CMPT 361: Introduction to Visual Computing, Summer 2022

Assignment 2

Object/Face Detection using Harris Corners and SIFT-like Descriptors

301449735
Hanra Jeong

1. Prepare Data

Level 1. One image should contain just the single object/your face.

Level 2. The second image should contain the same object/face with some very different objects/faces.

Level 3. The third one should contain the same object/face with some other similar objects/faces, the object/face of interest should have some feature variations caused by changes in viewpoint/illumination/makeup etc

Level 4. The last image should be really challenging, containing the same object/face with significant changes in features, and similar objects/faces in the same image.

*All the images are taken by me, no reference.

Table 1. Prepared data

			
Level 1, Single object : Membership card	Level 2, Same object with different object: lotion	Level 3, Same object with stickers on it, and with similar objects: 2 cards	Level 4, Same object with stickers on it and some part of it is hidden under the black paper, and with similar objects: 2 cards *This is more difficult than level 3, because most of the information on card is hidden by black paper.
			
Level 1, Single object : jam jar	Level 2, Same object with different object : lotion	Level 3, Same object from the different view, and with the similar object: oil pet	Level 4, Same object with the opened lid, and the lid is placed in front of it, with the similar objects: peanut butter jam jar and long bottle *This is more difficult than level 3, because some part of it is covered by it's lid and there are more bottles.
			
Level 1, Single person : myself	Level 2, Me and the very different object: cotton candy	Level 3, Me with the bear hair band on my hair and with the similar face: my friend	Level 4, Me with the hair band on my hair and wearing mask, and with the similar face: my friend * It is more difficult than Level 3, because majority of my face is covered by mask

2. Detect Harris Corners

2.1. Implement a Harris corner detector with a few tuneable parameters. (1 mark)

Tuneable parameters : alpha value and threshold value

Threshold value can be changed by the input parameter of "Harris_detector" function, but alpha value is decided as 0.05 which is the best suitable value for my data sets and found by trial and error.

2.2 Analyze the performance of your algorithm, in terms of different parameter settings and the resulted number/quality of detected features. Support your analysis with qualitative data, i.e., visualize the corners on your images in the report. Then freeze your code with the "optimal" parameters you have found for later tasks. Please also write the matlab commands in the report that the TA can use to reproduce your results. (1 mark)

Dataset 1: Alpha value = 0.05, Threshold = 0.01

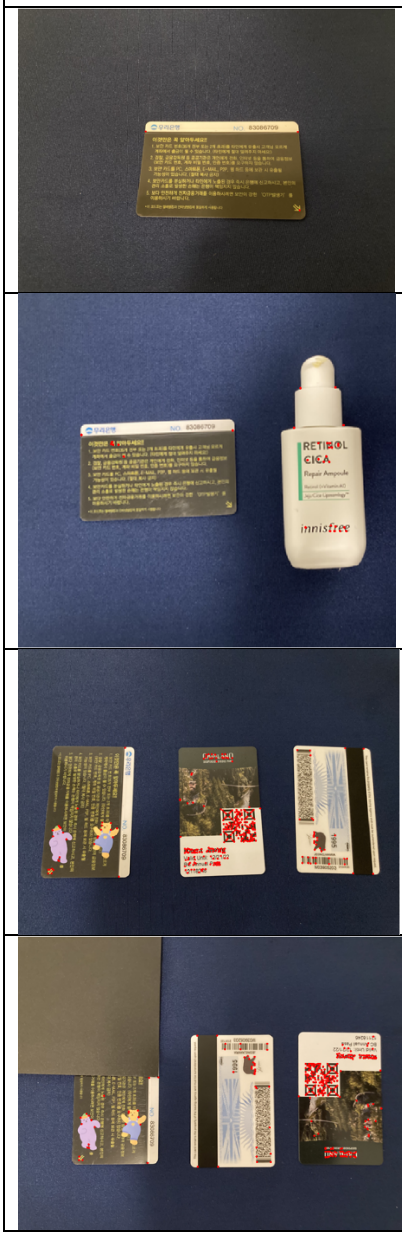
Dataset 2: Alpha value = 0.05, Threshold = 0.0001

Dataset 3: Alpha value = 0.2, Threshold = 0.0001

As the resulted data in Table 2 shows, as the threshold value gets smaller by comparing dataset1 and 2, there are more edges detected. On the other hand, as the alpha value gets bigger by comparing dataset2 and 3, less edges are detected.
Therefore, by considering these facts, the parameters are fixed as dataset2 : alpha value = 0.05 and threshold = 0.0001. These values are used throughout later tasks.

