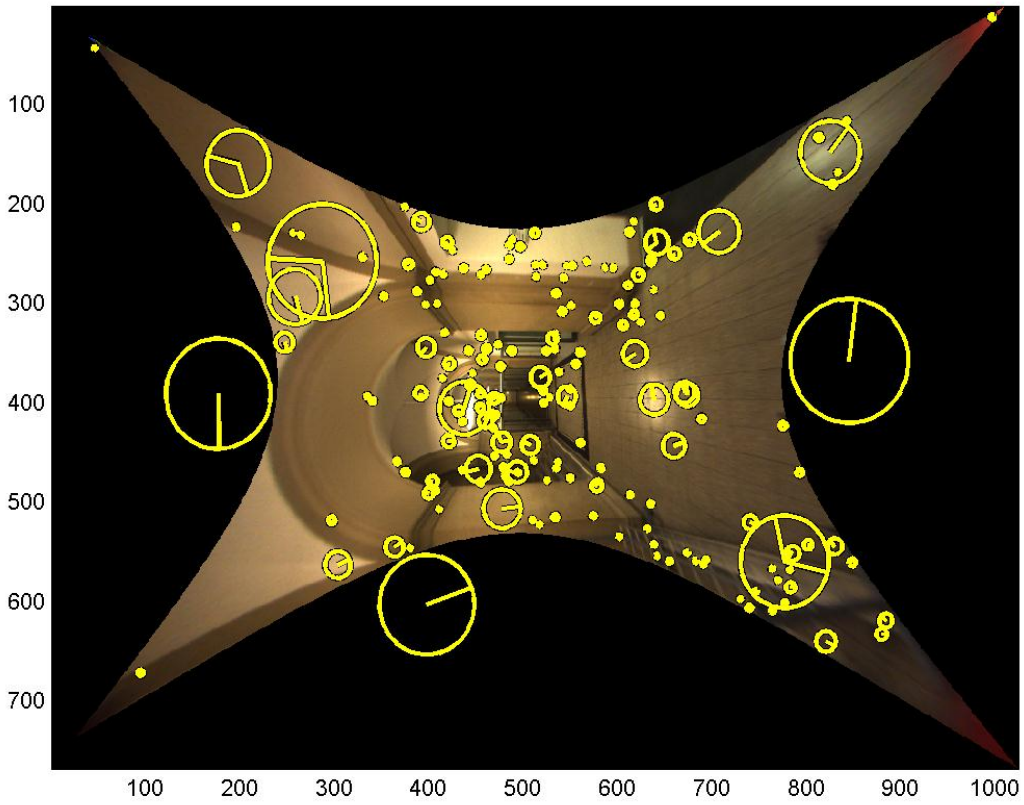
Computer Vision
Exercise 4

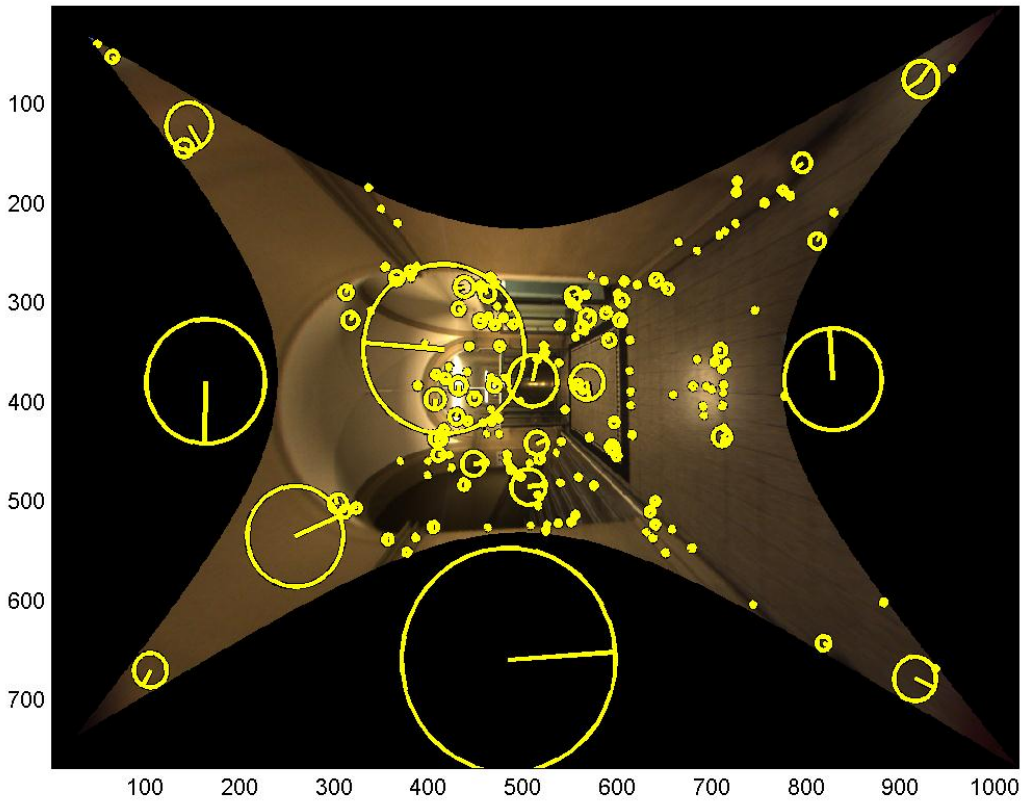
Amrollah Seifoddini

I iterated over the range of 1 to 12 for threshold in all the algorithms implemented. The figures are saved for every case, but I decided to not attach them all to this report. So, only some samples for threshold 5 demonstrated. You can run the code to generate all figures in less than 5 minutes. The graphs showing changes of inlier_count and iteration_count for the whole experiment extracted and also saved as mat and fig files in the source folder. The discussion over the graphs is at the end of this report.

