CSE483:

Computer Vision

Lab 1

Name: Alaa Mohamed Mohamed Hamdy Ahmed

ID: 19P6621

Group: 1 Section: 1

# Task Required

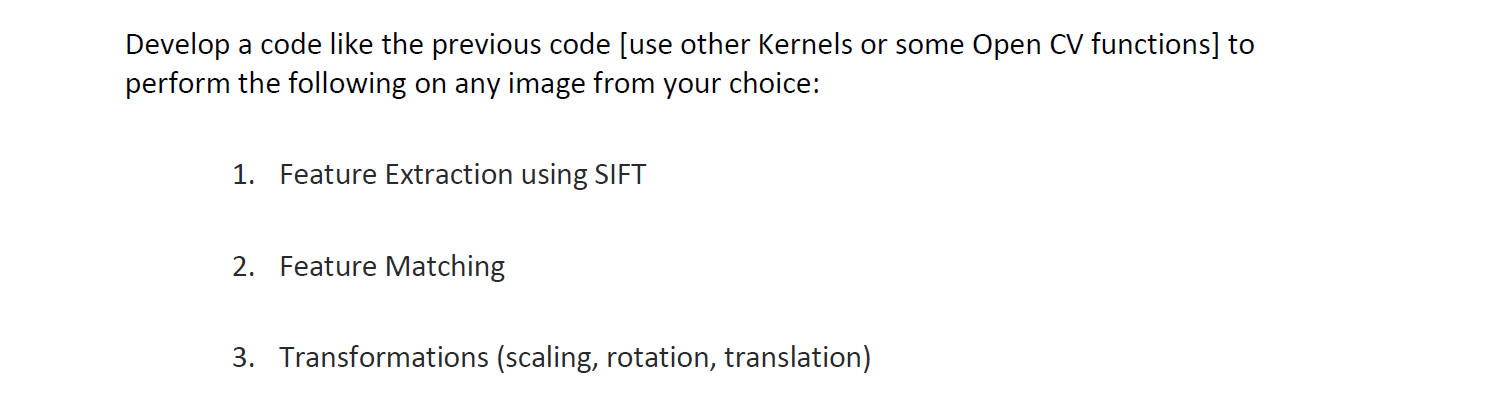


Table of Contents

[Task Required 2](#_Toc134569742)

[Test Images 5](#_Toc134569743)

[SIFT for Feature Extraction 5](#_Toc134569744)

[Code 11](#_Toc134569745)

[Applying Transformations 12](#_Toc134569746)

[Translation 12](#_Toc134569747)

[Code 13](#_Toc134569748)

[Output 14](#_Toc134569749)

[Rotation 15](#_Toc134569750)

[Code 16](#_Toc134569751)

[Output 17](#_Toc134569752)

[Scaling 18](#_Toc134569753)

[Code 19](#_Toc134569754)

[Output 20](#_Toc134569755)

[Github repository 21](#_Toc134569756)

Table of Contents

[Figure 1 Image for feature matching 5](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569626)

[Figure 2 Image for feature extraction 5](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569627)

[Figure 3 Extracted features from Image used for feature matching 8](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569628)

[Figure 4 Extracted features from Image used for feature extraction 8](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569629)

[Figure 5 Matched Image 10](#_Toc134569630)

[Figure 6 Translated image 14](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569631)

[Figure 7 Non-Translated Image 14](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569632)

[Figure 8 Un-rotated image 17](file:///C:\Users\Alaa%20Hamdy\Desktop\Visiona%20Assigment\Lab1.docx#_Toc134569633)

[Figure 9 Rotated Image 17](#_Toc134569634)

[Figure 10 Original size image 20](#_Toc134569635)

[Figure 11 Resized image 20](#_Toc134569636)

