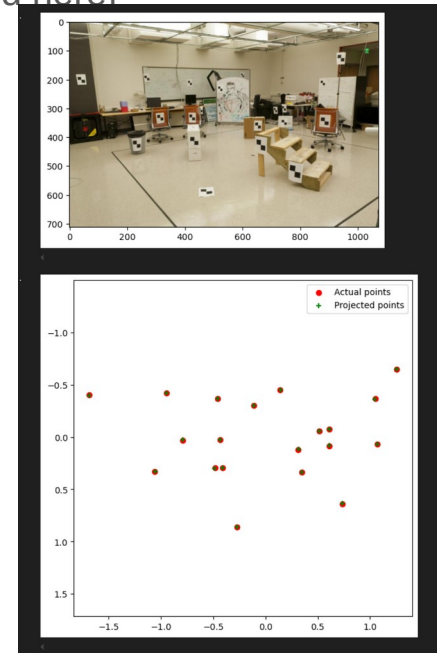
CS 6476 Project 3

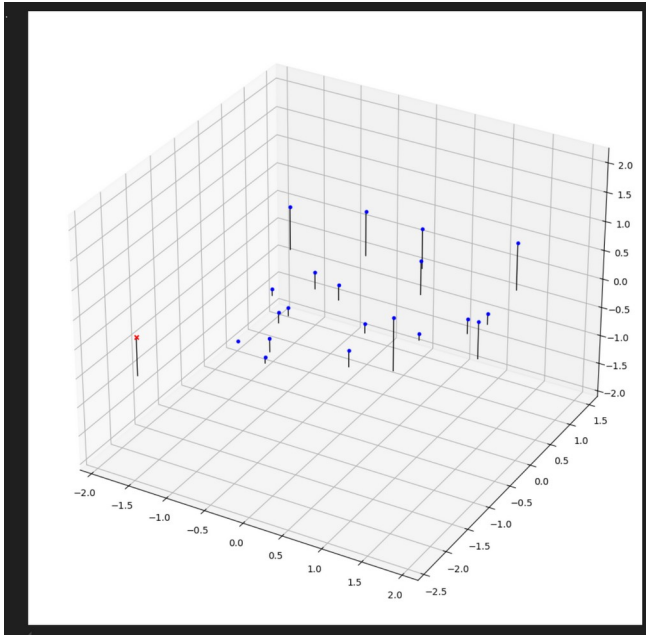
[Ruize Cao]
[rcao73@gatech.edu]
[rcao73]
[904012084]

Part 1: Projection matrix

[insert visualization of projected 3D points and actual 2D points for the CCB image we provided here]

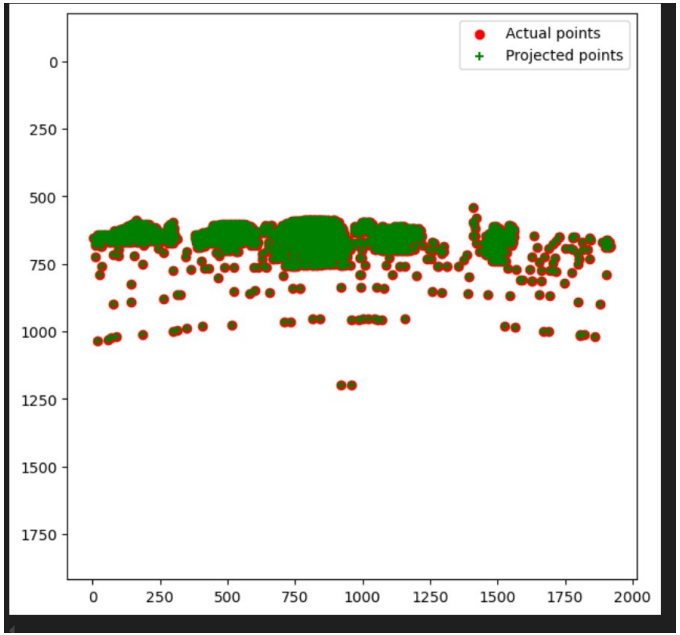


[insert visualization of camera center for the CCB image here]

