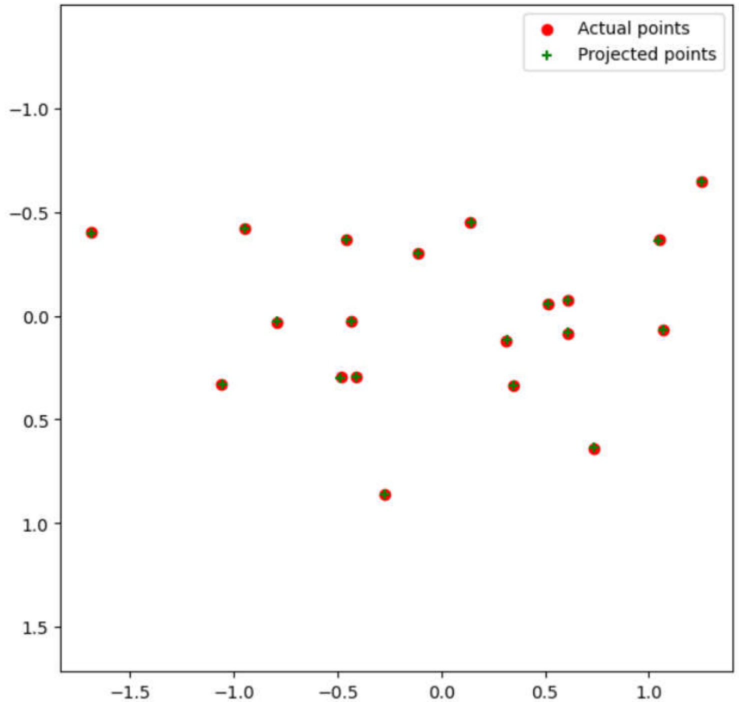
CS 4476/6476 Project 3

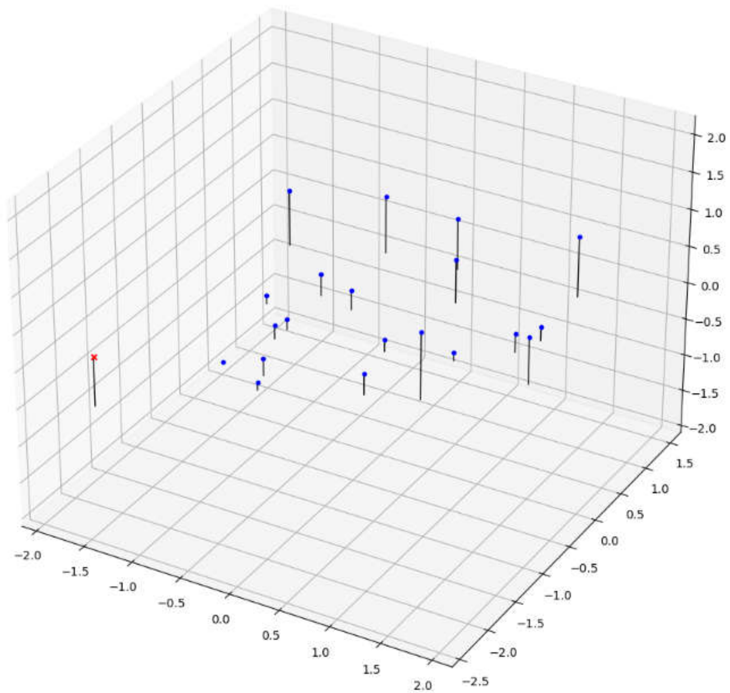
[Jiarui Xu]
[jxu605@gatech.edu]
[jxu605]
[903617486]