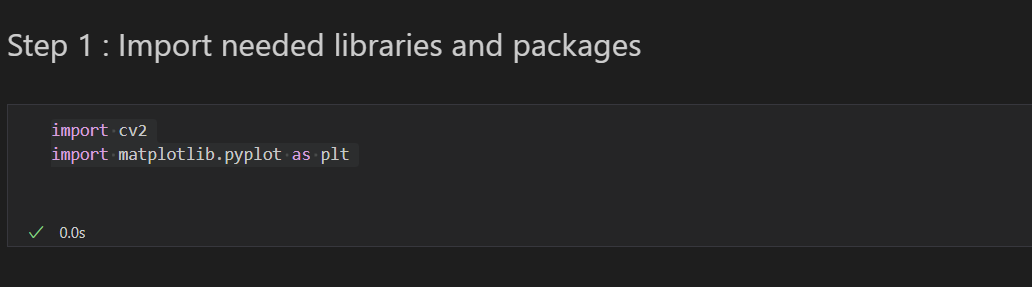
# Test Images

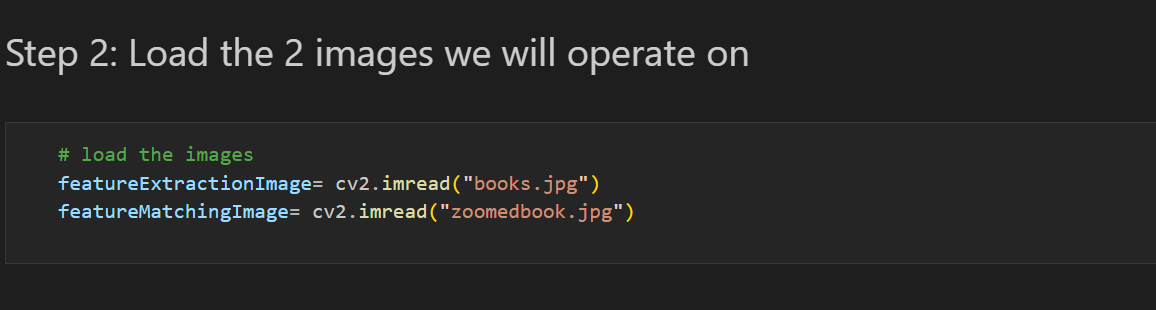


Figure 1 Image for feature matching

Figure 2 Image for feature extraction

# SIFT for Feature Extraction

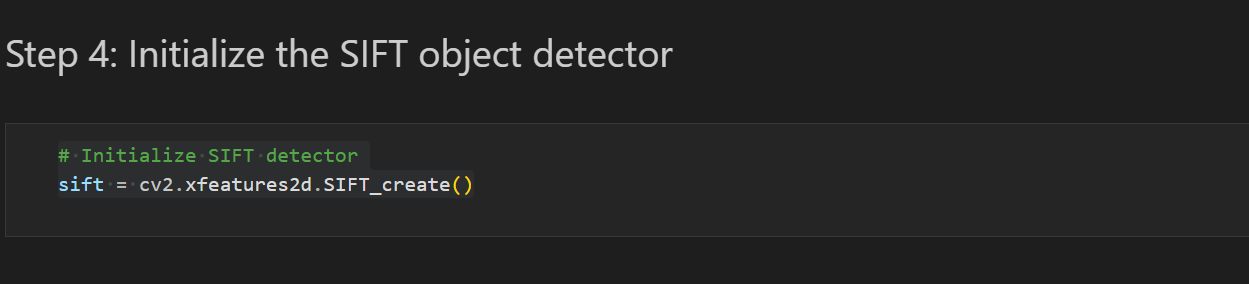




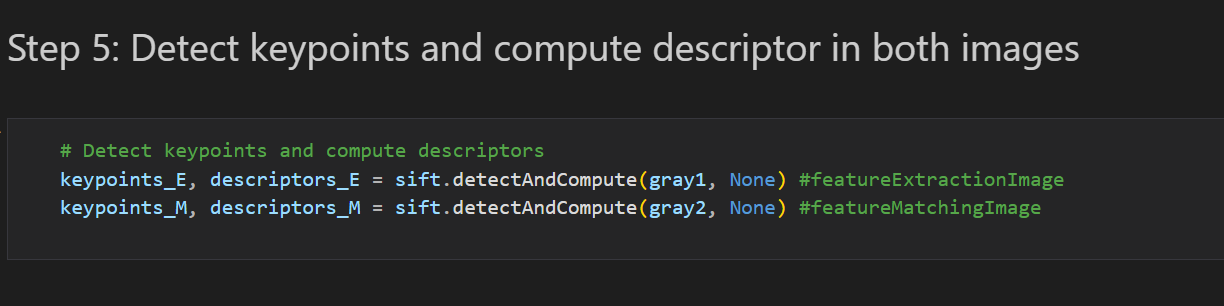
Load th two images from disk and stores them in the featureExtractionImage and featureMatchingImage variables.