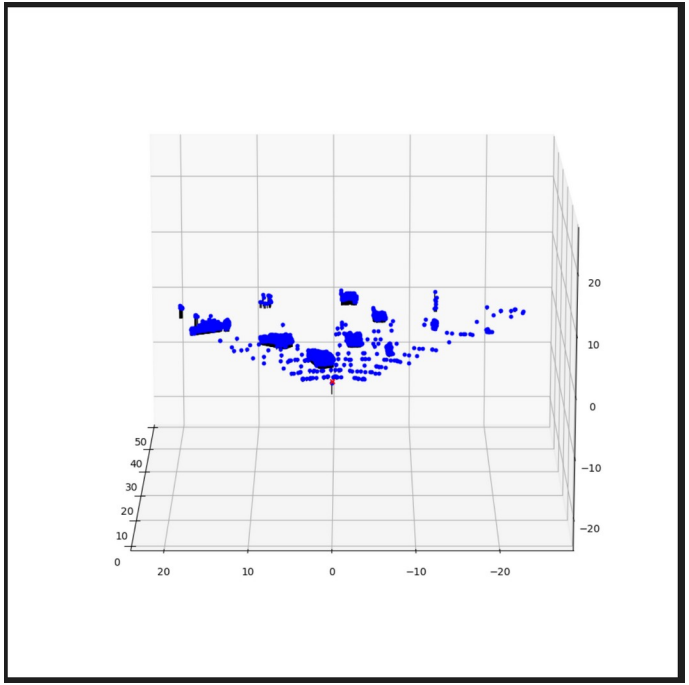


Part 1: Projection matrix

[insert visualization of projected 3D points and actual 2D points for the Argoverse image we provided here]



[insert visualization of camera center for the Argoverse image here]



Part 1: Projection matrix

[What two quantities does the camera matrix relate?]

It relates the 3D points in the real world and the

2D points in the image

[What quantities can the camera matrix be decomposed into?]

M=K[R t], the matrix can be decomposed into
K, R, and t, where K is the intrinsic matrix, R is the
Rotation matrix and t is the translation vector

[List any 3 factors that affect the camera projection matrix.]

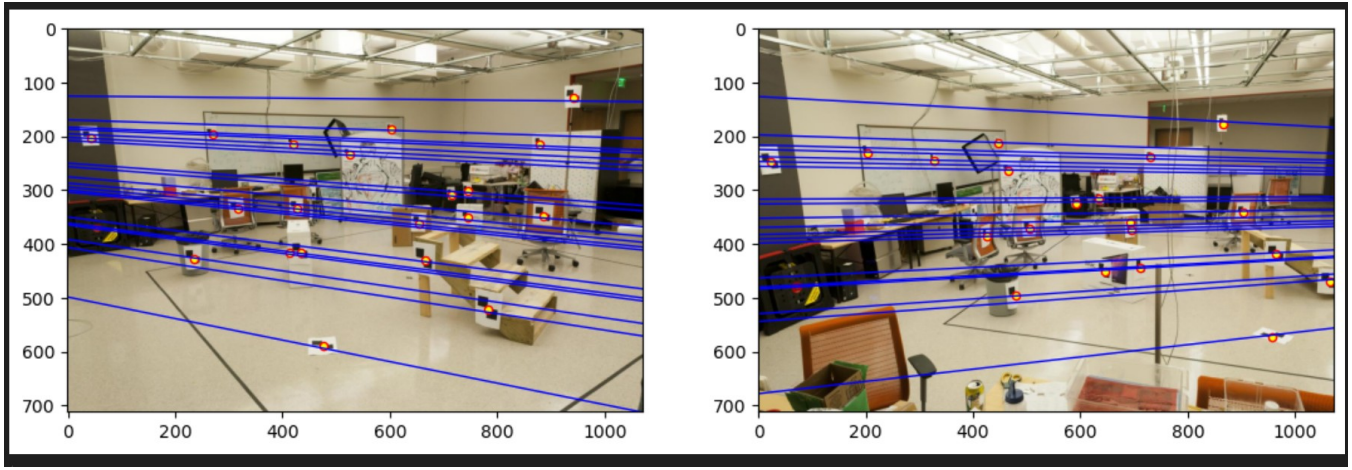
1. As for the intrinsic matrix, it can be affected by the camera parameters, like focal length, camera size, and principle point

2. The extrinsic matrix can be affected by the position and the orientation of the camera

3. Distortion parameters, such as radial and tangential distortion

Part 2: Fundamental matrix

[insert visualization of epipolar lines on the CCB image pair]



Part 2: Fundamental matrix

[Why is it that points in one image are projected by the fundamental matrix onto epipolar lines in the other image?]

