# 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)

# Step 2: Define corresponding points in the source and destination images

# Example points (4 points in the source and destination image)

# Source points (can be any 4 points from the original image)

pts\_src = np.float32([[50, 50], [400, 50], [50, 400], [400, 400]])

# Destination points (mapped to new locations)

pts\_dst = np.float32([[10, 100], [400, 50], [100, 500], [450, 450]])

# Step 3: Calculate the Homography matrix

homography\_matrix, \_ = cv2.findHomography(pts\_src, pts\_dst)

# Step 4: Apply the Homography transformation

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

warped\_image = cv2.warpPerspective(image\_cv, homography\_matrix, (width, height))

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

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

# Step 6: Display the original and warped 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 warped image

plt.subplot(1, 2, 2)

plt.imshow(warped\_image\_rgb)

plt.title("Warped Image (Homography)")

plt.axis('off')

plt.show()

# Optional: Save and download the warped image

cv2.imwrite("warped\_image.jpg", warped\_image)

files.download("warped\_image.jpg")