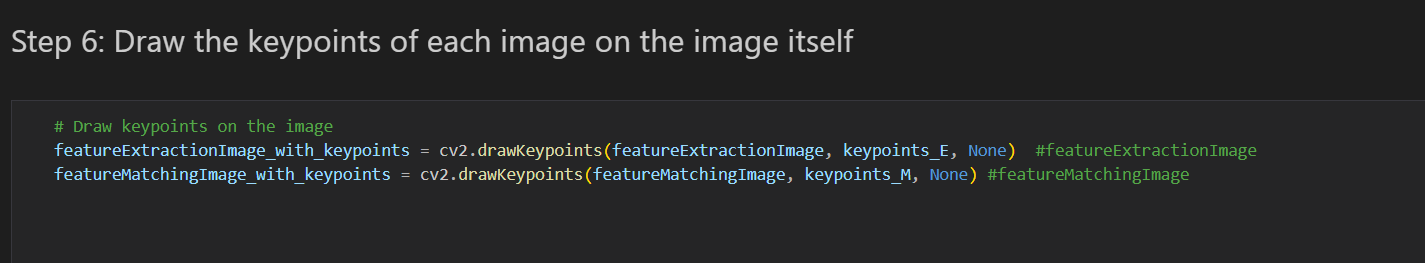
Convert the color images to grayscale using the cv2.cvtColor() function. This is necessary because most feature detectors and descriptors work on grayscale images.



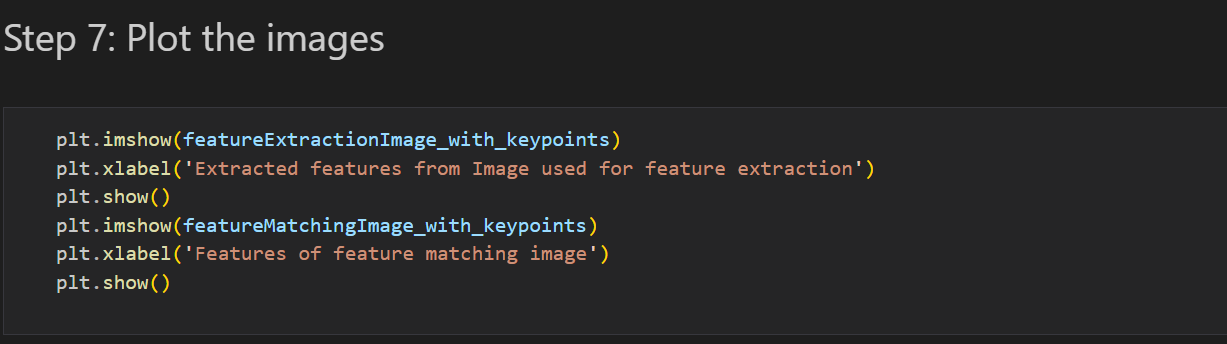
This code creates an instance of the SIFT detector using the cv2.xfeatures2d.SIFT\_create() function.



Uses the detectAndCompute() method of the SIFT detector to extract keypoints and descriptors for both images. The keypoints and descriptors are stored in the keypoints\_E, keypoints\_M, descriptors\_E, and descriptors\_M variables.