Part 1: Projection matrix

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

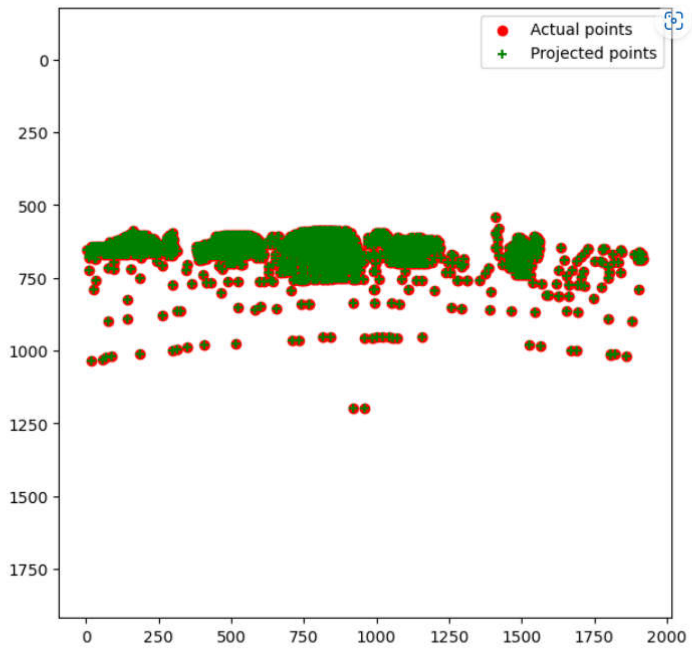


[insert visualization of camera center for the CCB image here]

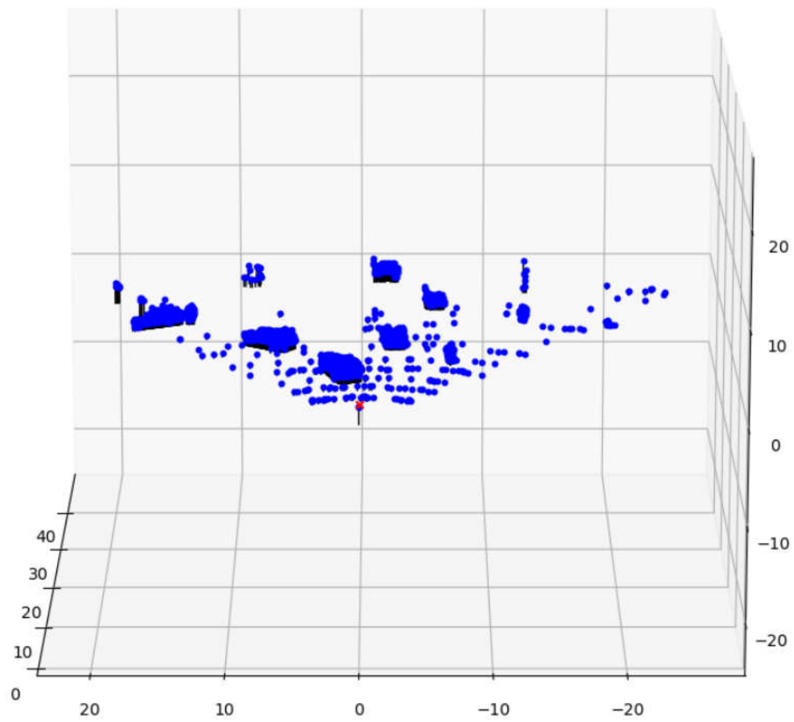


Part 1: Projection matrix

[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?]

