EEL_6990_EXAM_1

Garfield Jugar

Exam 1
EEL 6990
University of West Florida

Problem Number		
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

Problem 1) (20 Points)

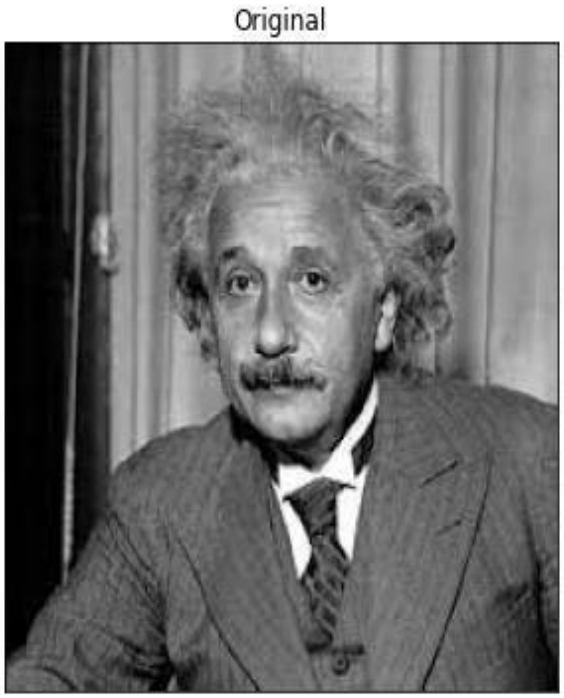
For the image titled “prob1.jpeg”, apply,

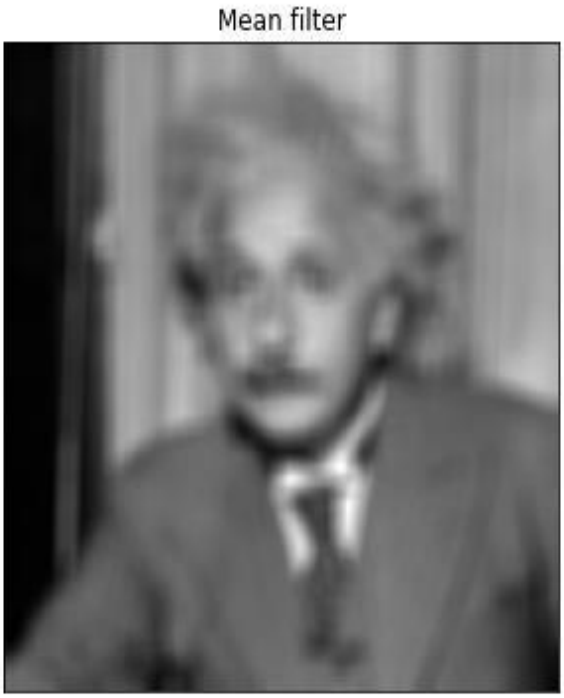
- mean filter
- sharpening filter and show your results. Add your script as appendix.

PART A – Mean Filter

To eliminate noise from an image, the mean filter is applied. The mean of the pixel values within an n × n kernel must be determined. After that, the mean replaces the pixel intensity of the central element. This smooths the image's borders and removes some of the image's noise. A mean filter can be applied to a picture with the blur function from the Open-CV library.

When working with color photos, photos first should be converted from RGB to HSV since RGB's dimensions are interdependent, whereas HSV's three dimensions are independent (this allows us to apply filters to each of the three dimensions separately.)