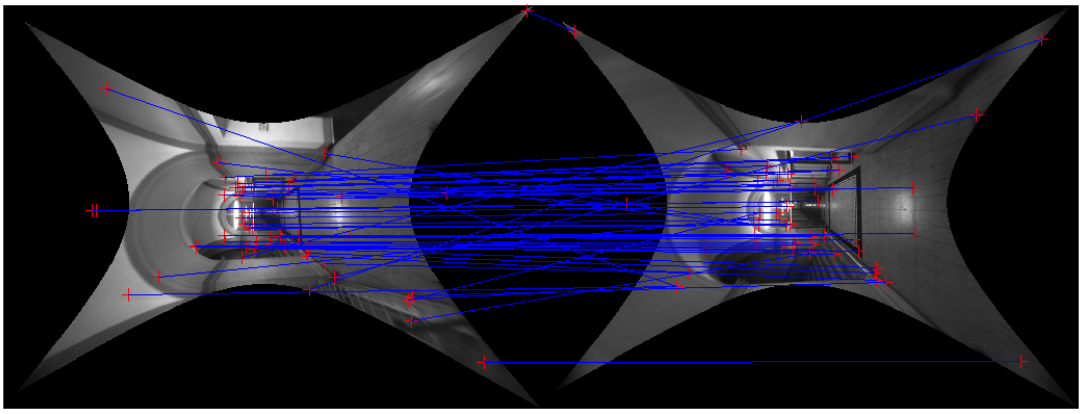
Part 1)

In this part, I used vl_sift library for extracting sift feature out of images. The results are shown below:





Then, the matching function of vl_sift called on these points to find the correspondances. Result is here:

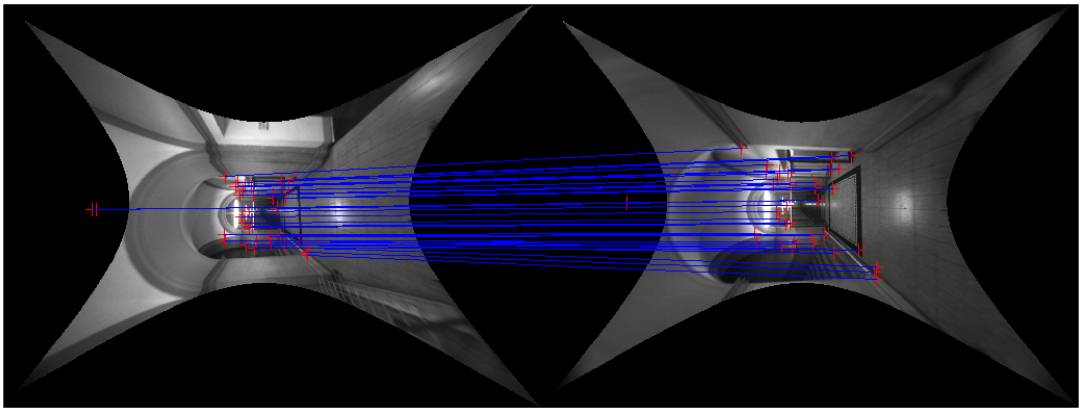


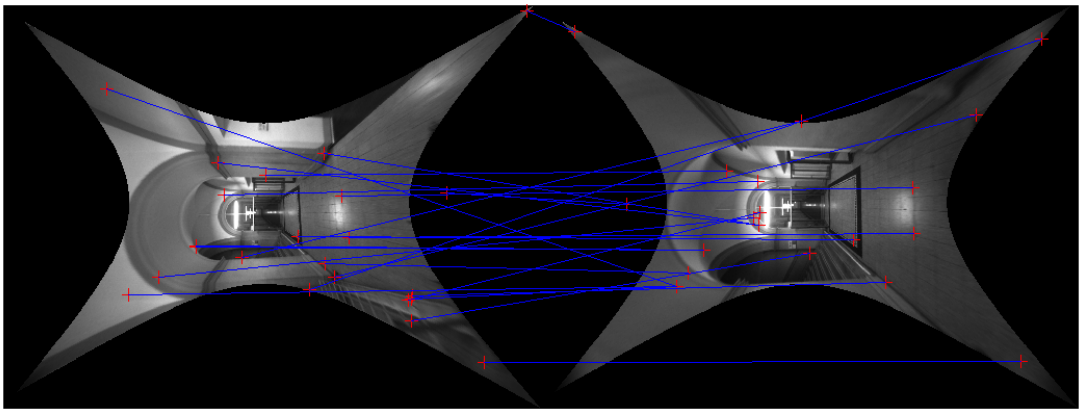
Part 2)