2D image points and 3D points

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

intrinsic parameters

extrinsic parameters

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

camera translation

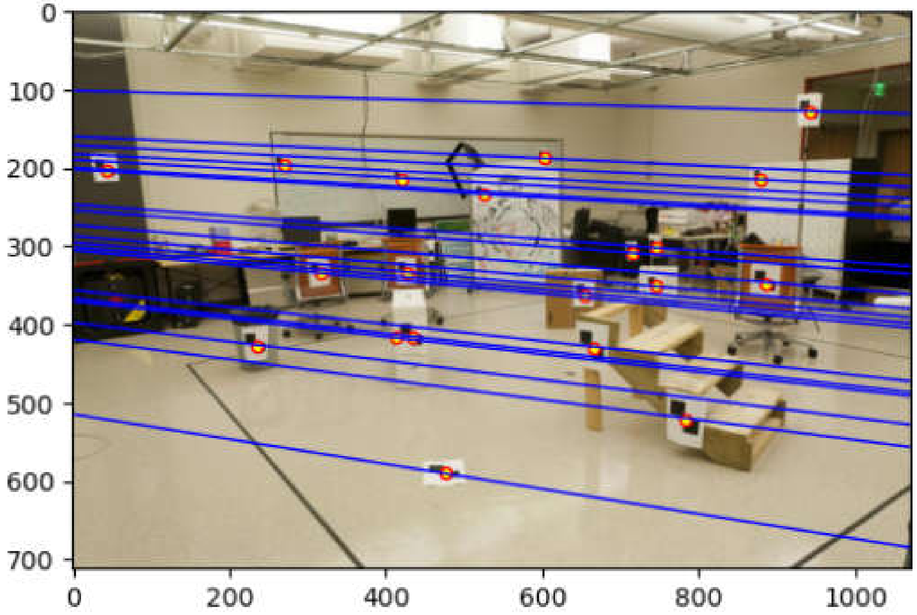
camera rotation

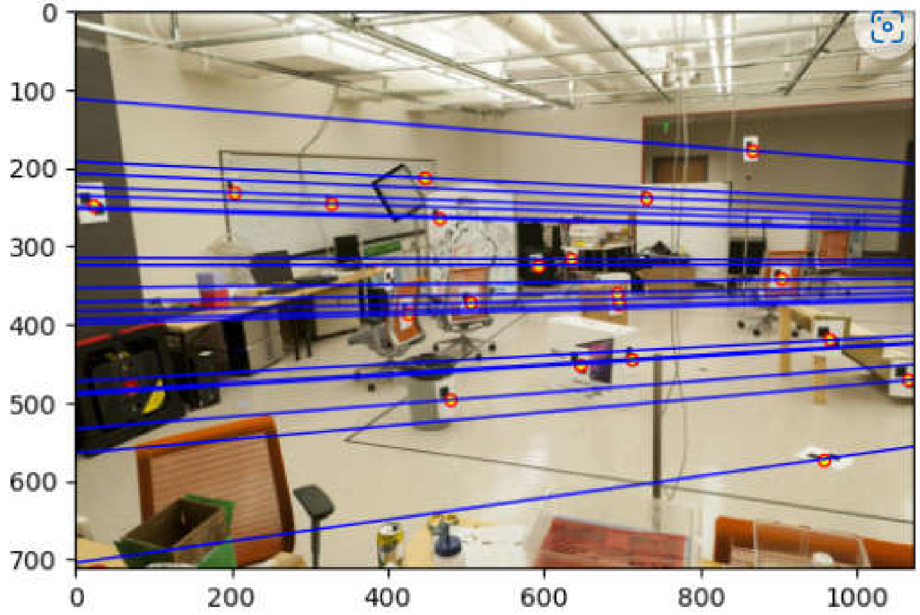
optical center

The focal length of the camera

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?]

Since we don't know the actual depth of the points, a point in the image represents a line in 3D space. In the other picture, this point is projected onto the epipolar line.

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

I think when camera centers are within the images, all the epipolar lines will converge on the epipole, and the epipole in the image is the optical center of the other camera.

The reason for that is: all the epipolar lines goes through the object and the projection of the epipole.

Part 2: Fundamental matrix

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

The two cameras' image planes are parallel.