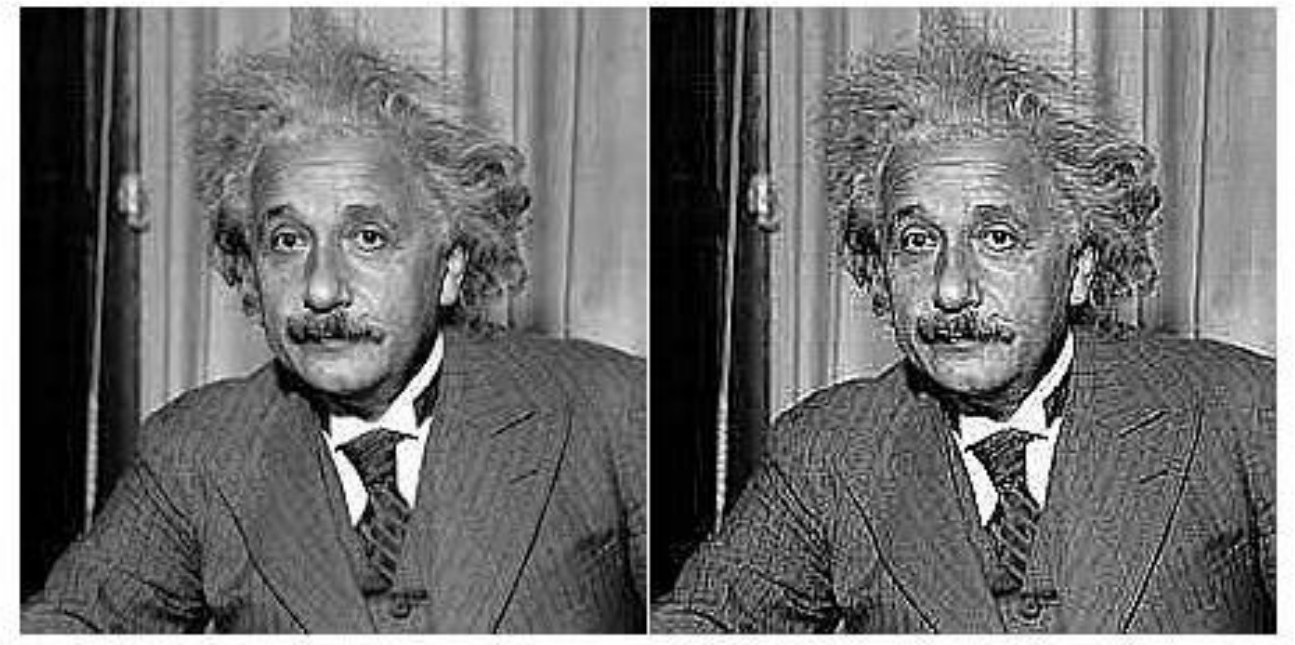




PART B – Sharpening Filter

Python contains the Pillow library's ImageFilter.SHARPEN class. When implemented, it implements a spatial filter that uses convolution to sharpen a given picture. In addition, the open() function of the Pillow's Image class is used to create an image object by giving the file name of the Image. The filter() function on the Image object is used to apply a filter to an image. Only the filter's class name is supplied as a parameter.

The name of the filter class supplied in the image below is ImageFilter.SHARPEN, and an object of it is created within. The convolution matrix for sharpening is available in ImageFilter.SHARPEN.



Original **sharpened**

Problem 2) (20 Points)

For the image titled “prob2.jpg”, apply Harris detector, and show your results. Add your script as appendix.

The Harris Corner Detection technique was created to locate an image's internal corners. The corners of an image are defined as the areas where there are considerable fluctuations in gradient intensity in all possible dimensions and directions. Corners extracted can be utilized to extract correct information because they are part of the image features that can be matched with features from other images. Harris Corner Detection is a method for extracting the corners of an input image as well as features from it.

Original

Transformed

Problem 3) (20 Points)

Using images titled “prob3-1.jpg” and “prob3-2.jpg”, implement SIFT and SURF feature detectors, and match detected features. Compare performance of two methods in terms of number of matches found between two images and show your results. You can find code from online resources.

PART A – SIFT

SIFT's high-level details are as follows:

Scale-space Identify locations and scales that can be allocated repeatedly under multiple viewpoints of the same scene or object using Extrema Detection.

Fit a model to establish the placement and scale of features, then select important locations based on stability.

Assignment for Orientation: Calculate the optimal orientation (or orientations) for each keypoint region.

Description of the main point: To characterize each keypoint region, use local image gradients at the scale and rotation you want.

An image's scale-space is a function L(x, y, a) obtained by convolutioning a Gaussian kernel (at various scales) with the input image.