Harris corner detector is implemented in the file name "Harris_detector.m" and in order to run this code, you can run the code "Harris_runner.m" which will automatically generate and save the resulted images on the same folder.

Table 2. Resulted data with different parameters. Each dataset 1~3 is described on the text.

Dataset 1 Alpha value = 0.05, Threshold = 0.01	Dataset 2 Alpha value = 0.05, Threshold = 0.0001	Dataset 3 Alpha value = 0.2, Threshold = 0.0001
		


SMUCKER'S
PURE
Raspberry Jam
Confiture de framboises
innisfree
Sunscreen
Canola oil

3. Develop SIFT-like Descriptors and Feature Matching: (8 marks in total)

3.1. Develop SIFT-like descriptors for Harris Corners. That is, you don't need to implement SIFT-detectors. (4 marks)

Please refer to the Sift_detector.m

3.2. Implement a feature matching method that can match detected corners based on their SIFT-like descriptors. (2 marks)

Please refer to the match.m

3.3. Tune the parameters of your algorithms to achieve the best results you can get. Write in your report with supporting results: first with matched features visualized so that the TA can inspect them in a qualitative way, and then report quantitative metrics such as precision/recall/f-score. Please also write the matlab commands in the report so that the TA can reproduce your results. (2 marks)

According to the Lowe's paper [6], he said "Matches for which the nearest neighbor was a correct match have a PDF that is centered at a much lower ratio than that for incorrect matches. For our object recognition implementation, we reject all matches in which the distance ratio is greater than 0.8, which eliminates 90% of the false matches while discarding less than 5% of the correct matches.".

Therefore, for my code, I tried to tune the hyperparameter such that I can reject all matches in which the distance ratio is greater than this value to increase the accuracy of match.

You can run the code "Task3Runner.m" which will automatically generate and save the resulted images on the same folder.

