CS 763/764: Computer Vision

Answers A

1.2.1 Questions on Synthetic Images

Question 1. For fixed nPoints list the valid combinations possible. In each case,
(a) Mention the parameter values used in the detectors to achieve the desired number of detected points.
(b) Mention average computation time in seconds (*.xx)

Solution.

nPoints = 300.

(a) For below possible combinations which included FAST as a detector, we used **cv2.ORB_create()** to implement the detector. We specified the nfeatures to be equal to 300 for first iteration and 1000 for next iteration (and picked the best **nm** matches). The other parameters like scaleFactor, nlevels, edgeThreshold, firstLevel, WTA_K, scoreType, patchSize, fastThreshold were set to default as they gave good results. (1.2f, 8, 31, 0, 2, HARRIS_SCORE, 31,20 respectively).

- fast-brief-scale
- fast-brief-view
- fast-brief-rot
- fast-brief-light

- fast-sift-scale
- fast-sift-view
- fast-sift-rot
- fast-sift-light

For below possible combinations which included DoG as a detector, **cv2.xfeatures2d.SIFT_create()** was used to implement the detector. We specified the nfeatures to be equal to 300 for first iteration and 1000 for next iteration (and picked the best **nm** matches). The other parameters like nOctaveLayers, contrastThreshold, edgeThreshold, sigma were set to default as they gave good results. (3, 0.04, 10 and 1.6 respectively).

- dog-sift-scale
- dog-sift-view
- dog-sift-rot
- dog-sift-light

- dog-brief-scale
- dog-brief-view
- dog-brief-rot
- dog-brief-light

(b) The average computation for each of the above case is shown in the following table. The code for the same is provided in the convincing directory. We fix the nPoints to 300 and compute the average time taken by each of the 16 combinations listed in the solution of above question, after running each case for 10 iterations.

Combination	Average Time (ms)
fast-brief-scale	190.41380882263184
fast-brief-rot	52.61228084564209
fast-brief-view	31.37805461883545
fast-brief-light	26.685762405395508
fast-sift-scale	164.8270606994629
fast-sift-rot	236.81409358978271
fast-sift-view	179.75962162017822
fast-sift-light	167.7939176559448
dog-brief-scale	158.4420919418335
dog-brief-rot	369.5844888687134
dog-brief-view	197.1574068069458
dog-brief-light	199.17142391204834
dog-sift-scale	291.7490482330322
dog-sift-rot	629.3117046356201
dog-sift-view	371.9486713409424
dog-sift-light	374.8852252960205

From above table, it is clear that fast detector and brief descriptor in tandem work much faster than any other combination consisting sift influence. Average performance of the detector-descriptor pair is summarized below.

keypoint-descriptor	Average time (ms) across all test images
fast-brief	75.272
fast-sift	187.290
dog-brief	231.088
dog-sift	416.967

Question 2. Mention the specs and OpenCv version of your machine used in the previous step to report performance.

Solution.
OpenCv version : 3.4.5.20
Machine specs :
Processor Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
Installed RAM 8.00 GB (7.87 GB usable)
System type 64-bit operating system, x64-based processor

Question 3. For (top) matches=5 will 5 lines be visually seen? Explain.

Solution.
In the brute force matching, we have implemented the top matches among the keypoints by sorting the matches on basis of distance and then picking the smallest **nm** matches. So when keypoints are much greater than **nm**, (Say top 50 among 300), we can get all the 50 lines and can visually see them. But say if keypoints and **nm** are equal and we chose them to be 5, here in this particular we may/may not see actually 5 lines. There are couple of reasons for this, firstly, the detectors when initialized with nfeature=keypoints, and other parameters set to default, they may or may not find all the specified number of keypoints in order to achieve the default parameters. Secondly in the function *cv2.BFMatcher()*, we pass an arguement called **crosscheck**, which is a boolean variable, this variable when True, will check if minimum distance of keypoint 1 in image 1 from keypoint 1' in image 2 is d, then the minimum distance of keypoint 1' in image 2 from keypoint 1 in image 1 should also be d, i.e. both should be nearest neighbour of each other.

Question 4. Provide an image (dog, sift, ?) named incorrect.png which shows 3 incorrect matches. Number and highlight these, and explain your answer. (You can use any image annotation tool of your choice but each group member will do one each, and name the tool used). **Solution.**
For this task, the three images we have used are :

- dog-sift-scale
- dog-sift-rot
- dog-sift-light

Explanation : In order to get the images with incorrect matches, we detected 20 keypoints and used all the brute force matches that were available rather than selecting top matches. As evident from the images, this gives rise to various incorrect feature detection as indicated. A common thing that we can notice among all these incorrect points is that they seem similar when visualized on a smaller scale, for example, point 2 annotated in first image maps to a point which has same type of background (a black background). Thus the matching is not directly arbitrary, but does have **some** commonness. For annotation, we used paint tool.