Image there is a 3D point that is projected on two different images. When we connect the projected point on the first image and that 3D point, there is a line. Since the camera position can be changed, the corresponding point on the second image can be projected from any points along that line. Therefore, a point in one image is projected into a eipolar line in the another image.

[What happens to the epipoles and epipolar lines when you take two images where the camera centers are within the images? Why?]

The epipolar lines will converge at epipoles. Since the epipolar lines on one image represent the projection lines that connect the 3D points with the projected points, all these projection lines should converge at the center of camera, which correspondingly is the epipole on the other image.

Part 2: Fundamental matrix

[What does it mean when your epipolar lines are all horizontal across the two images?]

The epipoles are at infinite away, therefore, the two image planes are parallel

[Why is the fundamental matrix defined up to a scale?

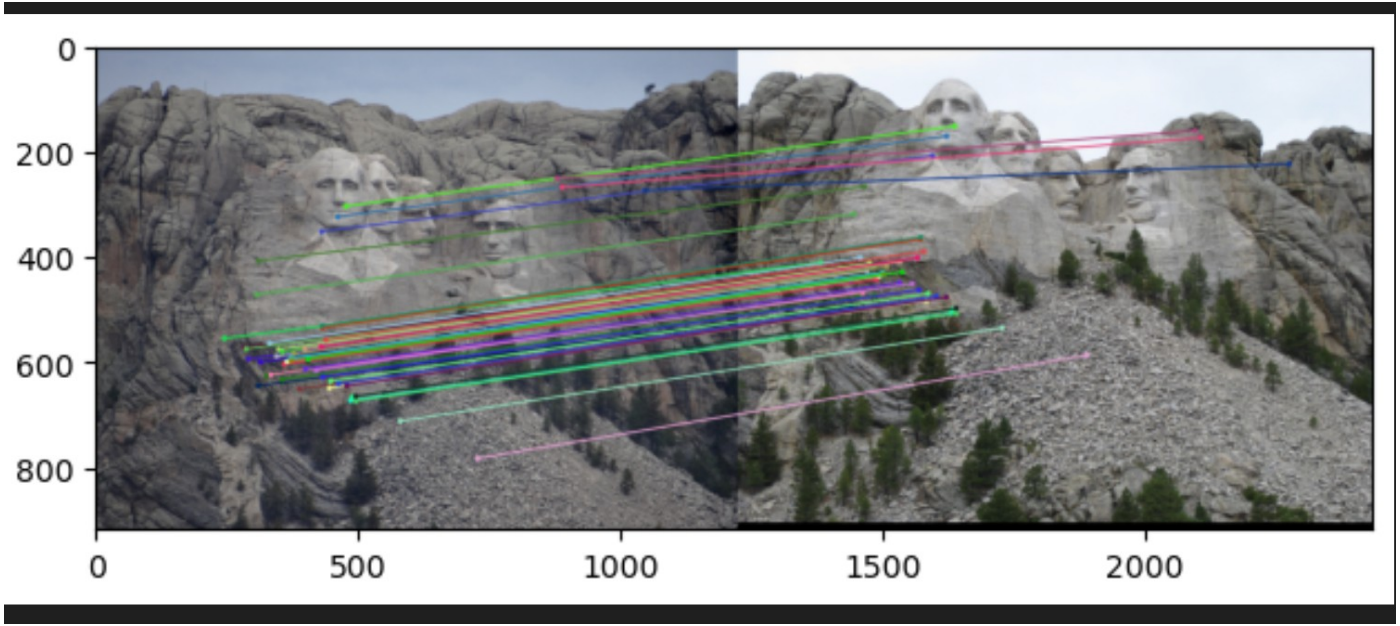
The epipolar constraint equation is homogeneous, meaning that scaling the matrix by any non-zero factor doesn't change the underlying geometric relationship between corresponding points.

[Why is the fundamental matrix rank 2?]