Draw the extracted keypoints of each image on the image itself, using drawKeypoints() function



Plot the 2 images

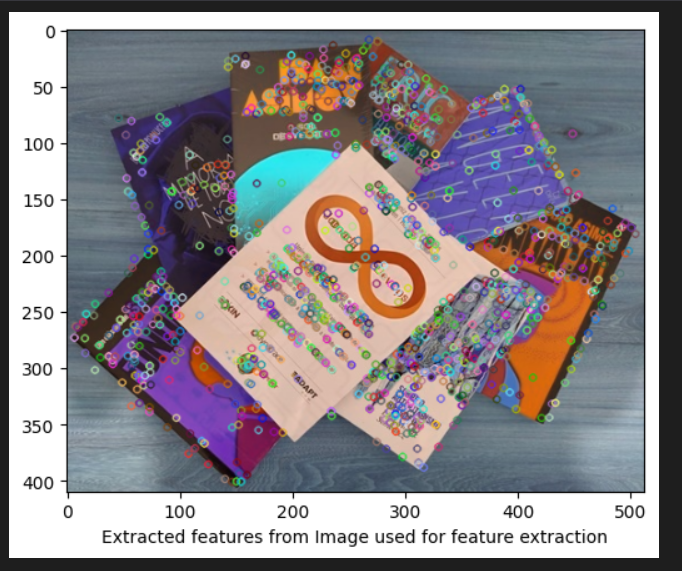
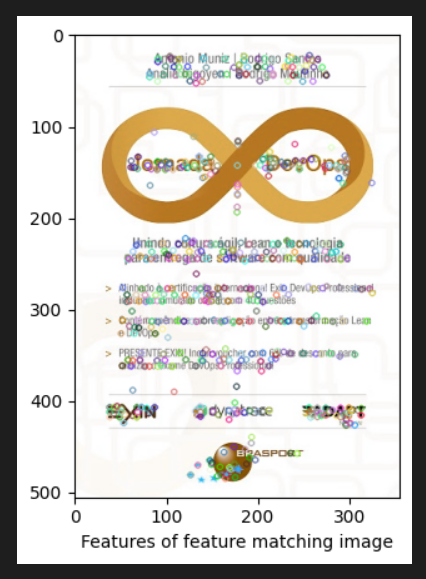
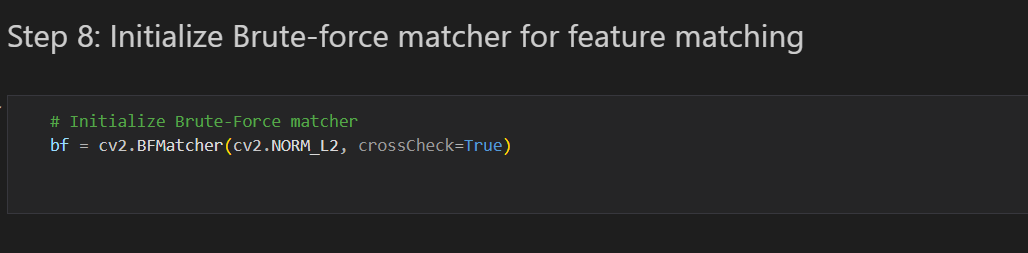
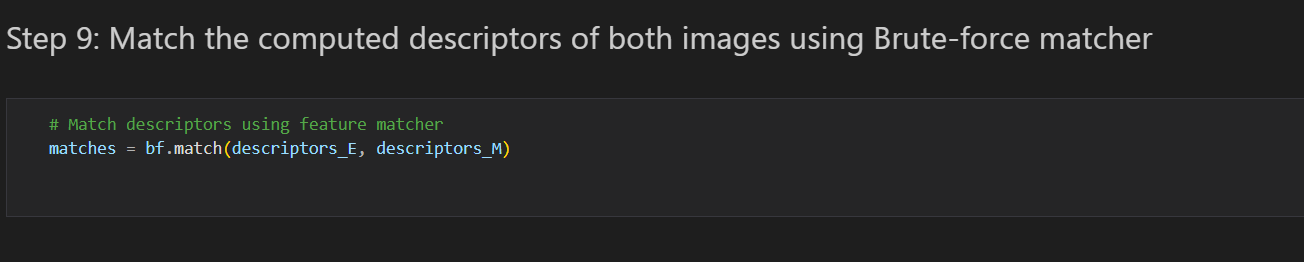
****

