaying with matplotlib

transformed\_image\_rgb = cv2.cvtColor(transformed\_image, cv2.COLOR\_BGR2RGB)

# Step 6: Display the original and transformed images side by side

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

# Display original image

plt.subplot(1, 2, 1)

plt.imshow(image)

plt.title("Original Image")

plt.axis('off')

# Display the transformed image

plt.subplot(1, 2, 2)

plt.imshow(transformed\_image\_rgb)

plt.title("Affine Transformed Image")

plt.axis('off')

plt.show()

# Optional: Save and download the transformed image

cv2.imwrite("affine\_transformed\_image.jpg", transformed\_image)

files.download("affine\_transformed\_image.jpg")

