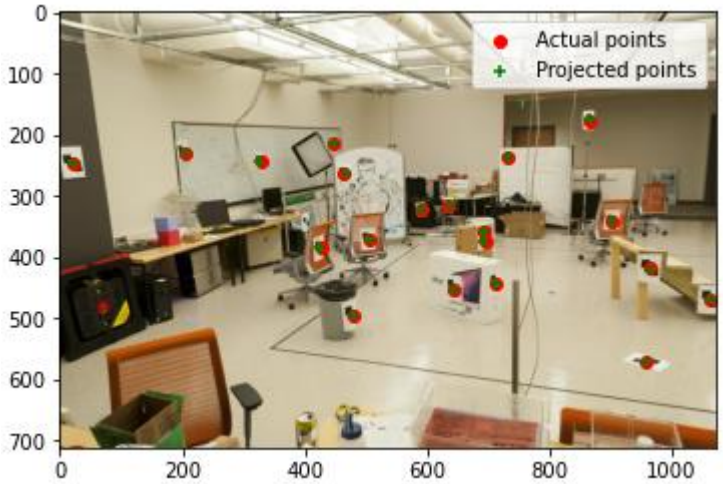
CS 6476 Project 4

Karan Sarkar
ksarkar9@gatech.edu
903569592

Part 1: Projection Matrix

<insert visualization of projected 3D points and actual 2D points for image provided by us and the total residual here>

Residual = 14.711445



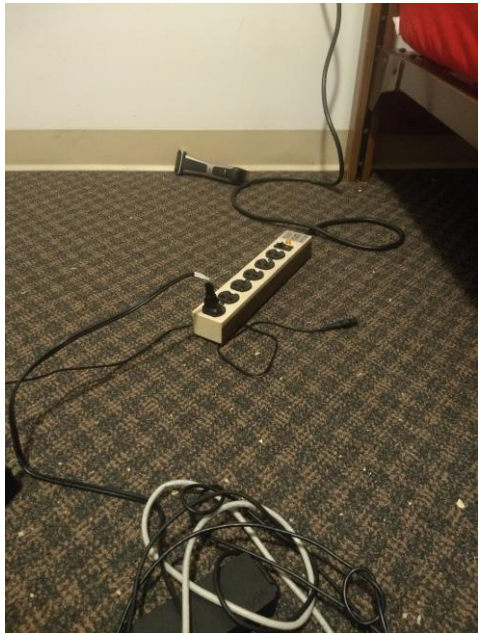
Part 1: Projection Matrix

At least how many 3D-2D point correspondences do you need to estimate the projection matrix? Why?

The projection matrix has 11 degrees of freedom. It is a 3 by 4 matrix so that gives us 12 degrees of freedom. We subtract one to account for a free scaling factor. Each correspondence gives us 2 degrees of freedom one for the x-coordinate and y-coordinate. Thus, we must have 6 point correspondences.