Figure 3 Extracted features from Image used for feature matching

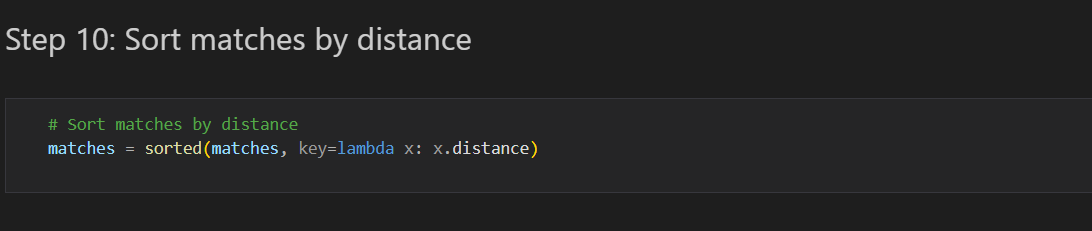
Figure 4 Extracted features from Image used for feature extraction

****

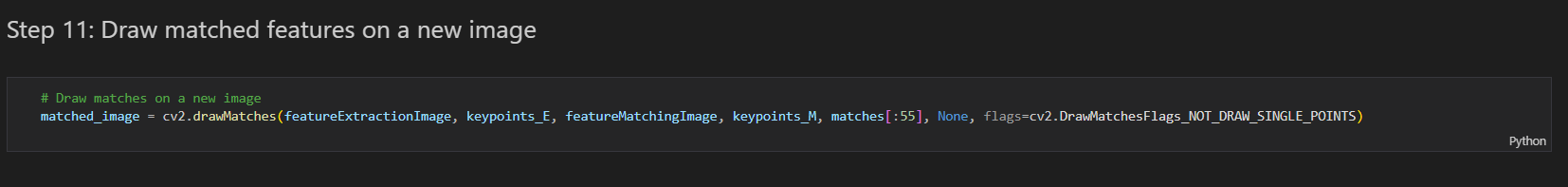
Create an instance of the Brute-Force matcher with L2 distance using the cv2.BFMatcher() function.

****

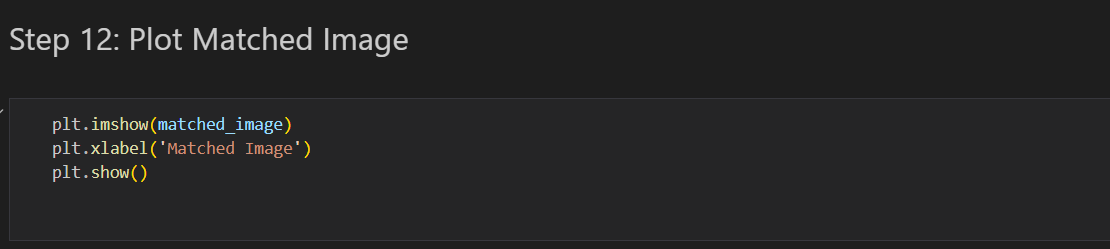
Match the descriptors from both images using the match() method of the matcher. The matches are stored in the matches variable.

****

Sort the matches by distance using the sorted() function and a lambda function that extracts the distance value from each match.

****

Use the drawMatches() function to draw the top 55 matches on a new image. The function takes the two input images, the keypoints and descriptors for both images, the matches to be drawn, and some optional parameters. The resulting image is stored in the matched\_image variable.