The keypoint in the images is found using the sift.detect() function. If you simply want to search a portion of an image, you can use a mask. Each keypoint is a unique structure with various characteristics such as its (x,y) coordinates, the size of the meaningful neighborhood, the angle that describes its orientation, the reaction that specifies keypoint strength, and so on.

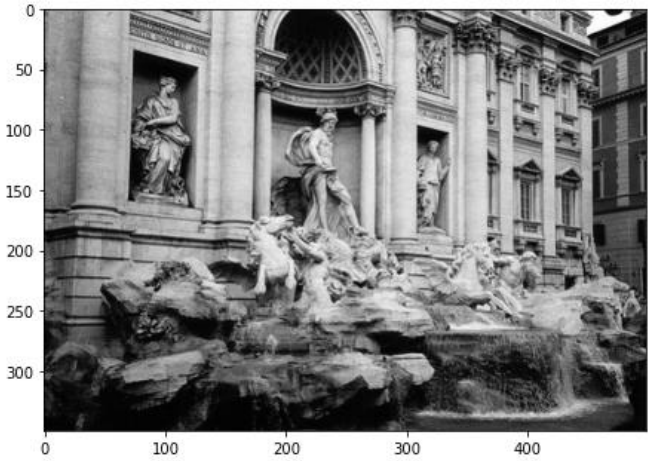
The cv2.drawKeyPoints() method in OpenCV additionally creates tiny circles on the locations of keypoints. If you provide it the flag cv2.DRAW MATCHES FLAGS DRAW RICH KEYPOINTS, it will create a circle with the keypoint's size and orientation.

The images used

For feature matching, we'll now employ the SIFT features. I was given two images for this reason, each taken from a different perspective.

The two photos I used are as follows:

]: <matplotlib.image.AxesImage at 0x26927fe2a90>



The number of matches found were

see the attched.ipynb file

Now we'll generate the SIFT features for both of these photographs. To get the keypoints, we must first create a SIFT object and then use the function detectAndCompute. The keypoints and descriptors will be returned as two values.

Let's find the keypoints in each image and display the total number of keypoints identified.

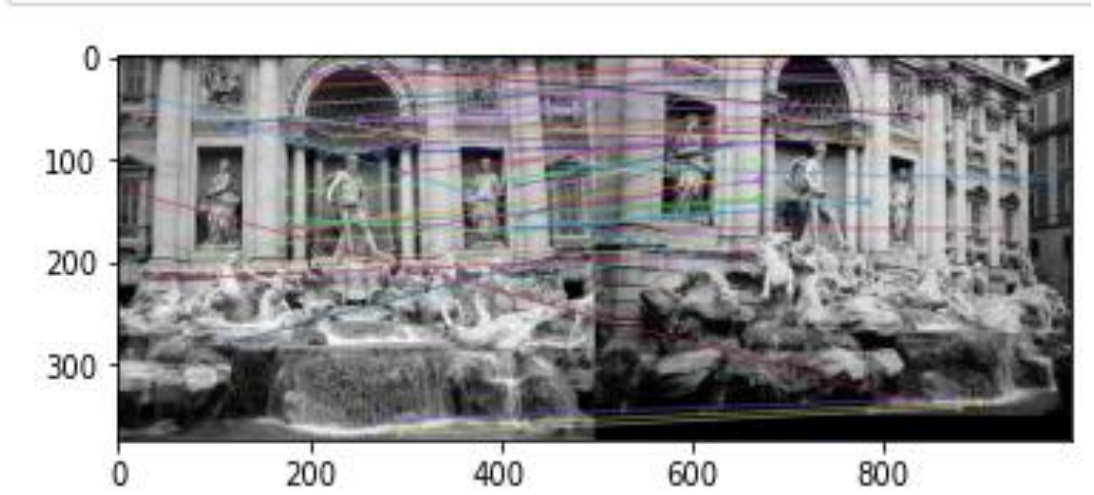
```
img1 = cv2.imread('prob3-1.jpg')
img2 = cv2.imread('prob3-2.jpg')

img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

#sift
sift = cv2.SIFT_create()

keypoints_1, descriptors_1 = sift.detectAndCompute(img1,None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2,None)

len(keypoints_1), len(keypoints_2)
```

Out[2]: (1923, 2274)

matched features between two images with lines

see the attched.ipynb file

Let's now try to match the features in image 1 with those in image 2. The match() function from the BFmatcher (brute force match) package will be used. We'll also draw lines across the features in both photographs that match. The drawMatches function in OpenCV can be used to accomplish this.