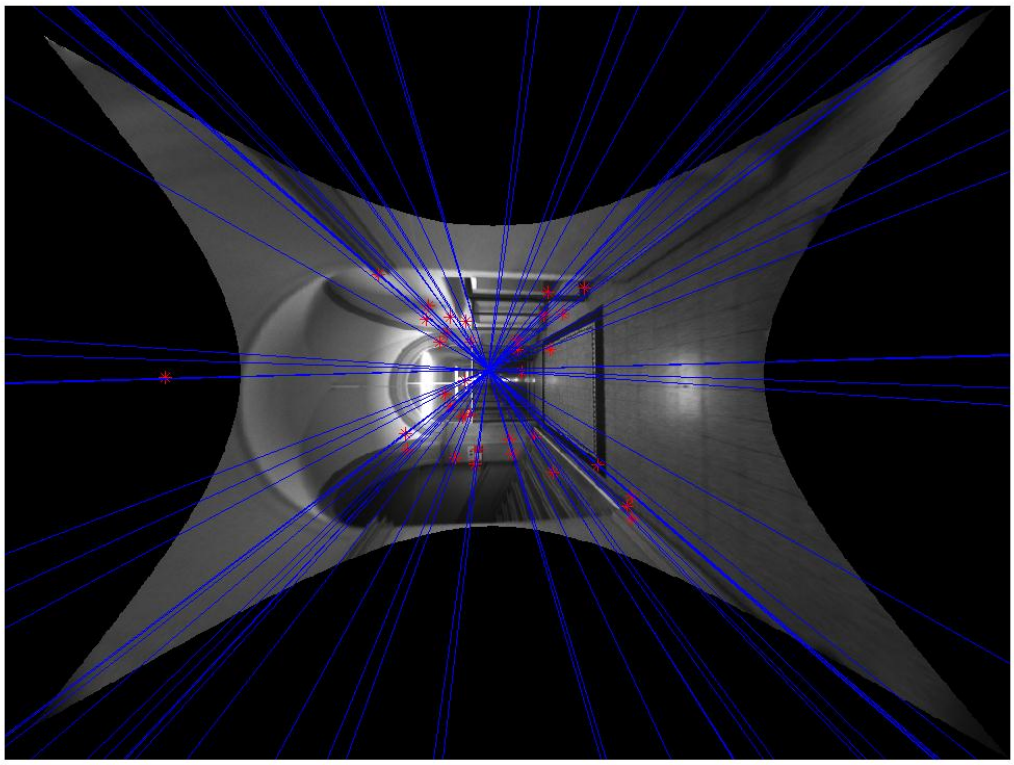
2.a)

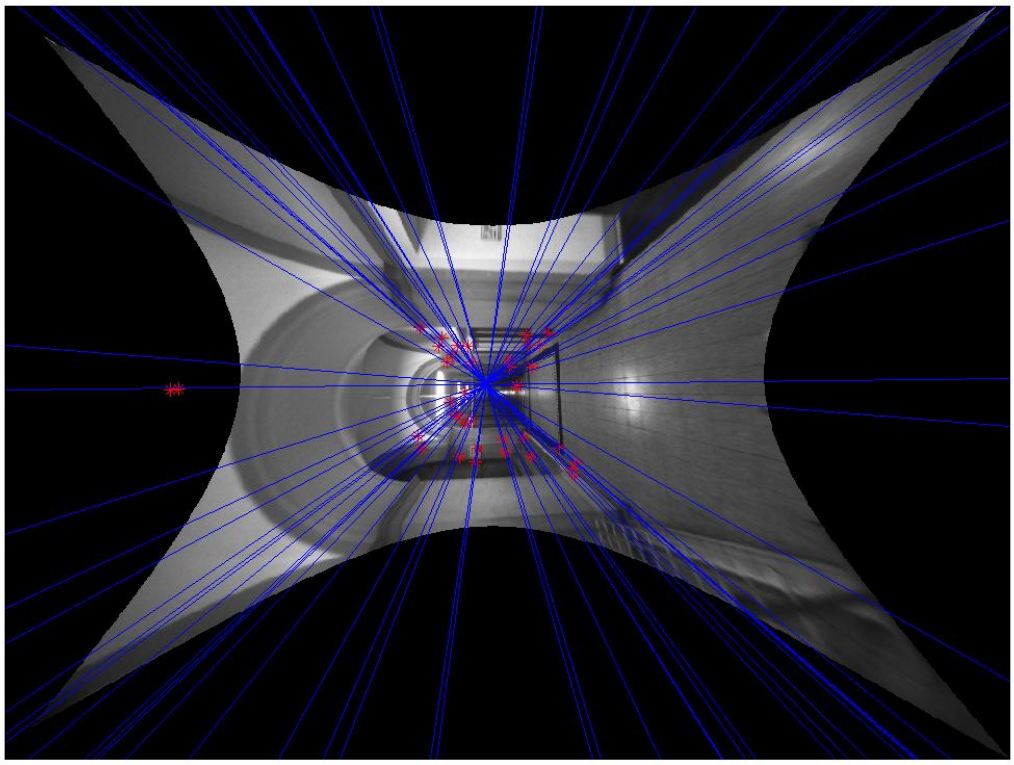
In this part, I implemented RANSAC algorithm with perpendicular distances. The shortest distance from a point to a line is computed with this formula: |ax0+by0+c|/sqrt(a^2 + b^2)