****

Plot the matched image

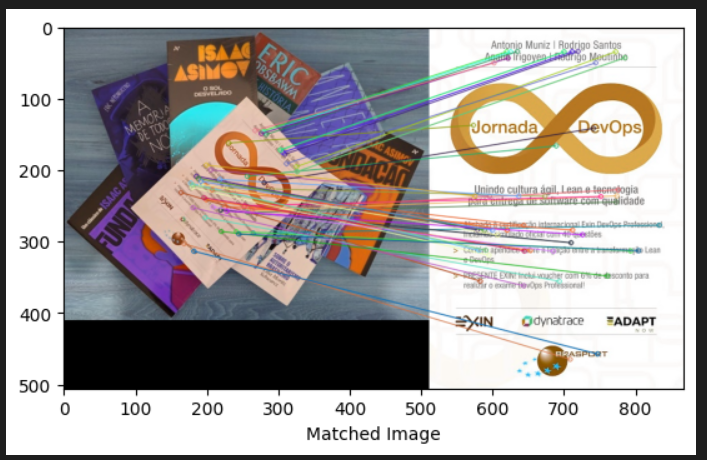
****

Figure 5 Matched Image

## Code

import cv2

import matplotlib.pyplot as plt

# load the images

featureExtractionImage= cv2.imread("books.jpg")

featureMatchingImage= cv2.imread("zoomedbook.jpg")

# Convert the images to grayscale

gray1 = cv2.cvtColor(featureExtractionImage, cv2.COLOR\_BGR2GRAY) #featureExtractionImage

gray2 = cv2.cvtColor(featureMatchingImage, cv2.COLOR\_BGR2GRAY)  #featureMatchingImage

# Initialize SIFT detector

sift = cv2.xfeatures2d.SIFT\_create()

# Detect keypoints and compute descriptors

#featureExtractionImage

keypoints\_E, descriptors\_E = sift.detectAndCompute(gray1, None)

#featureMatchingImage

keypoints\_M, descriptors\_M = sift.detectAndCompute(gray2, None)

# Draw keypoints on the image

featureExtractionImage\_with\_keypoints = cv2.drawKeypoints(featureExtractionImage, keypoints\_E, None)  #featureExtractionImage

featureMatchingImage\_with\_keypoints = cv2.drawKeypoints(featureMatchingImage, keypoints\_M, None) #featureMatchingImage

# Plot features extracted on both images

plt.imshow(featureExtractionImage\_with\_keypoints)

plt.xlabel('Extracted features from Image used for feature extraction')

plt.show()

plt.imshow(featureMatchingImage\_with\_keypoints)

plt.xlabel('Features of feature matching image')

plt.show()

# Initialize Brute-Force matcher

bf = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=True)

# Match descriptors using feature matcher

matches = bf.match(descriptors\_E, descriptors\_M)

# Sort matches by distance

matches = sorted(matches, key=lambda x: x.distance)

# Draw matches on a new image

matched\_image = cv2.drawMatches(featureExtractionImage, keypoints\_E, featureMatchingImage, keypoints\_M, matches[:55], None, flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)