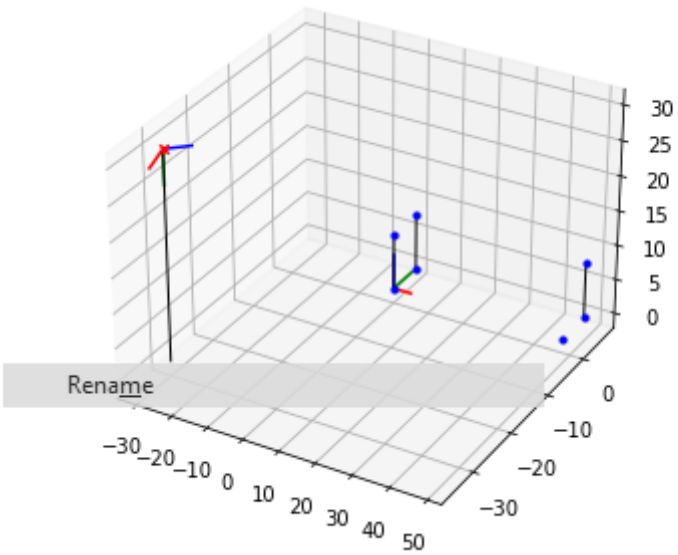
Part 2: Estimation with Your Own Images

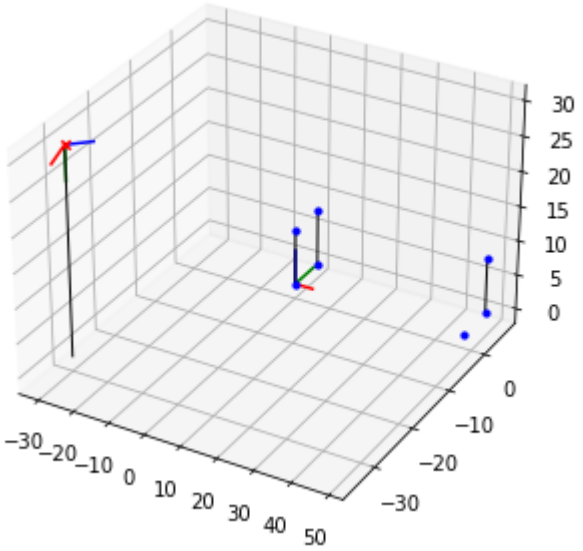
< insert the two images of your fiducial object here>



Part 2: Estimation with Your Own Images

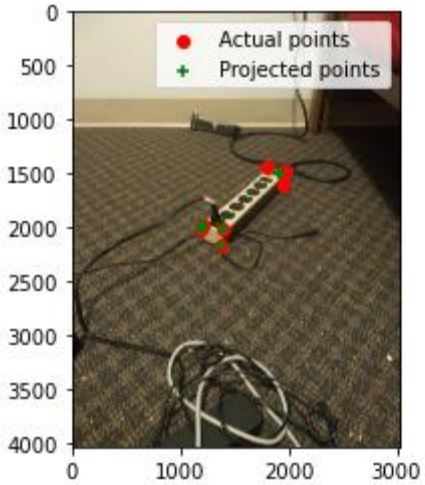
<insert visualization the initial guesses for rotation matrix and camera center for the two images you took here>

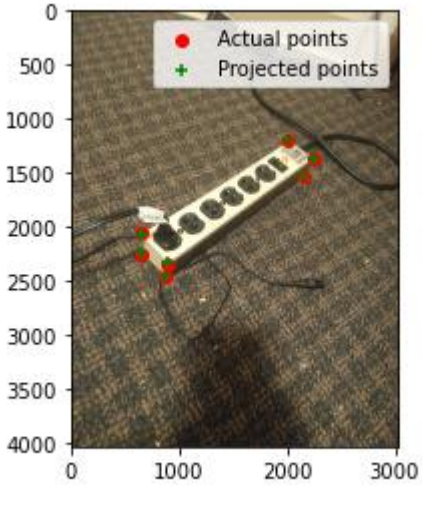




Part 2: Estimation with Your Own Images

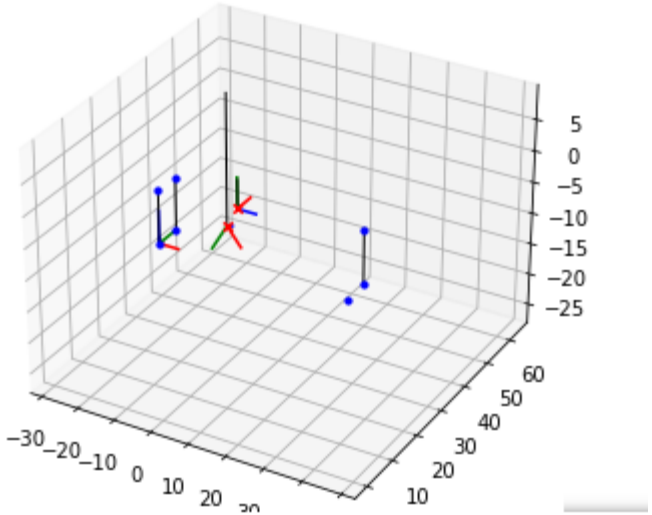
<insert visualization of projected 3D points and actual 2D points for both the images you took>