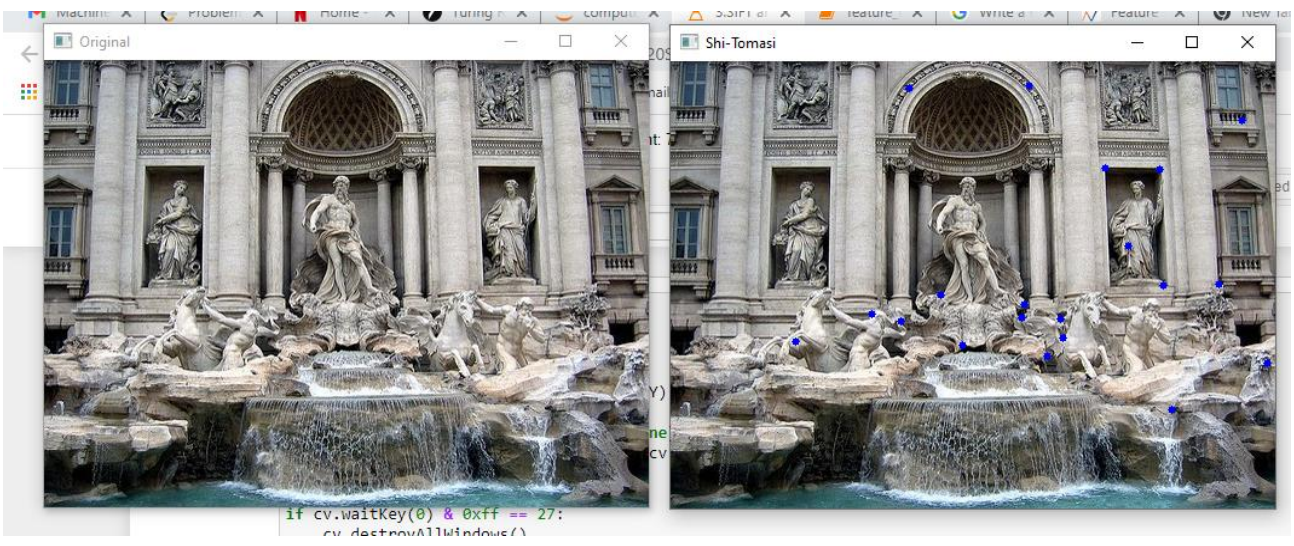
PART B – Shi Tomasi

Apart from the way the score (R) is derived, Shi-Tomasi is nearly identical to the Harris Corner detector. This produces a better outcome. Furthermore, we can locate the top N corners using this method, which may be handy if we don't want to detect each and every corner.

Shi-Tomasi proposed a new scoring function

$$R = \min(\lambda_1, \lambda_2)$$

If the score R for a pixel is larger than a particular threshold, the pixel is considered a corner. If we map this in 12 space, it looks similar to the Harris Corner Detector.



Comparison between SIFT and SHI Tomasi

The SIFT detector finds centers of blob-like features. Shi-Tomasi detector finds corners. Hence not possible for comparison Furthermore, SIFT detector operates at multiple scales, while the classic Shi-Tomasi does not.