Every point in the image1 must be projected into a epipolar line in the image2, except the center of image1 will be projected as epipole point in image2. In addition, all the epipolar lines in the image2 converge at the epipole, which imposes an additional constrains and deducts one degree of freedom. Meanwhile, the center of camera of image1 actually lies in the null space of fundamental matrix, since its projection is a point instead of a line. Due to the reasons above, the rank of fundamental matrix is 3-1=2

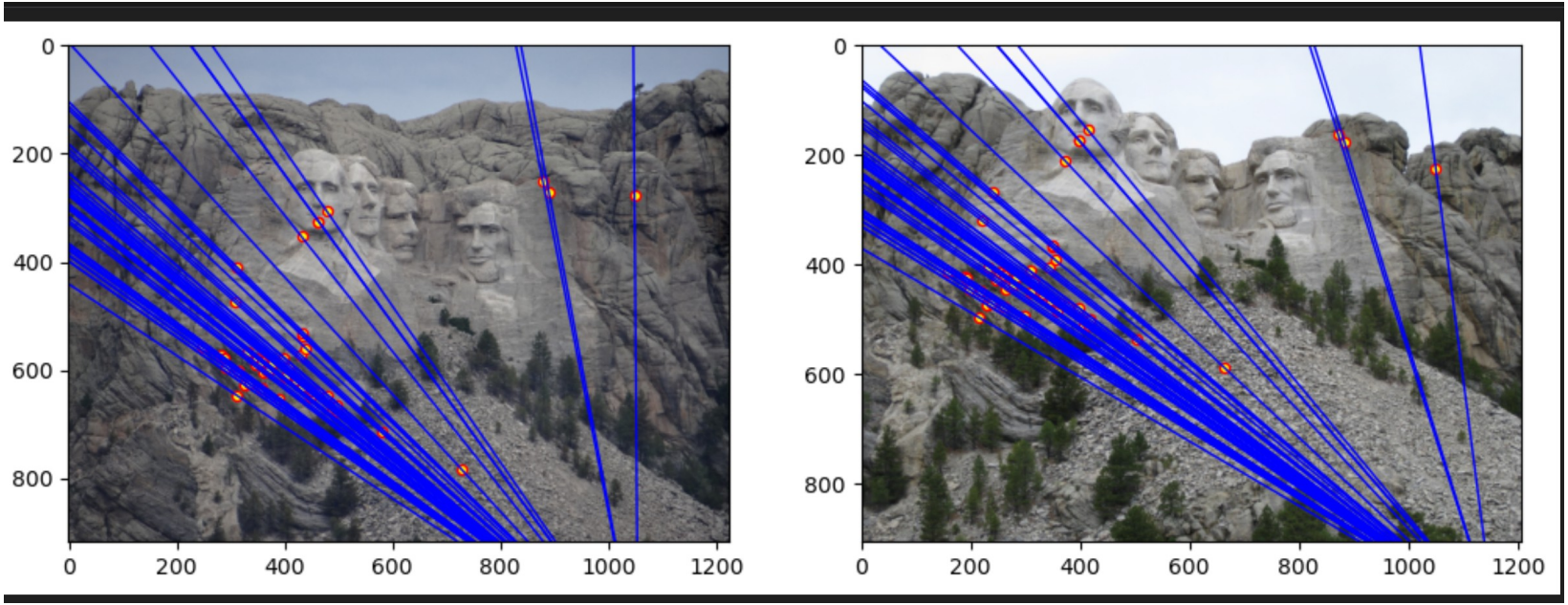
Part 3: RANSAC

[insert visualization of correspondences on Notre Dame after RANSAC]



Part 3: RANSAC

[insert visualization of epipolar lines on the Notre Dame image pair]



Part 3: RANSAC

[How many RANSAC iterations would we need to find the fundamental matrix with 99.9% certainty from your Mt. Rushmore and Notre Dame SIFT results assuming that they had a 90% point correspondence accuracy if there are 9 points?]

From the function compute_ransac_iterations(), i got the number is 10. It is based on the formula which is shown below.

$$N \geq \frac{\log(1 - P_{\text{success}})}{\log(1 - p_{\text{inliers}}^{\text{sample\_count}})}$$

[One might imagine that if we had more than 9 point correspondences, it would be better to use more of them to solve for the fundamental matrix. Investigate this by finding the # of RANSAC iterations you would need to run with 18 points.]

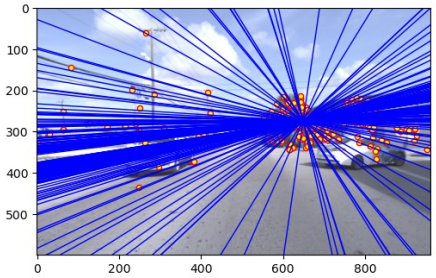
The # of RANSAC iterations is 29. More sample points do attribute to more accurate foundamental matrix, but it increases the computational load, since the # of RANSAC iterations increases almost three times.