Part 2: Estimation with Your Own Images

<insert visualization of both camera poses of images you took here>



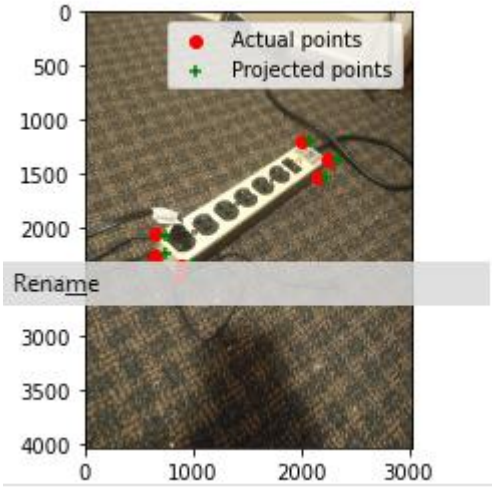
Part 2: Estimation with Your Own Images

- What would happen to the projected points if you increased/decreased the x coordinate, or the other coordinates of the camera center t? Write down a description of your expectations in the appropriate part of your writeup submission.

- If you adjust the camera x units in a certain direction. I believe the projected points will move x units in the exactly same direction. For example, if you move 5 the camera units left, the projected points will move 5 units left.

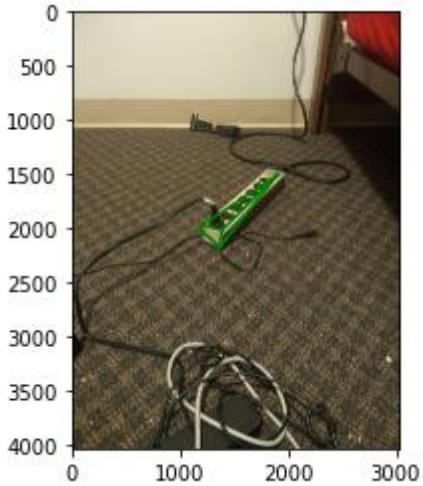
- Perform this shift for each of the camera coordinates and then recompose the projection matrix and visualize the result in the Jupyter notebook. Was the visualized result what you expected?

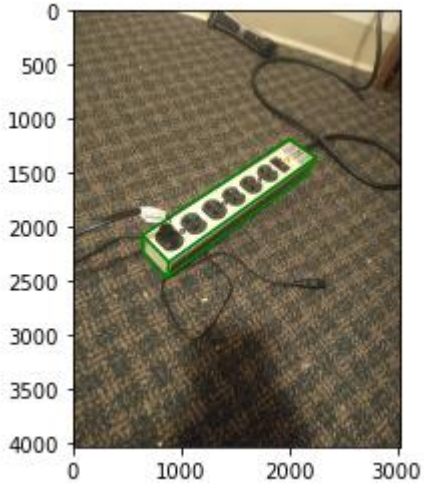
- The visualized result was exactly what I expected. I shifted the camera to the right by 100 units.



Part 2: Estimation with Your Own Images

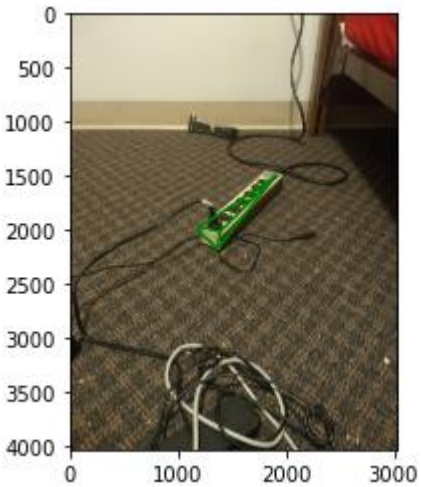
<insert both images you take with bounding boxes around your fiducial object>