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

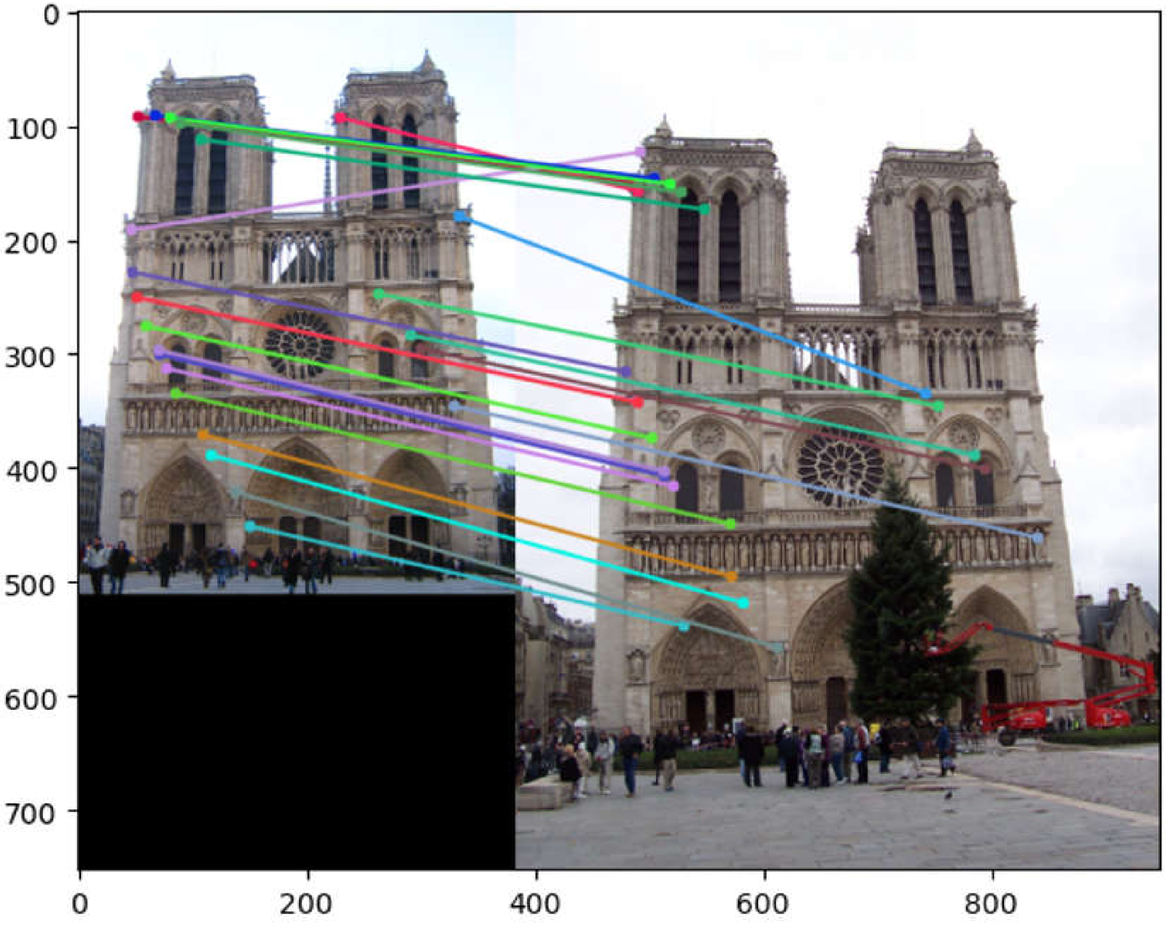
Because when a camera translates to another camera, the scale is only up to one scale.

[Why is the fundamental matrix rank 2?]

The reason for that is: the fundamental matrix is derived from the essential matrix, and the essential matrix is equal to =[T×]R, [T×] has rank 2.