Testcase 1: confidence < 0.97

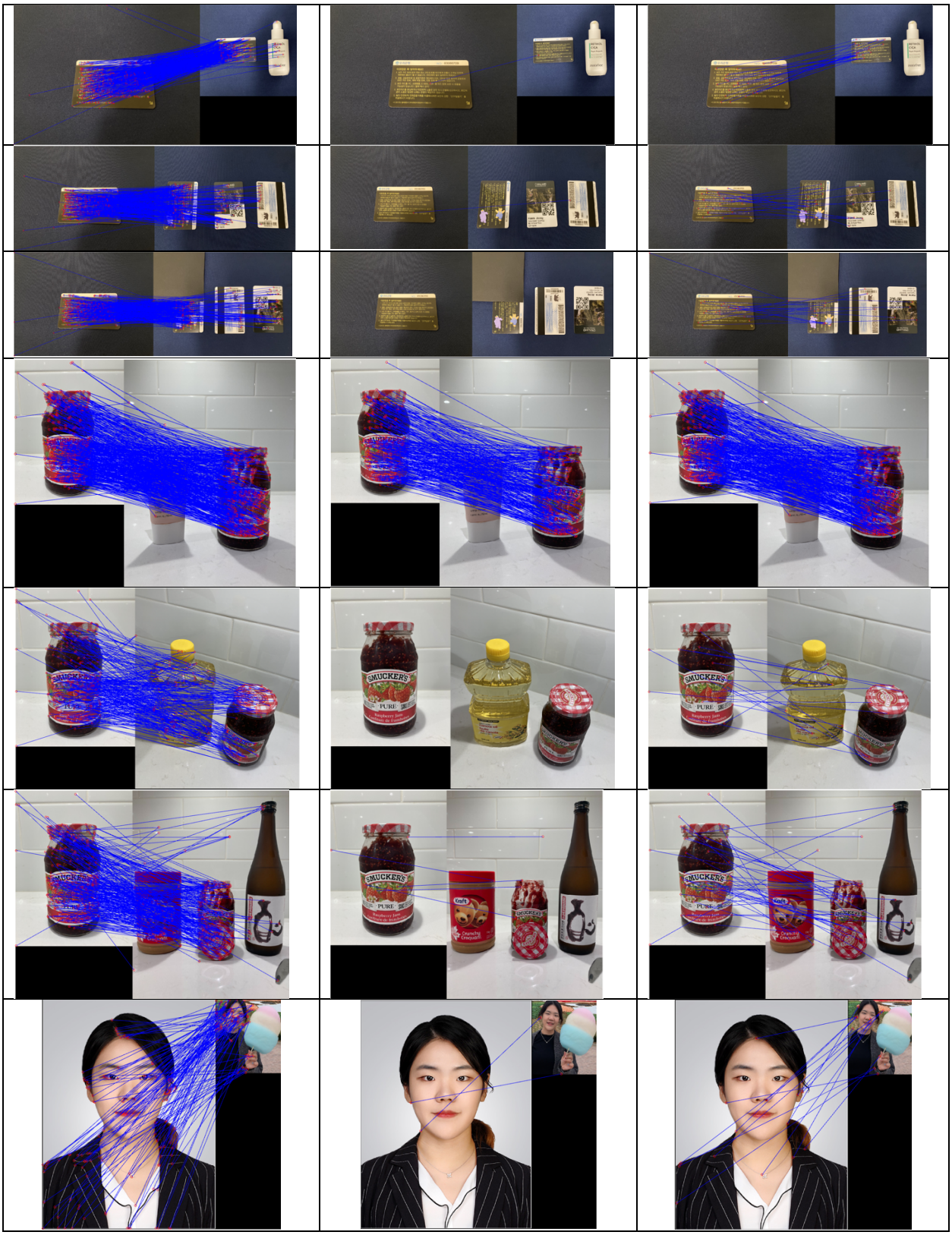
Testcase 2: confidence < 0.8

Testcase 3: confidence < 0.9

By doing this, as the Table 3 shows, to get the reliable amount of detected match points, the suitable confidence for my data is 0.97. The possible reason that my value is much higher than Lowe's is that to speed up the data computation, I minimize the image size, which cause the less detected Harris corner values and matching values from SIFT detector. Therefore, if I set the parameter to 0.8, it results the loss of nearly entire points for some images.

Table 3. Computed results for SIFT-like detector and match algorithms with various parameter values.

Testcase1. 0.97	Testcase 2. 0.8	Testcase 3. 0.9



4. Develop an object/face descriptor: (7 marks in total)

4.1. Based on the philosophy that you have learned from SIFT descriptors, design a basic higher-level descriptor that can help you detect the same type of objects or human faces, given an image of the object or a human face. E.g., detect all water bottles or detect all human faces in an image. (3 marks)

Please refer the detail to the Task41Runner.m.

General algorithm

1. Compute the Harris corner detection and get the coordinates
2. Using these coordinates, compute SIFT-like detector by using Sift_detector function from task 3, and finding the matching point by using match function from Task 3.
3. Clustering algorithm
    a. Sort the matching points' coordinates by x values
    b. Compute the distance of x-axis between (sorted) adjacent coordinates
    c. Making the group (clustering) by comparing the distance with parameter
    → By considering the size of image, the parameter for my data is 20
    d. Ignore the group with equal or less than 5 points (coordinates), to minimize the error and variance
    e. Draw the square box withing the clustered coordinates/ points to show the detected objects

4.2. Improve your algorithm so that your descriptor can detect the particular object/face among similar objects/faces. E.g., detect a particular water bottle in a few water bottles, detect your face in a family photo. (2 marks)

From the algorithm from 4.2, I counted the number of coordinates in each object, and then pick the object with the highest number of coordinates. This is because these coordinates are computed by match function, which means are the coordinates that are computed based on the input (original / target object). Therefore, the object with the high number of matched coordinates / points mean it is the object that we are looking for.

4.3. Describe your ideas/algorithms in your report, analyze their performance and discuss why or why not they work, with supporting qualitative analysis. Please also list the matlab commands so that the TA can verify your claims. (2 marks)

Please refer to the described algorithms from 4.1 and 4.2.

Task 4.1 is implemented in the file name Task41Runner.m and Task 4.2 is implemented in Task42Runner.m. You can run the codes which will automatically generate and save the resulted images on the same folder.