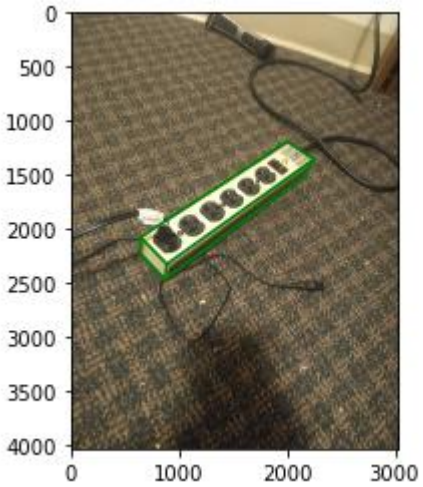




Part 3: Camera Coordinates

<insert both images you take with bounding boxes around your fiducial object>





Part 3: Camera Coordinates

The first three rows of the transformation matrix are just the extrinsic parameters of our camera, why is that?

First note that the transformation matrix can be thought of as the product of the inverse calibration matrix and the projection matrix where the calibration matrix is in the camera reference frame..

$$[\,{}^{w}\mathbf{R}_{c}^{T} \mid -\,{}^{w}\mathbf{R}_{c}^{T}\,{}^{w}\mathbf{t}_{c}] = \mathbf{K}^{-1}\mathbf{K}[\,{}^{c}\mathbf{R}_{w} \mid {}^{c}\mathbf{t}_{w}] = \mathbf{K}^{-1}\mathbf{K}[\,{}^{w}\mathbf{R}_{c}^{T} \mid -\,{}^{w}\mathbf{R}_{c}^{T}\,{}^{w}\mathbf{t}_{c}] = \mathbf{K}^{-1}\mathbf{P}$$

Now note that we can express the projection matrix as the calibration matrix times the matrix of extrinsic parameters.

$$\mathrm{P} = \mathbf{K}\,{}^{c}\mathbf{R}_{w}[\mathbf{I}| - {}^{w}\mathbf{t}_{c}]$$

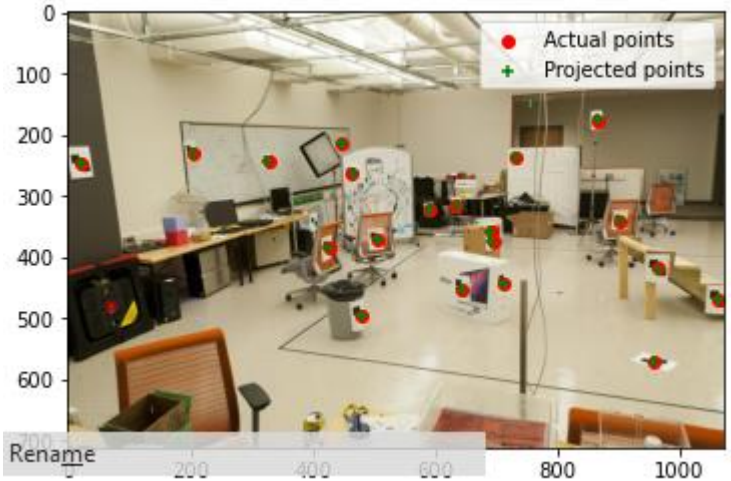
Therefore, transformation matrix cancels out the calibration matrix and we are left with the extrinsic parameters.

Say if we know the coordinates of 3D points in camera 1 coordinate system, and we want to transform them into camera 2 coordinate system, what should we do? You may assume that the poses of both cameras are known.

We multiply by the inverse of the transformation matrix of camera 1, to transform the coordinates into the world reference frame. Next, we multiply by the transformation matrix of camera 2, to transform the coordinates into camera 2’s coordinate system.

Part 4: DLT

<insert visualization of projected 3D points and actual 2D points for image provided by us and the total residual here>
Residual = `15.544953`



Part 4: DLT

Why do we need the cross product trick?