The iteration is fixed to 1000 for this part. I used my 8-point algorithm from exercise 2 in this RANSAC implementation to calculate the fundamental matrix given the correspondence points. The function is named fundamentalMatrix. I also used drawEpipolarLines from exercise 2 for showing the epiploar lines in images. The normalizePoints2d function from that exercise is used for normalizing the points whenever needed. One sample result of threshold 5 is shown below: (respectively:: inliers, outliers, epipolar lines img1, epipolar lines img2)









2.b)

Then, I changed the code from 2.a part to adaptively calculate iteration count based on desire Confidence rate, sample counts and current best_inlier_count in every iteration. Of course, I updated best_inlier_count and errors in every iteration. With this, I am sure that when we reach our desired confident rate, the loop with break in next iteration. So, we don't do any extra step. For initialization I put the 3 for iteration counter. The graph for iteration_counts of all implemented algorithms are shown at the end of this report. We can see that in almost every cases we can reach 0.99 confidence with less than 400 iteration.

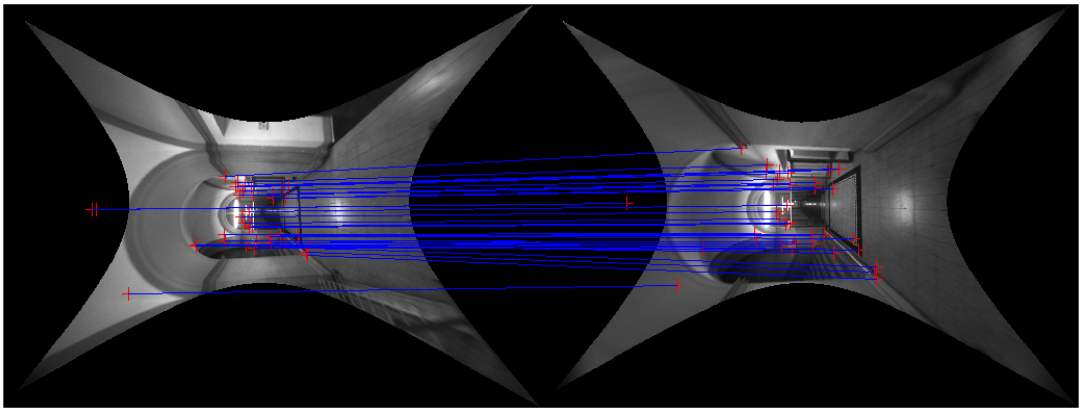
2.c)

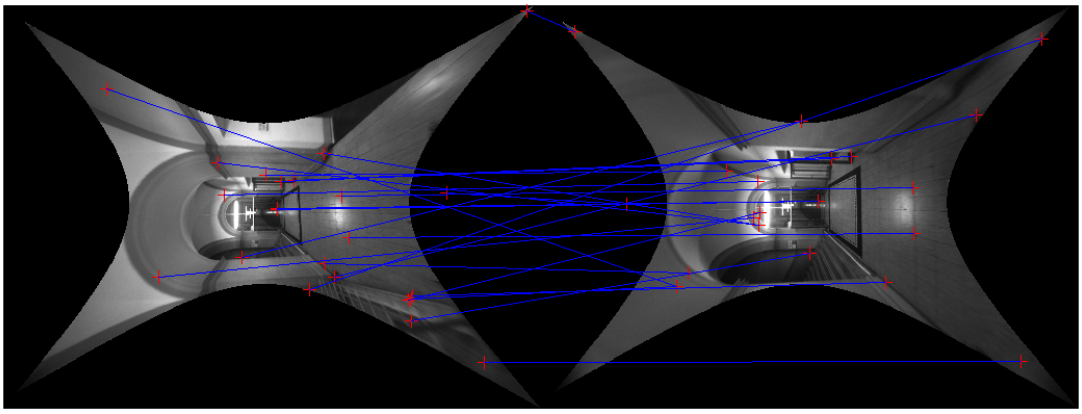
In this part, I replaced perpendicular distances by Sampson distances in 2.b code. I used the formula in exercise sheet to implement that. I observed that this distance is more robust to threshold variation.

