CSC391 project2 report

Alex Zhang

Oct 2023

Part1

Rotation

In every case I tried to rotate image to 270 degree clockwise.
For SIFT, I think the result shows that SIFT is rotation invariant after drawing the matches between two images. The keypoints in bed sheet and human face can match with each other and the result shows that the accuracy is around 65%. The result matches textbook's result.

For FAST, in this case, the number of keypoints is already different. After trying to create matching, I will say that FAST is not rotation invariant. I have to increase the threshold to make my code consider good matches.

For Harris Corner Detection, there is only few matches. I read from some online resources said that Harris should be rotation invariant, so I check whether it will match without any rotation. If there is no rotation, all keypoints are matched, but if there is a 270 rotation, really a few matched. One probability is that there seems to be some identical corners like wood corners on both windows, It may confuse the detection. Second probability is that my hair seems to be smooth, it is hard to draw corner from harris detection and the descriptor may also confused.

Using ORB seems to create many matches even if I rotate 270 degree. It has better accuracy than using SIFT. One explanation for this is that when rotation degree is proportional to 90, ORB outperforms SIFT because the features for ORB usually stick to the center of image. This will make feature matching easier because the position of features will always be vertical or horizontal.

Scale

To test the scale invariant, I first shrink the image to 0.6 of the original.
For SIFT, I think it can still draw many matches with given keypoints, but compare to the original image, it still has many dismatches. I consider SIFT as scale invariant because the accuracy is about 33%. The result in textbook is about 40% One reason I think when SIFT didn't perform well when shrinking the image than expanding image is that, the number of keypoins decreases a lot when shrinking. The percentage will decrease.

I think FAST is scale invariant. In my image, to say whether FAST is robust on scaling depends on threshold. In my case, there are lots of keypoints missing after I shrinked the image. This will decrease the accuracy for finding matching features.
For Harris corner detection, I think it also didn't do well if I try to shrink the image. We can see that there are many dismatches if I only took part of matches. Also the for the features that seems to be matched, my setting of threshold didn't count most of it because they are not close enough to be accepted

by threshold. I can say Harris is not scale invariant.

For ORB, I think it is scale invariant and it has even better result than SIFT. One reason for this is that ORB has fewer features than SIFT and most of the features lie on that stripe pillow. After shrinked the image, features on stripe pillow still preserved which I think this is the main reason of having better result than SIFT.

Low-light illumination

I tried to make testing image to decrease the illumination.
For SIFT, the result indicates that SIFT is not really robust on low-light illumination. Since there is no scaling or rotation, if we see a parallel line and the destination is the same, it is a good match. Based on the matching feature, there are not so many parallel lines.

For FAST, I saw many parallel lines. I still think many features matches with each other though there are some mismatches. FAST should be robust to low-light illumination.
For Harris, I think it is doing fine for low-light illumination. There are many matches after I decrease the illumination. I think the change for gradient on corner after decreasing light can still be detected for Harris. So Harris corner detection should be robust for low-light illumination.

For ORB, It is hard to say ORB is robust in low-light illumination or not. It is true that there are some parallel lines, but there are also many intersections. Based on my observation, it has similar performance with SIFT which is also what textbook shows. I think ORB can be robust for low-light illumination but not good as FAST and Harris in this image.

Part2

SIFT

The following image is using SIFT as detector and descriptor.

Figure 1: SIFT

BRISK

the following image is using BRISK as detector and descripor.

Figure 2: BRISK

Harris

The following image is using Harris corner as dectector and SIFT as descriptor.

Figure 3: Harris

ORB

The following image is using ORB as detctor asn descriptor.