The cross product trick expresses the cross product as a matrix multiplication. This allows us to write the expression as a matrix times a vectorized version of the Projection matrix. Having the problem as minimizing a matrix vector product allows the problem to be solved in a closed form.

We pick up the eigenvector with the smallest eigenvalue. Will this eigenvalue be exactly zero? Why/why not?

It will not be zero because of measurement and precision error.

EC: RANSAC Iterations Questions

How many RANSAC iterations would we need to find the fundamental matrix with 99.9% certainty from your Mount Rushmore and Notre Dame SIFTNet results assuming that they had a 90% point correspondence accuracy?

You would need 9 iterations.

One might imagine that if we had more than 6 point correspondences, it would be better to use more of them to solve for the fundamental matrix. Investigate this by finding the number of RANSAC iterations you would need to run with 12 points.

It would take 20 iterations.

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

It would take 55 iterations if sampling 6 points and 495 iterations if sampling 12 points.

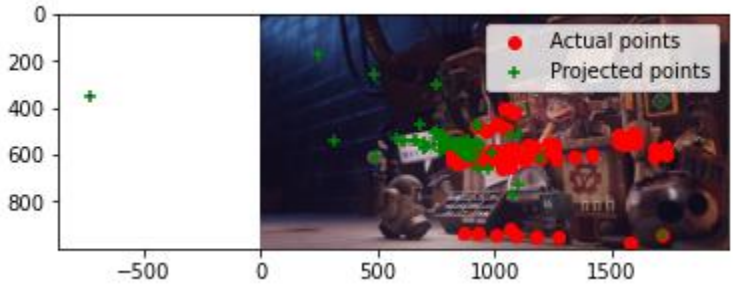
EC: RANSAC

<insert the number of inliers from your best model here>

6 inliers.

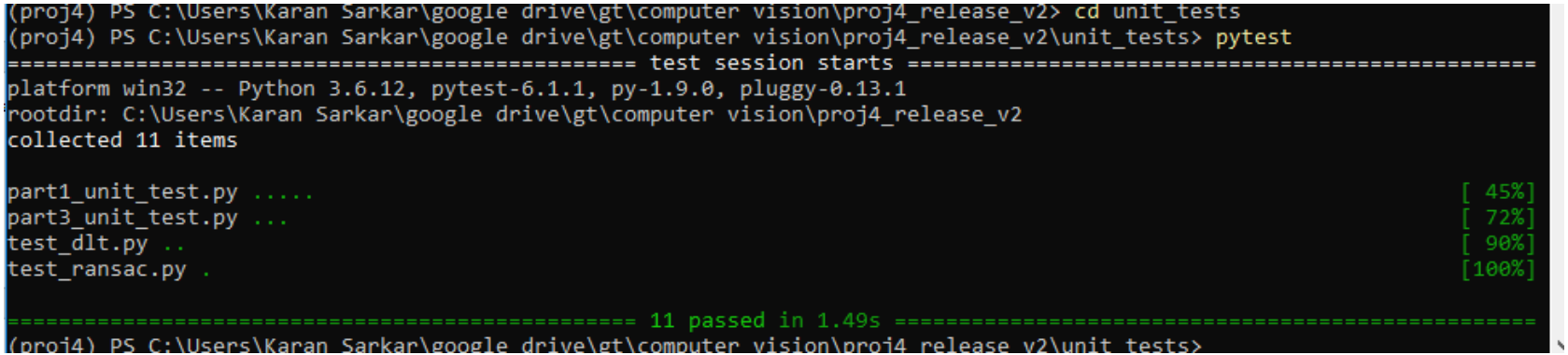
<insert the image and total residual here>

Residual= 30935.237466



Tests

<Provide a screenshot of the results when you run `pytest` from the unit tests directory with your final code implementation (note: we will re-run these tests).>



Conclusions

<Describe what you have learned in this project. Feel free to include any challenges you ran into.>

I did not really have any problems doing it. I found it very interesting how we were able to get different accuracy and residual values by using Levenberg Marquardt, Direct Linear Transform and Ransac.