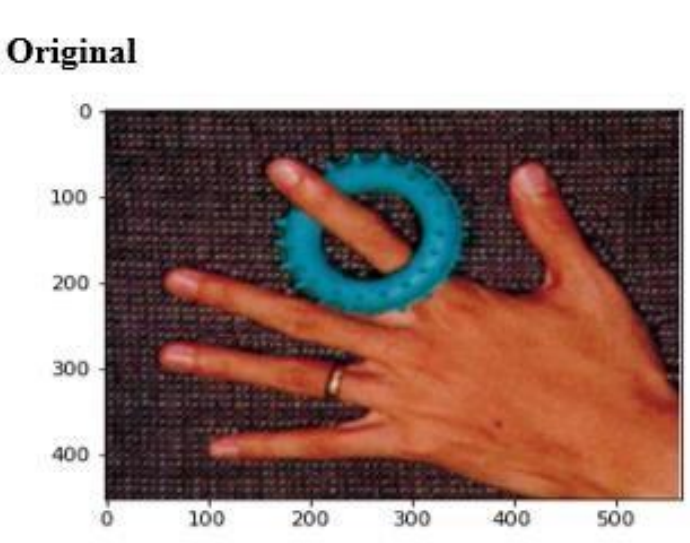
.

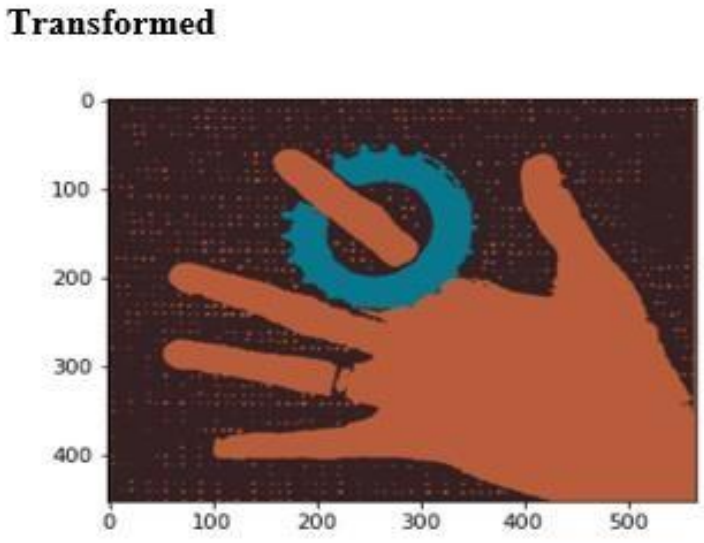
Problem 4) (20 Points)

For the image titled “prob4.png”, implement k- means clustering based segmentation, and show your results. You can find code from online resources.

Clustering algorithms are unsupervised algorithms, meaning they don't use labeled data. It is used to distinguish between different classes or clusters of data depending on how similar the data is. Data points from the same group are more like each other than data points from other groups.

One of the most widely used clustering methods is K-means clustering. The number of clusters is represented by k.





Problem 5) (20 Points)

Using images titled “prob5-1.jpg”, “prob5-2.jpg”, and “prob5-3.jpg”, implement feature detection & matching with method of your choice, resample all of the images onto the first image’s coordinate frame and align, and show your results. You can find code from online resources.