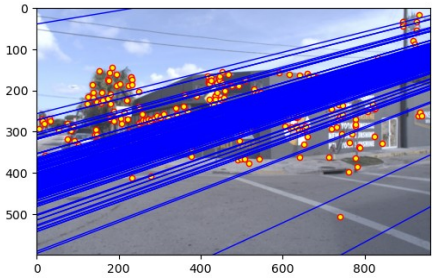
[If our dataset had a lower point correspondence accuracy, say 70%, what is the minimum # of iterations needed to find the fundamental matrix with 99.9% certainty?]

The # of RANSAC iterations is 112, since the correspondence accuracy decrease, it will significantly makes it harder to have a sample that all points are inliers. Therefore, it will significantly increase the # of RANSAC iterations.

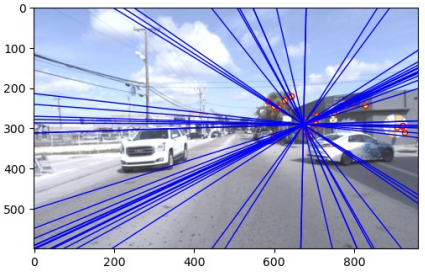
Part 4: Performance comparison

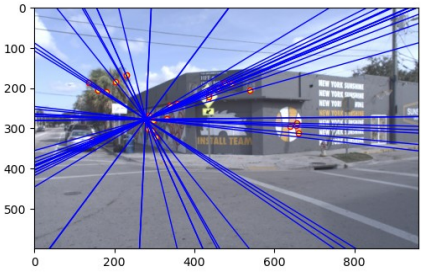
[insert visualization of epipolar lines on the Argoverse image pair using the linear method]





[insert visualization of epipolar lines on the Argoverse image pair using RANSAC]





Part 4: Performance comparison

[Describe the different performance of the two methods.]

For the linear method, the performance on the left image is acceptable, but for the right image, the performance is quite bad, even the epipolar lines do not converge.

For the RANSAC method, the performances on both image are good, clear epipolar lines and all lines converge at the epipoles.. Correct and valuable corresponding pairs are selected for calulate the foundmaental matrix, epipolar lines and epipoles.

[Why do these differences appear?]

For the linear method, the output is quite sensitive to noises and outliers, since all pairs are involved for calculation.

For the RANSAC, it will continuously sample a number of corresponding pairs for estimated iteration number, which is calculated based on desired successful probability, sample number, and the inliers probability. Therefore, it can filter unexpected corresponding pairs, and find the best sampled dataset for calculating the foundamental matrix.

[Which one should be more robust in real applications? Why?]

The RANSAC one is more robust. Since it is less sentive to noises by continuously sampled the corresponding pairs to find best sampled dataset for calculation.

Part 5: Visual odometry

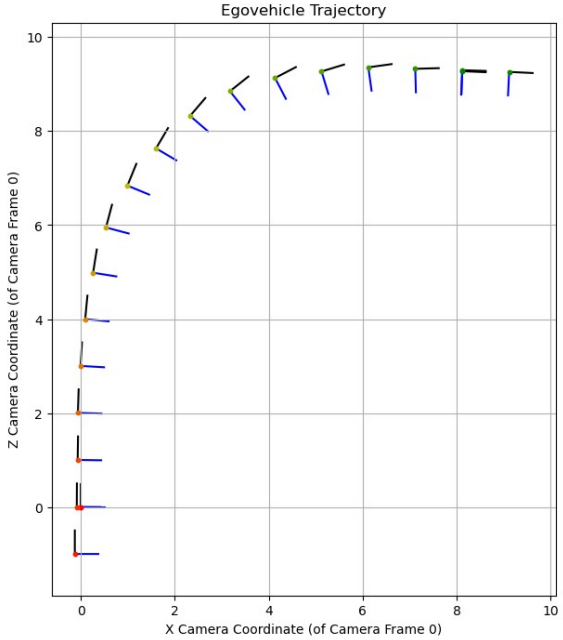
[How can we use our code from part 2 and part 3 to determine the “ego-motion” of a camera attached to a robot (i.e., motion of the robot)?]
We can use the ransac_foundamental( ) function in part3, which uses compute_ransac_iterations( ) in part3 to compute iteration numbers, and uses compute_fundamental_matrix( ) to repeatly compute the best foundamental matrix based on sampled feature points from ransac algorithm. If we get the foundamental matrix and the intrinsic matrix, we can gain the essential matrix E, which can be decomposed it into a translation matrix and a rotation matrix. Hence, we gain the translation vector and rotation matrix between the center of two images. Based on that we can calculate the motion of the camera

[In addition to the fundamental matrix, what additional camera information is required to recover the ego-motion?]