FIGURE 1. dog-sift-scale original

FIGURE 2. dog-sift-scale annotated

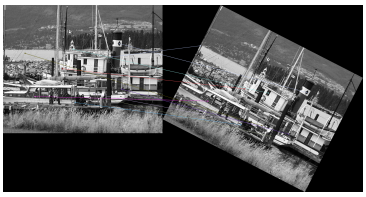

FIGURE 3. dog-sift-rot original

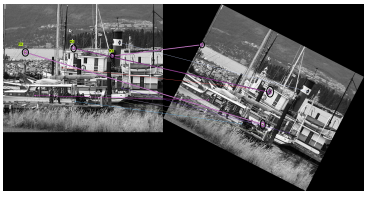

FIGURE 4. dog-sift-rot annotated

FIGURE 5. dog-sift-light original

FIGURE 6. dog-sift-light annotated

Question 5. When using FAST interest points, which descriptor shows better performance in matching interest points on an average (across all the pairs)?
(a) What can be an advantage of the inferior descriptor method over the superior?
(b) In which of the 4 transformation the difference in performance between the superior and inferior descriptors is the maximum and why?

Solution.
The FAST interest points, work best when used with BRIEF descriptor on an average across all the pairs in matching the interest point as well as in taking the minimum time. Minimum can be noticed from the solution of question 2 and for performance related to matching the interest points, we can look at the result and FAST with BRIEF gives matching of more distributed points across the image.
(a) The advantage of using an inferior descriptor can be that we can cross check if there are few keypoints that we might have missed in the superior descriptor. Also a hybrid implementation can sometimes bring out the advantages of both the different methods (used in detector and descriptor). Example : fast-sift for rot performs better than fast-brief for rot
(b) The difference is maximum in case of **rot** transformation. The reasons that we could think of is that the image used for **rot** transformation have potentially a lot of feature points that can be mismatched.

1.3.1 Questions

Question 1. Explain your choice of (detector, descriptor).

Solution.

The (detector, descriptor) pair we used is (SURF, SURF). Our main reason for using them was that unlike sift and orb, the object of SURF class does not take nfeatures as an input, rather it takes thresholds, and other arguments such as extended (decides whether we want a detailed descriptor or not), also upright (if we want it to be rotation invariant or not). SURF, unlike SIFT, does faster computations, also it detects more keypoints, there by adds more description to the descriptors to achieve faster computations.

Question 2. Write a short note (two paragraphs at least) on the observations and inferences.

Solution.
Using this detector and descriptor pair, First thing we observed is that the detector is not taking the number of features as input like the other detectors we used instead it is taking the Hessian Threshold as input and giving all the features as output that have a hessian greater that the threshold. In this detector to control the number of features to detect we need to change the hessian threshold. As the value of hessian threshold increases, the number of detected features reduces.

If we have a look at the output generated after matching the detected features of the source and transformed image, we observe that this feature, detector pair gives correct matches for rotation and scale transformation and the reason for this that we think is the detection of more number of features using less hessian threshold. However, this feature- detector pair doesn't perform well when the source and target images are taken under different lightening or under different view this is because of the approximations made to make the algorithm faster.

Question 3. Let's assume we are stuck with a detector and a descriptor given to us. What is under our control is the matcher, and assume we are concerned with accuracy. Suggest heuristics that can be used to prune out 'false' matches.

Solution.

To prune out the false matches we can go by following two methods, one of which we have implemented in our solution itself. The detailed description about these methods is depicted below.

1) ***Fixed number of points, variable threshold*** : After finding the descriptors, we use the brute force matcher, and this brute force matcher will give us the corresponding distances for each possible match (matches will be less than the number of keypoints if we set the crosscheck flag in the brute force matcher to be true). Now we sort these keypoints on the basis of their distances in ascending order. Thus, now we can pick nm number of keypoints as our features using the list slicing operation.
2) ***Fixed threshold, variable number of points*** : Secondly, the output of brute force matcher can be sliced such that we only take the points lying under a particular distance threshold. For example if we take the threshold to be equal to median of the list containing the distances of keypoints (the output of the brute force matcher), then we can select only those matches where the distance is less than the median. By considering the median in place of mean, we make sure that we can remove as many outliers as we can.