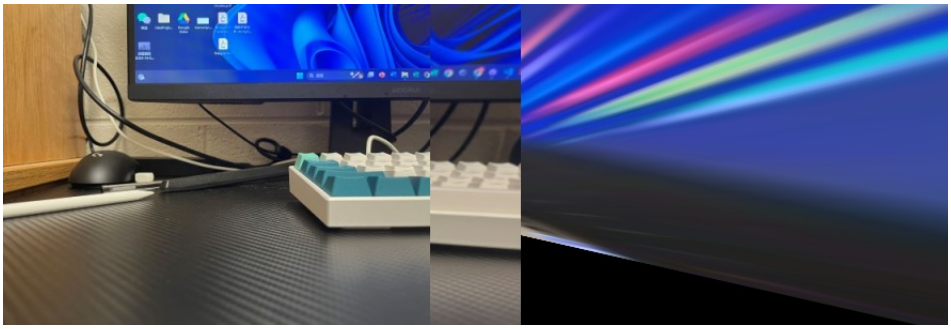

Figure 4: ORB

Result

From my observation, I think using Harris as detector and SIFT as descriptor has the best stitching. SIFT as detector and descriptor is the follow ones. ORB has the worst performace because I can hardly tell what object is in the last chunk.

I'm surprised that Harris actually is the best among four. SIFT, ORB,and BRISK all are well developed feature matcher, and Harris is just a corner detector and it even used SIFT to find descriptor. I thought SIFT algorithm will at least be better than Harris.

Also for ORB, I don't quite get it why it doesn't perform very well in this set of images. I knew that opencv actually uses ORB in its stitcher class, indicating it should be kind of powerful. One potential reason I think is that ORB needs more like parameters than other three. I even changed the threshold for good matches for ORB or it will not give my any output. Another one is that ORB runs way faster than SIFT, probably this is also one reason why opencv chose it.

To improve the quality, I think there are few ways. The first one is trying combining different detector and descriptor (Like Harris-SIFT). It may have better quality. The second case is trying to rotate, scale, or increase the illumination for images before doing stitching. SIFT didn't do well in low-light case, probably increasing the light will help SIFT better find matches. Or analyze the image and extract some property (like light, contrast), and apply detector that has best match in this case(FAST). It is even possible to do multiple runs on different feature matcher. Find the overlap between these matches and get the best stitched image.

Part3

One descriptor I learnt about is BRIEF(Binary Robust Independent Elementary Features). Basically, BRIEF stores feature descriptor in a binary encoding way. This can really reduce the storage requirement for keypoints descriptor. It can also possible increase the speed for generating descriptor. I learnt from opencv document that SIFT creates its keypoints descriptor by dividing sub-blocks from its neighbourhood and creating bin orientation histogram. total 128 bin value store as vector. This will takes a lots of space.
BRIEF introduced pairwise intensity comparison based on image patches. For example, in an image patch, first doing smoothing by applying Gaussian kernel, and then choosing n-pairs pixel (X,Y). After choosing, define a test on intensity of X and Y, if I(Y) is bigger than I(X), then its result is 1, else it will be 0. The author in BRIEF's paper actually mentioned 5 different ways to choose n-pair, and they ended up choosing one isotropic Gaussian Distribution. After conducting this method, they did some further test which I observe that the speed for using BRIEF as an descriptor is much faster than using SIFT. The computation time for descriptor is about 30 times faster overall.
For the recognition rate, BRIEF did really well compare to SIFT if we blur the image. Usually it has about 70% of accuracy but SIFT only has around 50% to 40%. However, we can further observe that BRIEF is not rotation invariant because once the rotation degree exceeds 30, the accuracy dropped to around 0.
In the paper, they use SURF as detector and then BRIEF as descriptor to do feature matching. To further test how fast BRIEF can do in feature matching, I wrote some codes to measure time doing SIFT for detector and descriptor, and the time for SIFT as detector and BRIEF as descriptor. It turns out for a really small image, BRIEF always runs faster than SIFT (0.05 second compared to 0.09 second). We can even increase the time by replacing FAST as detector instead of using SIFT(0.02 second this time).

For application of feature detection, I ask this question to chatGPT, and I found one application for

feature detection is medical imaging. There should be some really complicated algorithm for feature matching that identify special structures like blood vessels and tumors. People need to have an algorithm that is scale invariant, rotation invariant, and light invariant because some anatomical landmarks have different sizes, in different positions which requires an accurate algorithm to identify. Usually this involves many usage of different detection methods and deep learning is frequently used.