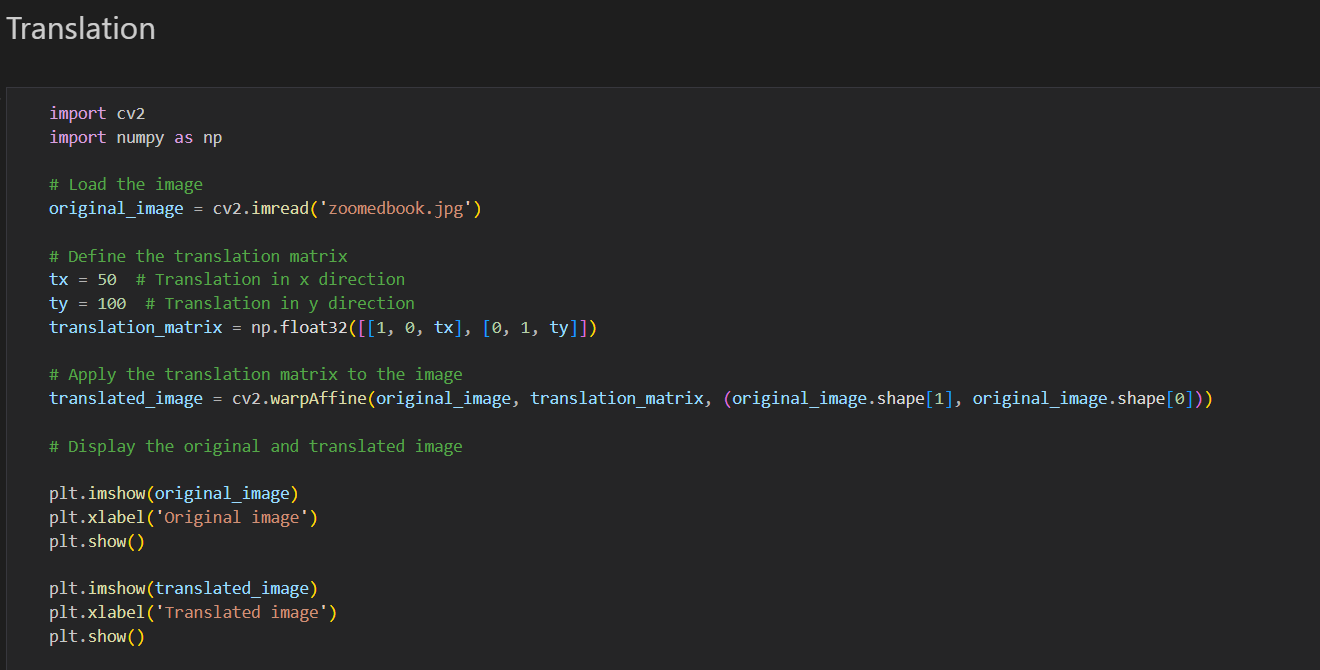
plt.imshow(matched\_image)

plt.xlabel('Matched Image')

plt.show()

# Applying Transformations

## Translation



We first load the image using the cv2.imread() function. Then, we define the translation matrix using the np.float32() function. The translation matrix is a 2x3 matrix, where the first row represents the x direction translation (in pixels), the second row represents the y direction translation (in pixels), and the third row is a dummy row (not used in translation).

Next, we apply the translation matrix to the image using the cv2.warpAffine() function. The function takes the image, the transformation matrix, and the output image size as input, and returns the translated image.

Note that in this example, we have translated the image by 50 pixels in the x direction and 100 pixels in the y direction, but you can use any translation values you want. Also, keep in mind that translating an image can result in a loss of image content if the translated area goes beyond the boundaries of the original image.

### Code

import cv2

import numpy as np

# Load the image

original\_image = cv2.imread('zoomedbook.jpg')

# Define the translation matrix

tx = 50  # Translation in x direction

ty = 100  # Translation in y direction

translation\_matrix = np.float32([[1, 0, tx], [0, 1, ty]])

# Apply the translation matrix to the image

translated\_image = cv2.warpAffine(original\_image, translation\_matrix, (original\_image.shape[1], original\_image.shape[0]))

# Display the original and translated image

plt.imshow(original\_image)

plt.xlabel('Original image')

plt.show()

plt.imshow(translated\_image)

plt.xlabel('Translated image')

plt.show()

### Output

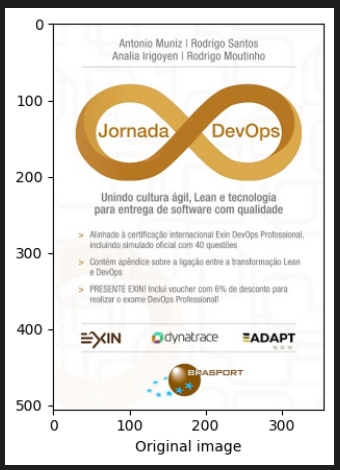




Figure 6 Translated image

Figure 7 Non-Translated Image

## Rotation

We first load the image using the cv2.imread() function, and then we get the dimensions of the image using the shape attribute. We then define the rotation angle and the rotation point, which is the center of the image.

Next, we use the cv2.getRotationMatrix2D() function to create the rotation matrix. This function takes three arguments: the center of rotation, the rotation angle in degrees, and the scale factor (which is set to 1.0 in this example).