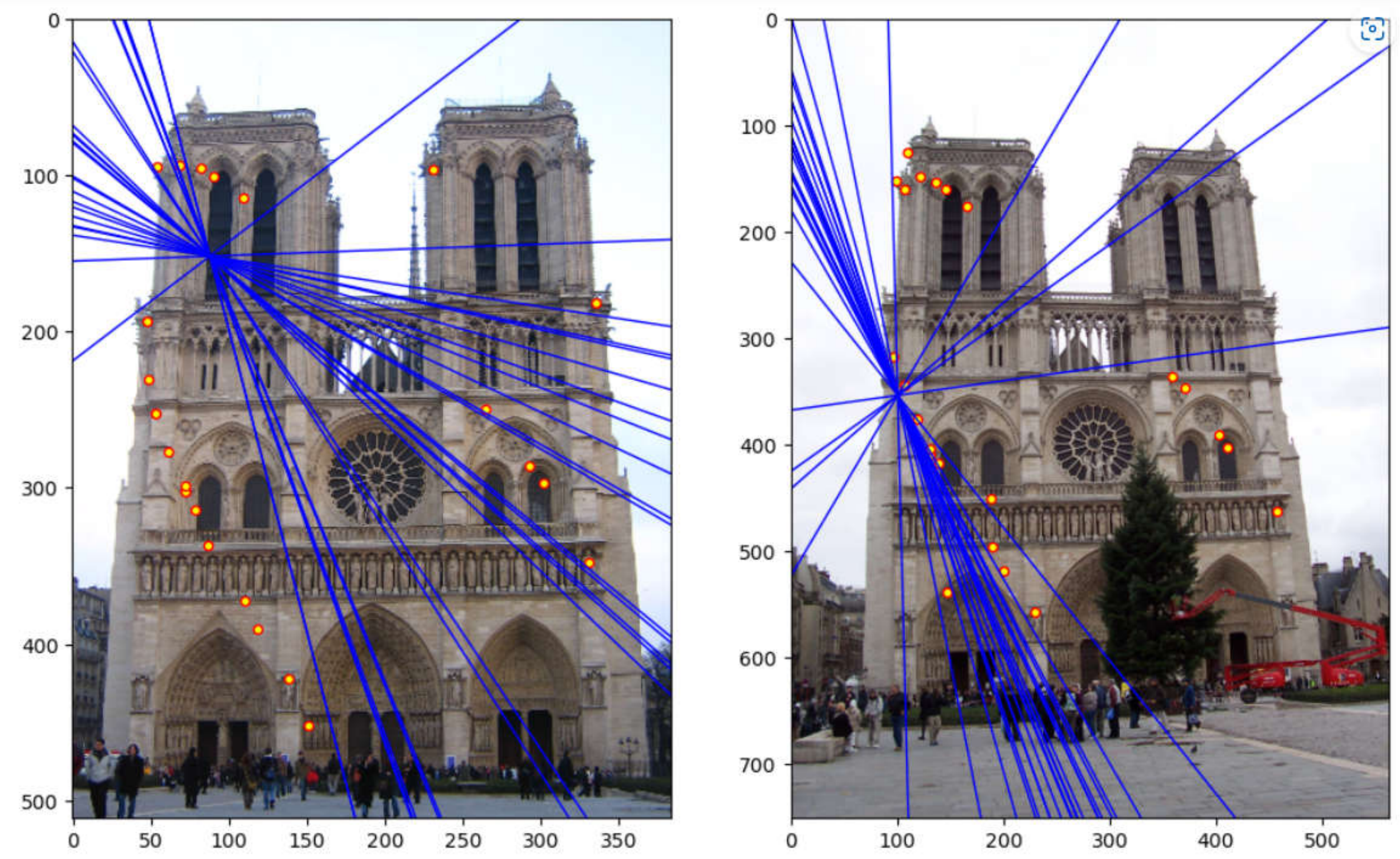
Part 3: RANSAC

[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?]

13 iterations

[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.]

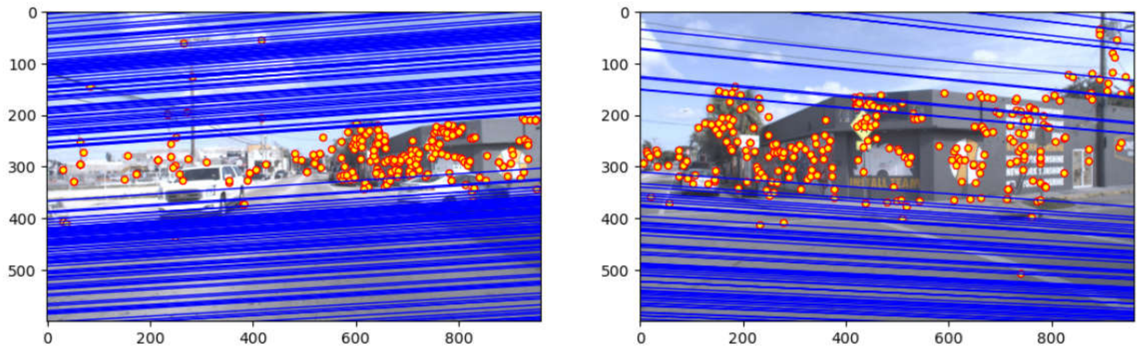
43 iterations

[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?]

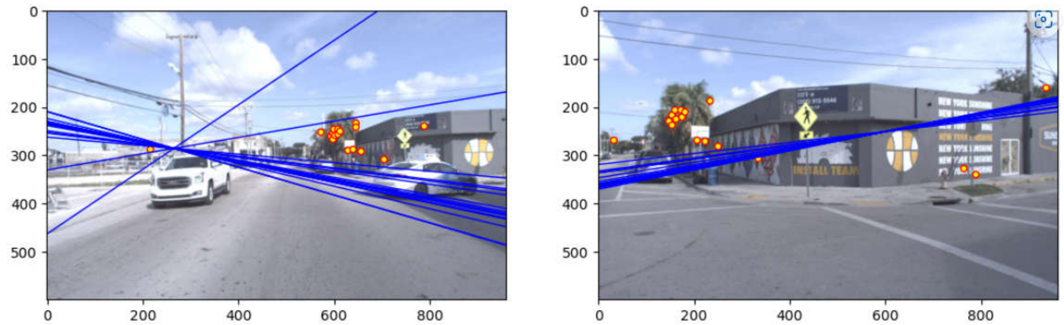
117 iterations

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.]

The positions of the epipole of the two methods are different. We can see clearly that the epipole is out of the image in linear method, but for RANSAC it is not.

Also RANSAC has fewer and more precise matching points.

[Why do these differences appear?]