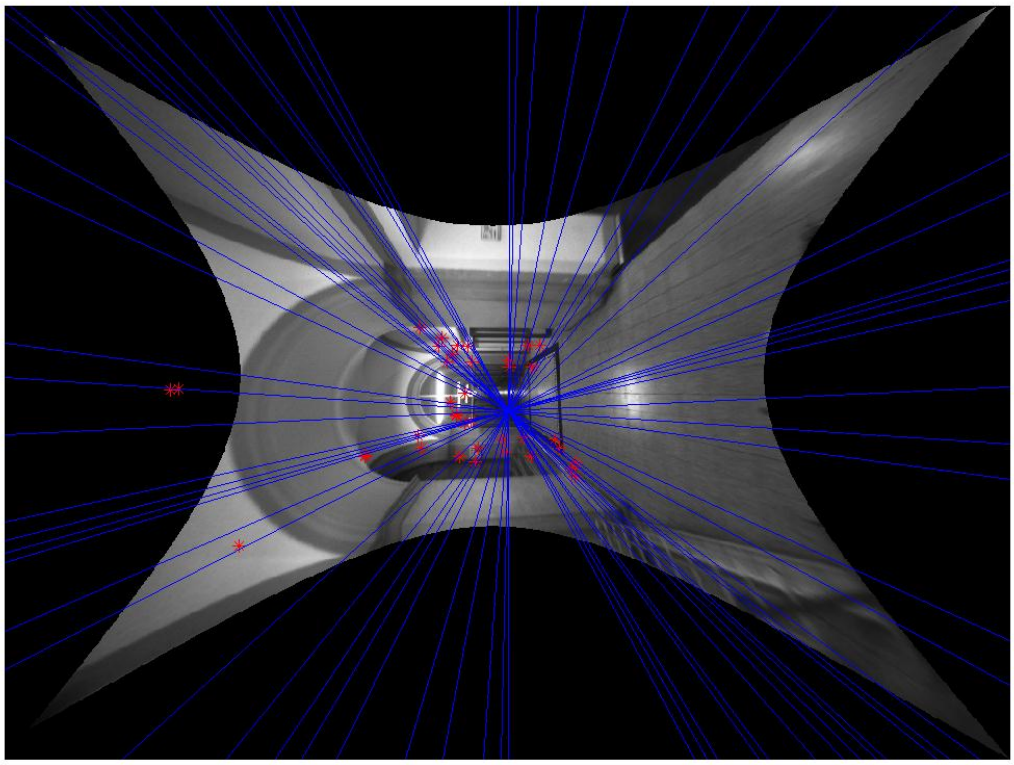
Part 3)

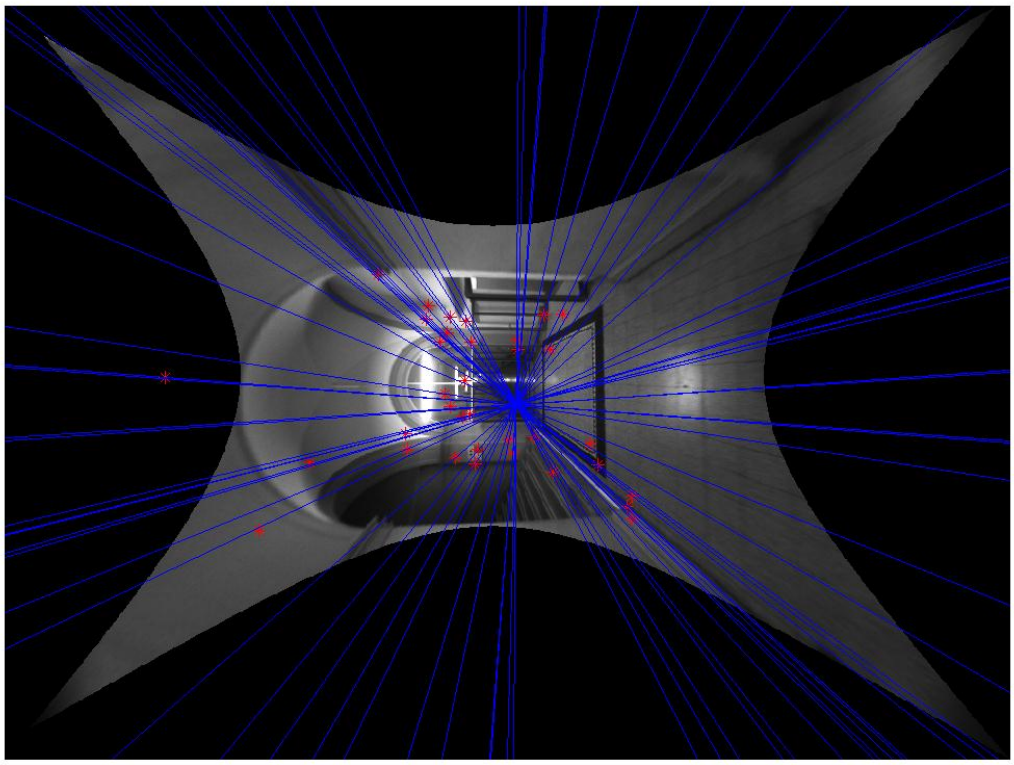
In this part, I used the code from 2.c but instead of fundamental matrix, I used the code by Nist´er and Stewenius which was provided as Mex files to compute probable essential matrixes correspondent to 5 matching points. Then I iterated over these essential matrixes which are 10 at most, and computed

fundamental matrix given that essential matrix and K (camera intrinsic matrix). The next steps are similar to 2.c. One sample result of threshold 5 is shown below: (respectively:: inliers, outliers, epipolar lines img1, epipolar lines img2)