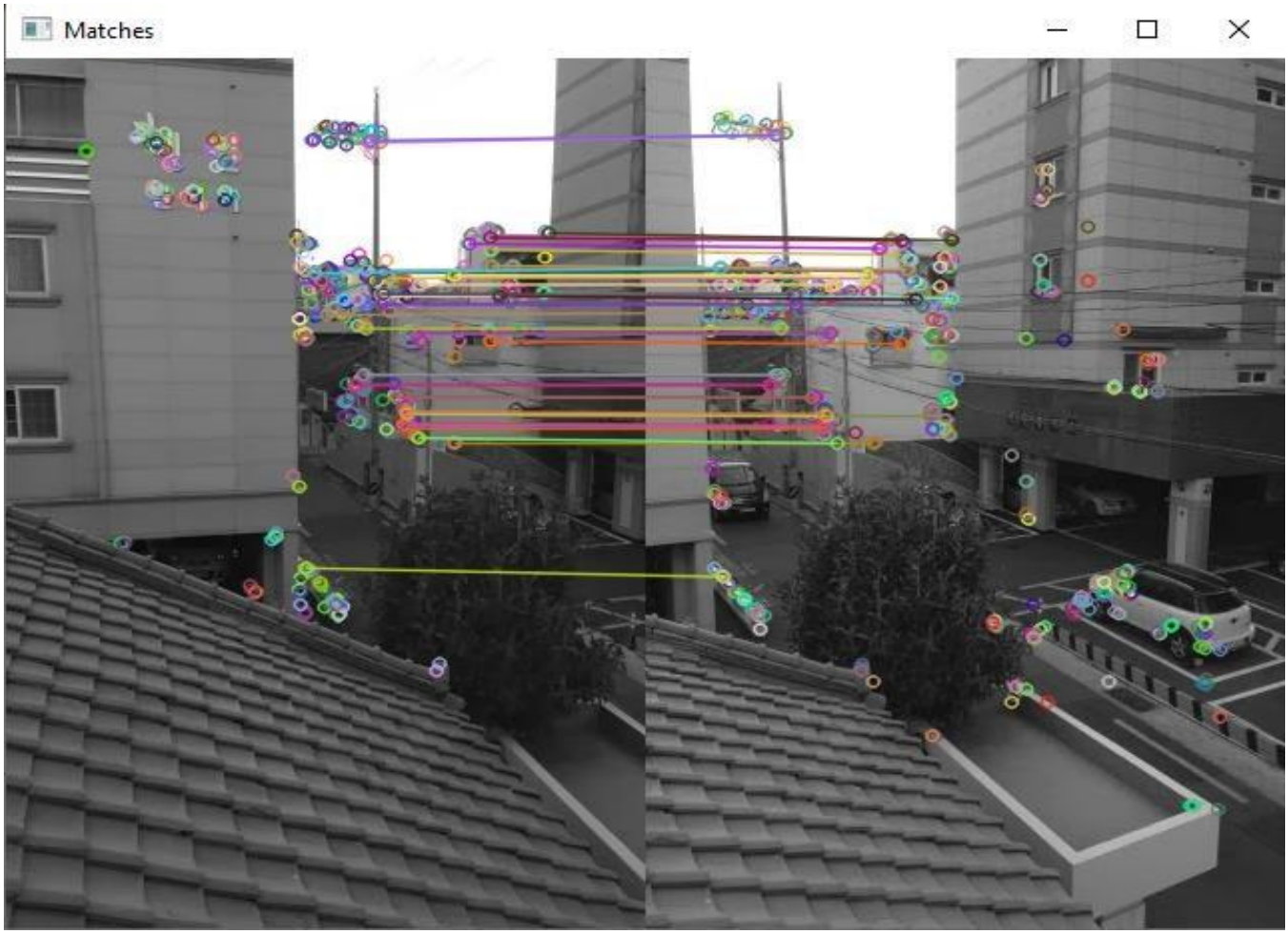
In OpenCV, a Brute-Force (BF) matcher or a FLANN-based matcher to match features between pictures. BF Matcher compares the descriptor of a feature in one image to the descriptors of all other features in another image and provides the match depending on the distance. It takes a long time because it compares all the qualities.

Main Feature Detection And Matching Detection Component:

Identify the Point of Interest

Description: Each feature point's local appearance is represented in a fashion that is (hopefully) invariant under changes in illumination, translation, scale, and in-plane rotation. For each feature point, we usually end up with a descriptor vector.

Matching: Descriptors are compared across photos to find features that are comparable. We can generate a collection of pairs (Xi, Yi) (Xi', Yi') for two photos, where (Xi, Yi) is a feature in one image and (Xi', Yi') is its corresponding feature in the other.


Matches

References

Image Filters in Python Manvir Sekhon, https://towardsdatascience.com/image-filters-inpython-26ee938e57d2

Introduction to SURF (Speeded-Up Robust Features), https://medium.com/databreach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e

Introduction to SIFT(Scale Invariant Feature Transform), https://medium.com/databreach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

Python | Corner detection with Harris Corner Detection method using OpenCV Last Updated: 21 Jan 2019. https://www.geeksforgeeks.org/python-corner-detection-with-harris-cornerdetection-method-using-opencv/

Sharpen-filter Using Pillow - The Python Image Processing Library, https://pythontic.com/image-processing/pillow/sharpen-filter

Appendix:

Code Length too long, click here for codes:

https://drive.google.com/drive/folders/1qEnTd_gpAXUuMU0g8YqsrlfQ93Owq4vr?usp=sharing