Finally, we apply the rotation transformation to the image using the cv2.warpAffine() function, which takes three arguments: the input image, the transformation matrix, and the output image size. We then plot the original and the rotated image

### Code

import cv2

import numpy as np

# Load the image

original\_image = cv2.imread('books.jpg')

# Get the image dimensions

(height, width) = original\_image.shape[:2]

# Define the rotation angle

angle = 45

# Define the rotation point (center of the image)

center = (width // 2, height // 2)

# Create the rotation matrix

M = cv2.getRotationMatrix2D(center, angle, 1.0)

# Apply the rotation transformation to the image

rotated\_image = cv2.warpAffine(original\_image, M, (width, height))

# Display the original and rotated image

plt.imshow(original\_image)

plt.xlabel('Original image')

plt.show()

plt.imshow(rotated\_image)

plt.xlabel('Rotated image')

plt.show()

### Output



Figure 8 Un-rotated image

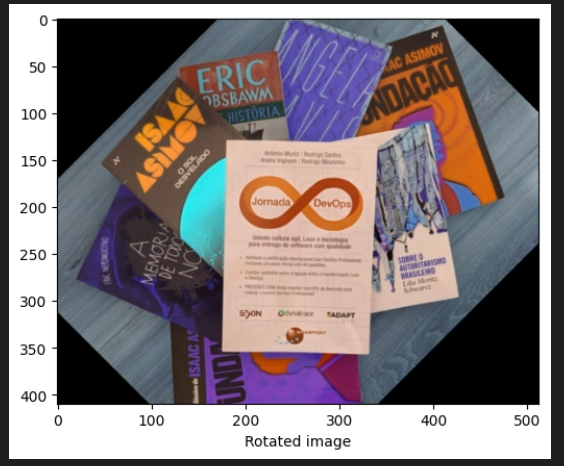


Figure 9 Rotated Image

## Scaling

We first load the image using the cv2.imread() function. Then, we get the original image size using the shape attribute of the image. Next, we define the scale factor (in this case, 0.5), which will be used to calculate the new image dimensions. We calculate the new dimensions by multiplying the original dimensions with the scale factor and converting them to integers using the int() function. Finally, we use the cv2.resize() function to resize the image to the new dimensions.

Note that in this example, we have scaled the image by half, but you can use any scale factor you want. Also, keep in mind that scaling an image can lead to distortion, so it's important to choose the right scale factor for your specific use case.

### Code

import cv2

# Load the image

original\_image = cv2.imread('books.jpg')

# Get the original image size

original\_height, original\_width = original\_image.shape[:2]

# Define the scale factor

scale\_factor = 0.5

# Calculate the new image dimensions

new\_height = int(original\_height \* scale\_factor)

new\_width = int(original\_width \* scale\_factor)

# Resize the image

resized\_image = cv2.resize(original\_image, (new\_width, new\_height))

# Display the original and resized image

plt.imshow(original\_image)

plt.xlabel('Original image')

plt.show()

plt.imshow(resized\_image)

plt.xlabel('Resized image')

plt.show()

### Output



Figure 10 Original size image

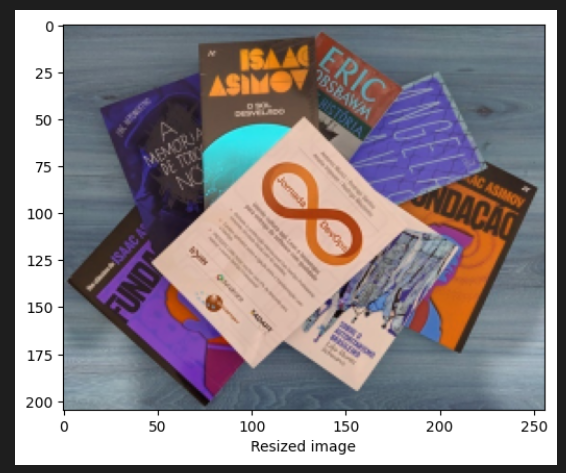


Figure 11 Resized image

Note that the image x-axis scale have been reduced to the half.

## Github repository

https://github.com/Alaa-Hamdy/Lab1-CSE483-Computer-Vision.git