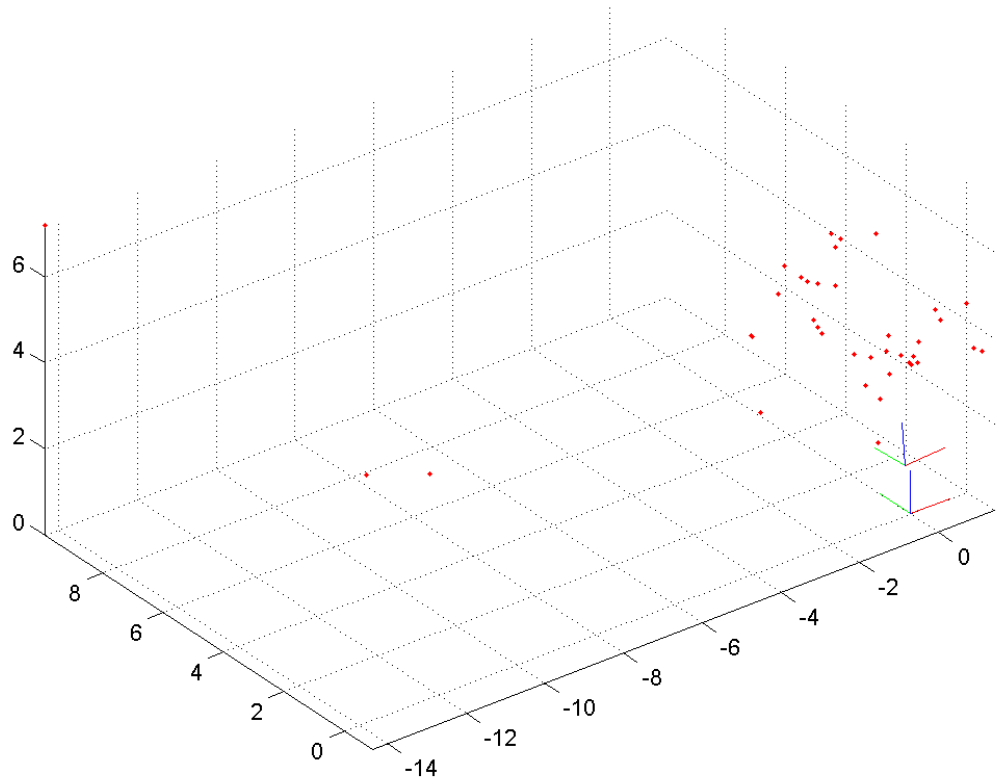






Part 4)

In this part, I used my code from exercise 2 (plus some of TA's guides in the feedback) to show all inlier points and camera poses in a 3D image. For extracting camera poses I used decompose function from exercise 2 and for computing the 3D coordinates of points, I used linearTriangulation function from exercise 2. A sample result is shown here.



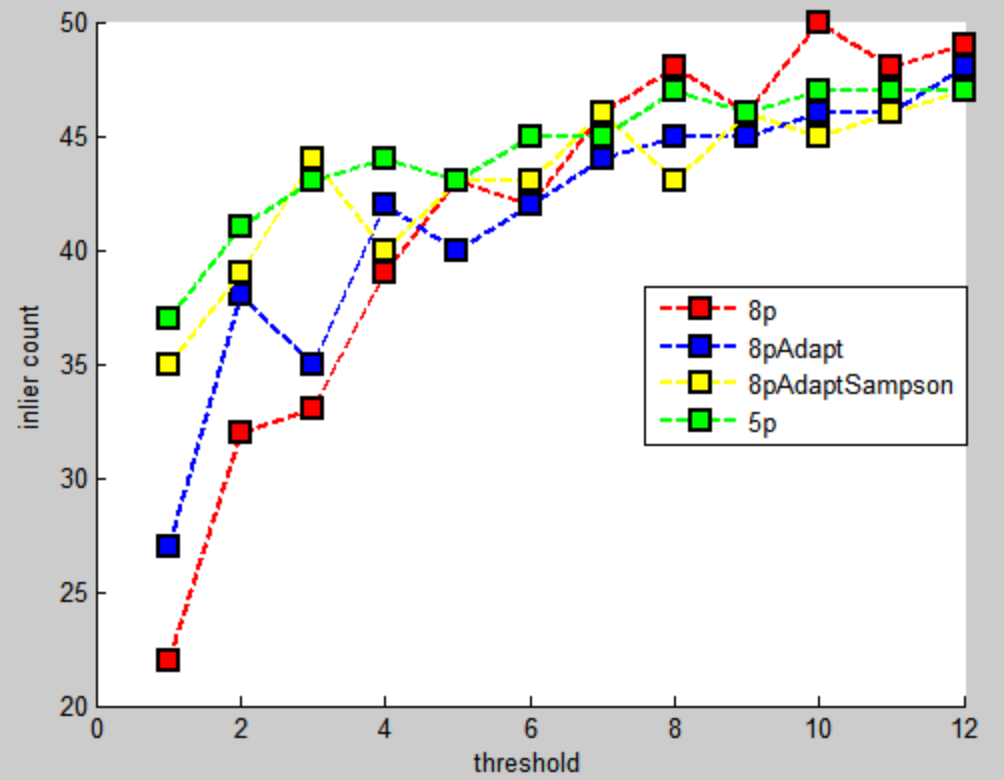
Discussion:

Simpson distances are less sensible on threshold. Small increase in inlier ratio observed for relatively large threshold changes. In small threshold changes, the behavior is un-deterministic and highly variant on data points. In last case, (5point algo) the changes are higher and more meaningful, probably because we know the camera matrix.