The performance of my algorithm and descriptors are quite good and accurate to detect the target objects as marked as red box in Table 4 with the input data. However, the accuracy can be varied by tuned parameters and input images.

Although object detection is all succeed, as the Figure 1 and 2 show, there are 2 failed for detecting the target cases. This happened because for the case 1, as the Figure 1 shows, due to the different view point, the detected data edges / points from jam is much less than the original image. On the other hand, this detecting method doesn't compare the color, therefore, the letters / text are recognized as the same, even if they are on the different objects.

Figure 1. Failed case with jam

For the case 2, as the Figure 2 shows, the front face is recognized with more edges compared to the face behind. Therefore, for this data image, I was at the behind where the other person is front of me. This happens due to the variation of light and auto-filter system by my camera.

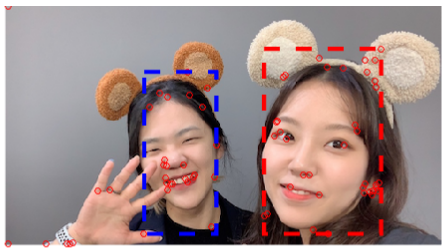
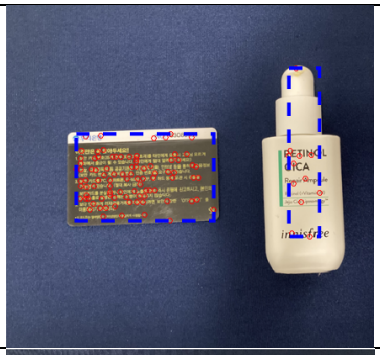
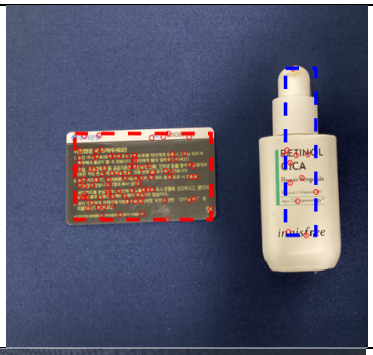


Figure 2. Failed case with my face

The reason why the matching coordinates are used rather than the coordinates computed from Harris corner to detect the object and find the target is because by doing is because, the coordinates computed from Harris corner include a lot of noises. On top of that, data set 1 (card) and data set 2 (jam) are taken with the objects that are separated with each other horizontally, but it is difficult to find and abnormal to take human pictures that are totally separated with each other horizontally. Therefore, the detected coordinates from the clothes or any other accessories often mixed between more than two people, causing the low quality of object detection. Therefore, I used the matching coordinates.

To increase the accuracy and generalize the algorithm, the known clustering algorithm such as hierarchy clustering or K-means clustering algorithm can be used to detect objects. Moreover, more higher resolution of images will also help to increase accuracy.

Table 4. Computed results for task 4.1 and 4.2

Target object	4.1	4.2
		
		
		
		

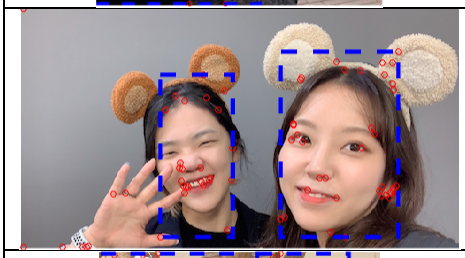

SMUCKER'S
PURE
Raspberry Jam
Confiture de framboises



Kraft

Kraft






Reference.

[1] Yin, K. (n.d.). Cmpt 361 Matlab Lab 1. Lecture.

[2] Yin, K. (n.d.). CMPT 361 MATLAB Tutorial. Lecture.

[3] Yin, K. (n.d.). CMPT 361 SIFT. Lecture.

[4] Yin, K. (n.d.). CMPT 361 Corner Detection. Lecture.

[5] Yin, K. (n.d.). CMPT 361 Feature Matching. Lecture.

[6] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2), 91–110. https://doi.org/10.1023/b:visi.0000029664.99615.94

[7] Computer Vision Project. (n.d.). Www.cc.gatech.edu. Retrieved July 10, 2022, from https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj2/html/mbalusu3/index.html

[8] Szeliski, R. (2020). COMPUTER VISION : algorithms and applications. Springer Nature.

** Details of reference for the code is cited in the code as the comment, but not here.*