RANSAC can capture precise matching points during iterations. But linear method cannot.

So linear method’s epipole will be influenced more by mismatched points than RANSAC.

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

RANSAC.

Because in real applications, mismatched points are every where. If they have great influence on epipole, the result would be meaningless.

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)?]

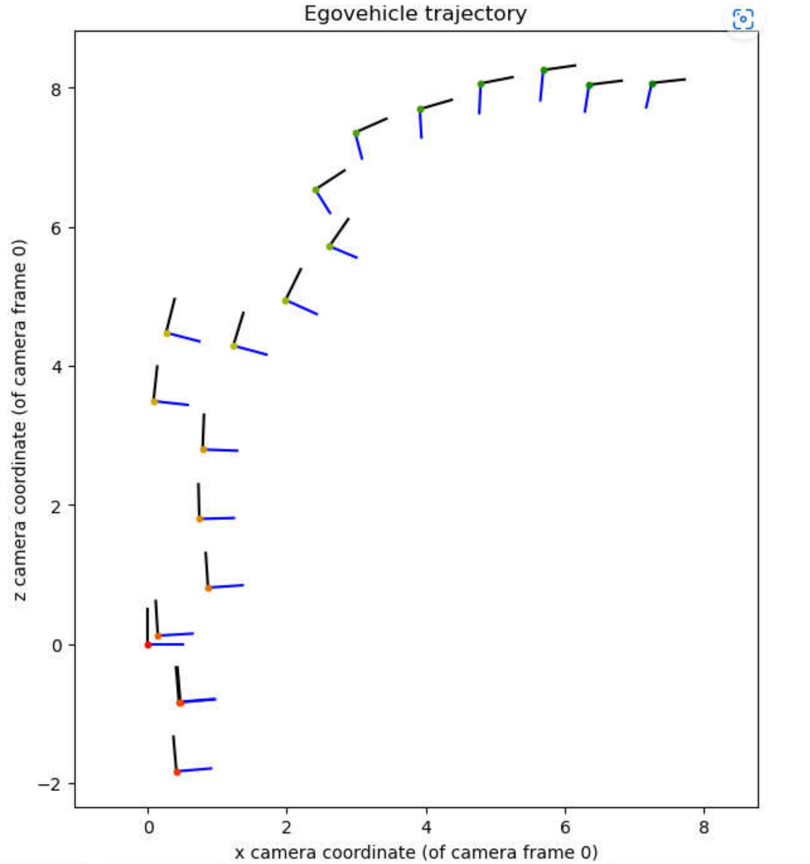
We can calculate the position of the center of the camera using part2 and part3. When time goes by, the position the center of the camera will change based on the “ego-motion”, so by comparing the changing of the position, we can determine that.

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

intrinsic parameters

Part 5: Visual odometry

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



Part 6: Panorama Stitching

Implement
1.I loaded the images and converted the images to grayscale.
2.Then I computed the SIFT keypoints and descriptors
A Scale Invariant Feature Transform (SIFT) feature or keypoint, is a selected circular region of an image with an orientation.
3.Find Top M matches of descriptors of 2 images
Now that I have SIFT keypoints and descriptors, I need to find the distance of each descriptor of image 1 to each descriptor of image 2.
bf = cv.BFMatcher()
4.I select the top M matches of the descriptors. Here, I took the value of M to be 2.
matches = bf.knnMatch(des1,des2, k=2)
5.Choose/get interest points for the pair of images.
I aligned the 2 images using homography transformation
6.Find candidate matches among the interest points. Also use RANSAC.
I calculated the homography matrix of the 2 images, which require atleast 4 matches, to align the images. It uses Random Sample Consesus (RANSAC), which produces right results even in presence of bad matches.
7.function to_tx() and to_img(mtr) are for changing the representation in order to make my implementation more simple
8.Implementing warpPerspective. As for affine transform, some pixels from original image might be mapped outside or into subwindow of the original image.
9.Project each image onto the same surface and stitch (warp operation)
Once I have the homography for transformation, I can warp the image and stitch the two images.
Replicate:
Just put two images named 1.jpg and 2.jpg in file additional_data and run the following code:
The result image will be printed in file additional_data

```
from vision.part6_panorama_stitching import panorama_stitch
imageA = "additional_data/2.jpg"
imageB = "additional_data/1.jpg"
panorama = panorama_stitch(imageA, imageB)
```

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**)].