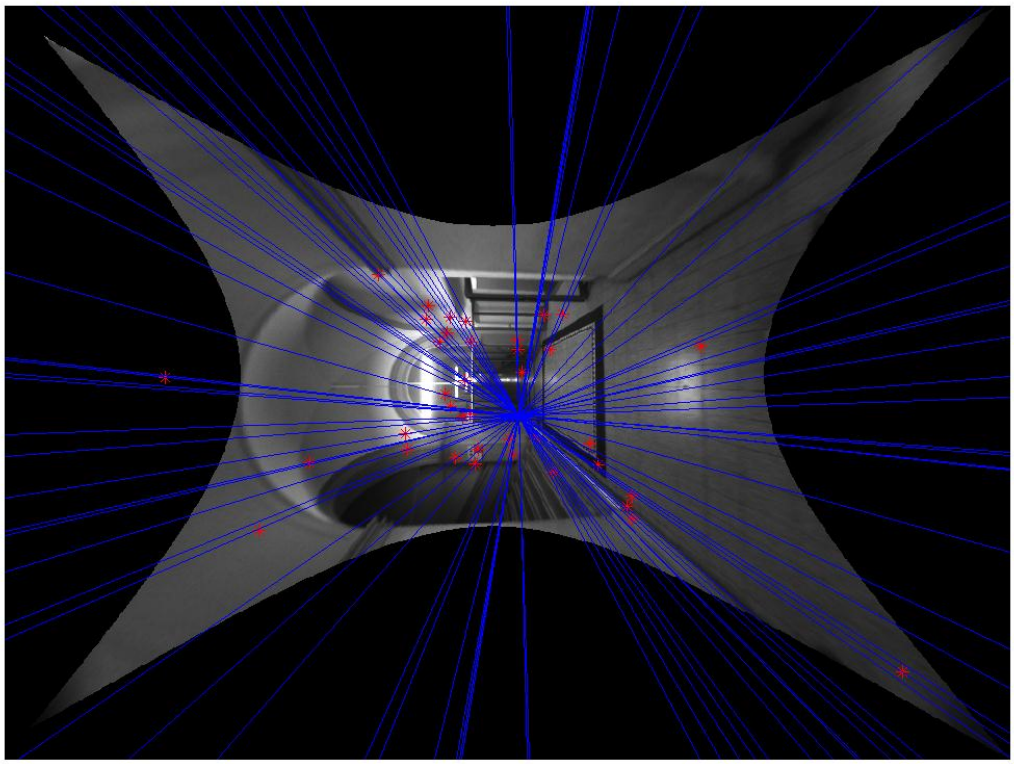
Orthogonal distance is highly sensible to threshold and will affect inlier ratio dramatically.

For the adaptive one, changes are less than normal one, because we are

With changing the threshold from 1 to 12 pixel we see a broad change in inlier_count and iteration_count in the following graphs.