We also need intrinsic matrix K, sice E=K.T @ F @ K, which F is foundamental matrix, and K is intrinsic matrix

Part 5: Visual odometry

[Attach a plot of the camera’s trajectory through time]



Part 6: Panorama Stitching

[Please add a README style documentation here for your implementation of panorama stitching with: description of what you implemented, instructions on how to replicate the results in clear steps that can be followed by course staff. Failure to replicate results by following this documentation will result in point penalties on this question of the assignment.]

code logic:

The program firstly loads the images pair from the folder “additional_data\”. Then it converts the colored images into gray images. Then the programs can use either SIFT or ORB algorithm to gain feature points from two images. You can decide which algorithm to use by changing the value of “choice” variable in the codes. After that, distance ratio matches are deployed to gain matched feature points.

Based on that, the homography matrix H is computed via the function cv.findHomography(src_pts, dis_pts, cv.RANSAC). Then the points in image B is mapped in the coordinates of image A by H @ pts_B.

When the mapped points of pts_B fall beyond the region of image A, the value of pts_B will be directly utilized in panorama image. On the contrary, when mapped points fall inside the region of image A, a blending algorithm will determine the value of this point in panorama.

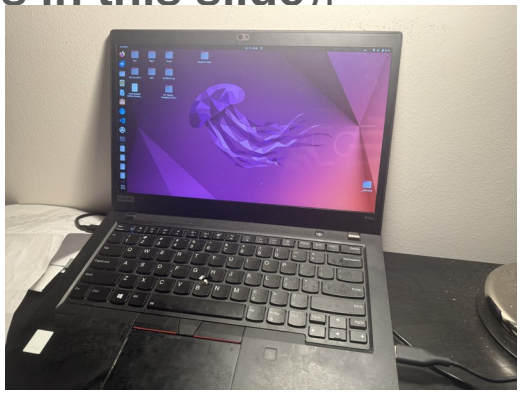
results replicate:

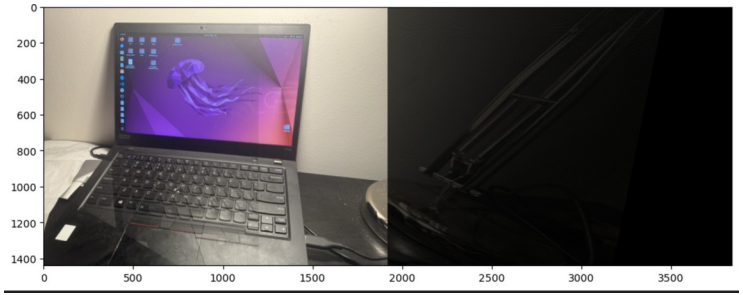
```
from vision.part6_panorama_stitching import panorama_stitch
imageA = "additional_data/right.jpg"
imageB = "additional_data/left.jpg"
panorama= panorama_stitch(imageA, imageB)
plt.figure(figsize=(12,6))
plt.imshow(panorama)
```

Part 6: Panorama Stitching

[Insert visualizations of your stitched panorama here along with the 2 images you used to stitch this panorama (**there should be 3 images in this slide**)]

.





Part 6: Panorama Stitching

[Discuss the results of your panorama stitching. What feature detectors and matchers did you use? How did they influence the quality of the final panorama?].

The result of my panorama stitching is not ideal, since the right part of the panoma is dim meanwhile the features inside are stretched. It seems that the stretched geometry and changed intensity of the transformed image are big problems. I used the SIFT feature detectors and ratio distances matchers to filter matches. Both of them play a role to provide a realiable points matches, which will be based for calculating the homography matrix between two different images. Therefore, the quality of provided points matches directly determines the quality of final panorama.