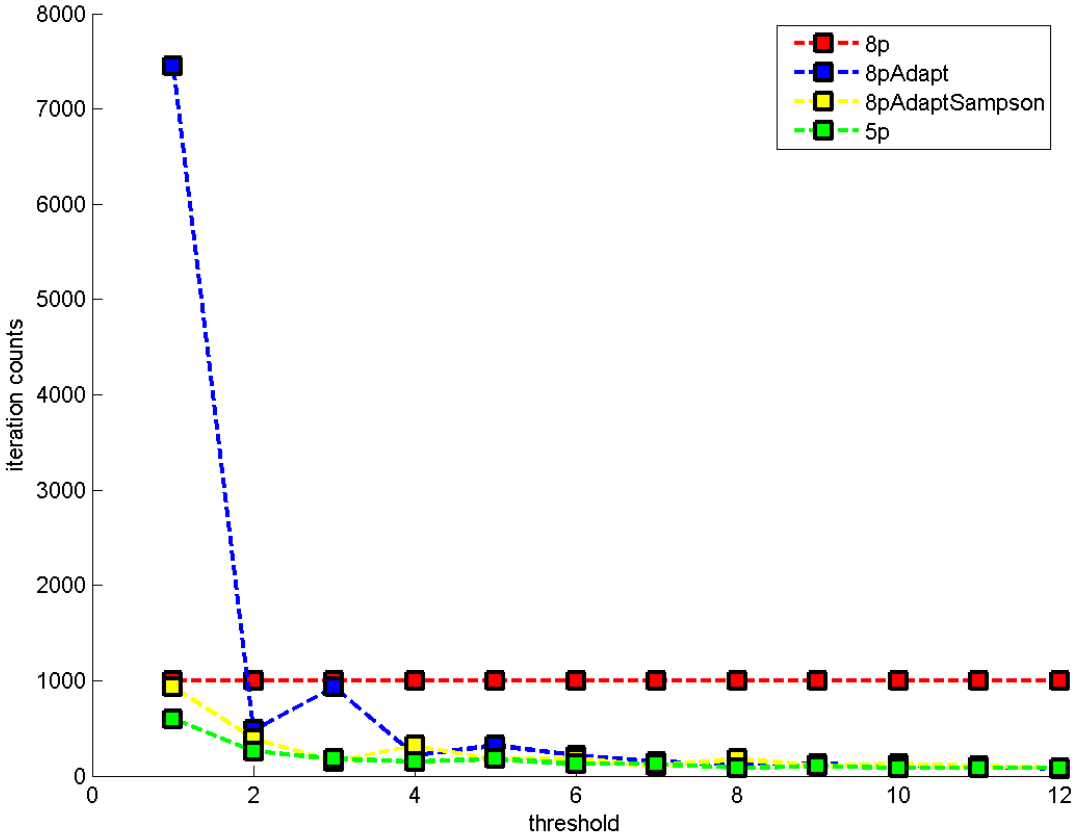


Generally we see that inlier count grows with threshold but of course the accuracy of our matching decreases and the epipolar lines will converge in not very good points. For example, this is the epipolar lines for threshold 10 which converged in a relatively bad point:



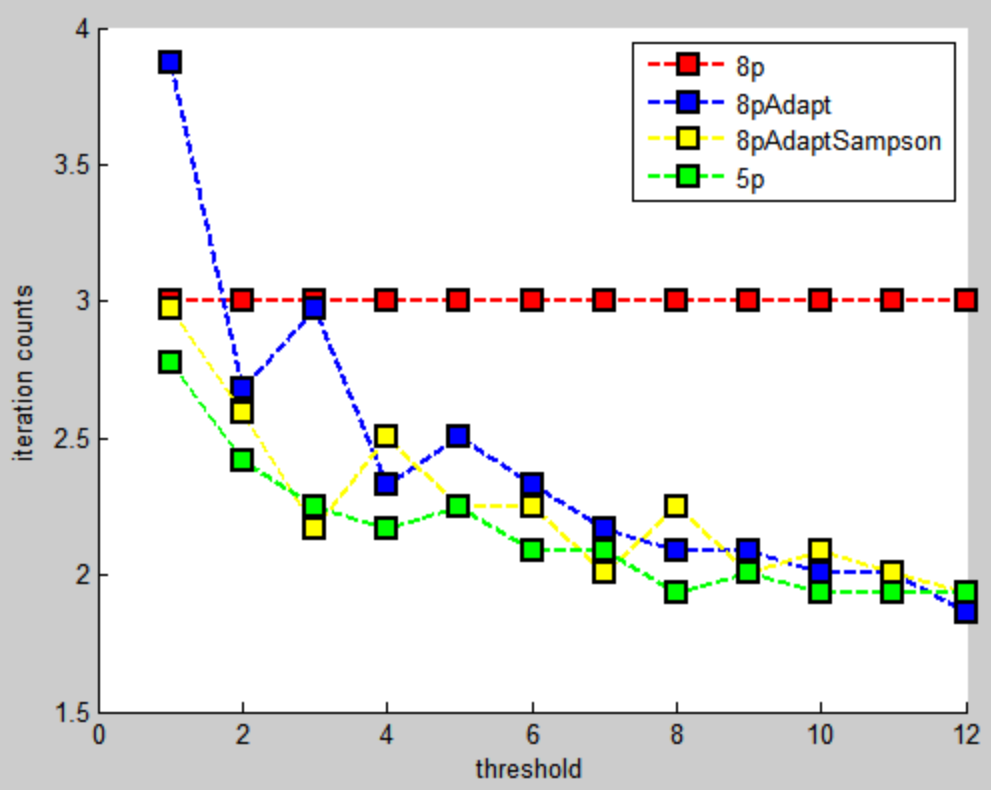
Regarding the inlier count we see that 8point and 8point_adaptive algorithms re highly sensible to threshold and change drastically while 5point algorithm is less variant over threshold changes. I think it is mainly because of the Sampson distance, which is more robust to threshold changes. Also we see that inlier count for 5p algorithm is most of the time superior (especially in low threshold which we desire). So, it seems that solving a 10 degree polynomial worth for low thresholds. The other point is that in small thresholds, the behavior is somehow un-deterministic and highly variant on random data points.

The iteration count graph for all algorithms and thresholds is shown here:



We see that for threshold 1, the 8p_adaptive algorithm hardly struggles to reach the desired accuracy. After that point, generally, iteration count decreases with threshold growth. However, we see that 5p algorithm almost everywhere has the least iteration count which will again prove it worth to solve 10 degree polynomial for that. We also see that Samspn distnce perform overally better thatn perpendicular distance.

These interpretations are more visible when we show the iteration count graph in y-logaritmic sacle.



This is the y-logaritmic scale for iteration_counts.