Daniel Perkins

Dr. Lee

ECEN 631

23 March 2025

Task 1

In this task, I used goodFeaturesToTrack() to detect good features in each image. Then, I used calcOpticalFlowPyrLK() to find points in the next frame and determine the flow between the corresponding features. Furthermore, whenever the number of points was less than 300, I called the goodFeaturesToTrack() function again to get 400 features.

After some fine tuning, I got desirable results. The video can be found at the link: https://youtu.be/fiKHSfZHYAA. I found that the most optimal number of frames to skip was 5. This was small enough for the features to be close to each other, but far enough to allow us to visualize the movement. I also used a pyramid level of 5 and window size of (21 x 21). Both parameters were chosen after testing various values. To my surprise, they did not seem to have that much of an effect on the results. This is likely because the movement in the video was slow enough and the features were dense enough.

Overall, I learned that it is relatively straightforward to determine the motion of features in a video. One has to do a lot of fine tuning to find an optimal window size and pyramid level, based on the speed of the movement in the video. But, once those parameters are determined, you can get satisfactory results (with some noise).

Task 2

This time, with the same goal in mind, I used a template matching technique. To calculate the motion, I skipped 5 frames (as it worked in the first task). I tried two different things:

1) I used the ideas from the previous homework assignment. I set up a SIFT detector and used BFMathcer() to match the points from frame to frame. Of the matches, I took the best ones and used those to detect the movement of each point. This worked well. But, it was a little slow. It is equivalent to using the matchTemplate() function where the search window and template are the entire

image (and an optimized algorithm is used to remove features that are matched too far away).

2) I tried using the goodFeaturesToTrack() function to find features. Then, I took a window (of size 20 on each side) of the image around each feature for both the new image and the old (template). After that, I called matchTemplate() to help me determine how each feature moved across the image. This worked relatively well. But, unfortunately, the points chosen at each frame were slightly different, and the order in which they occurred always differed. So, I struggled to get good results.

Overall, I think I prefer the method used in task 1, as it was faster and easier to use for me. Feature matching works as well. It is easy to do if the whole image is used to find features each time. It is likely more accurate and definitely much faster if a small window around each feature is used. But, it is hard to keep track of which points correspond to each other well without a lot of fine tuning and careful.

A video of the output from the best version can be found at:
https://youtu.be/sQA7CbY3nSY

Task 3

For this task, I used my cameras intrinsic and extrinsic parameters, along with the motion of the points to determine the fundamental, essential, rotation, and translation matrices between the frames. Undistorting the points led to quite a bit of errors. So, since my camera isn't too distorted anyway, I found it optimal to pass in the points as is to the OpenCV findFundamentalMat() function (the code for both is included). I found the essential matrix using the given formula (and normalized it after taking the SVD). Then, I used the recoverPose() function to find R and T. My results are shown below. The original image is first displayed. It is followed by the new image where points and lines are drawn to show how the image moved. Finally, for each type of movement, the matrices F, E, R, and T are shown.

The intrinsic parameters of my camera are given by the matrix:

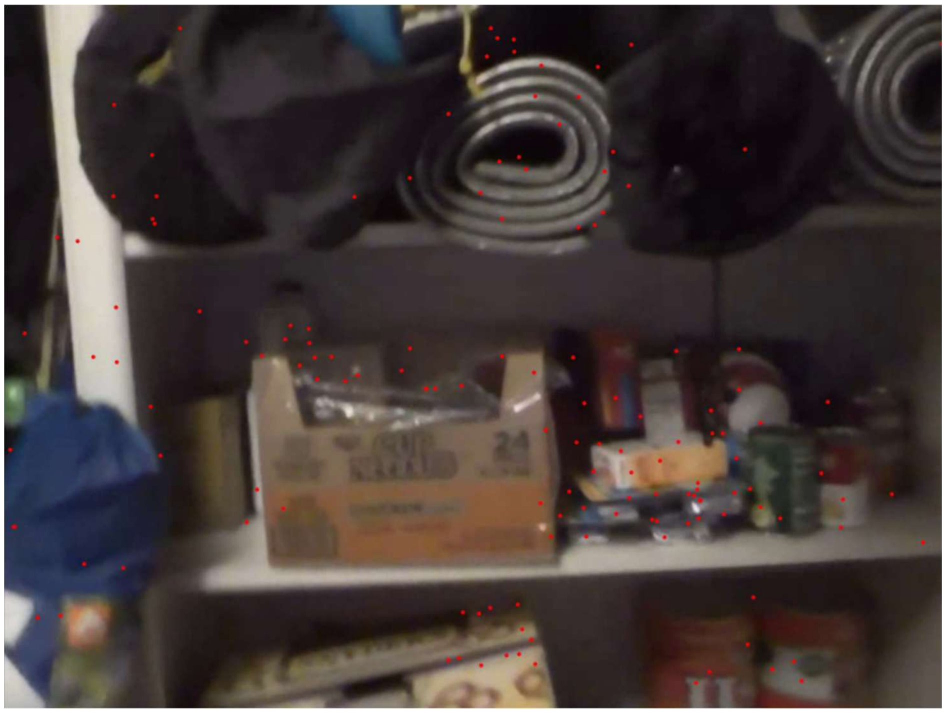
$$\begin{bmatrix} 667.804 & 0 & 326.097 \\ 0 & 639.903 & 245.018 \\ 0 & 0 & 1 \end{bmatrix}$$

The distortion parameters of my camera are:

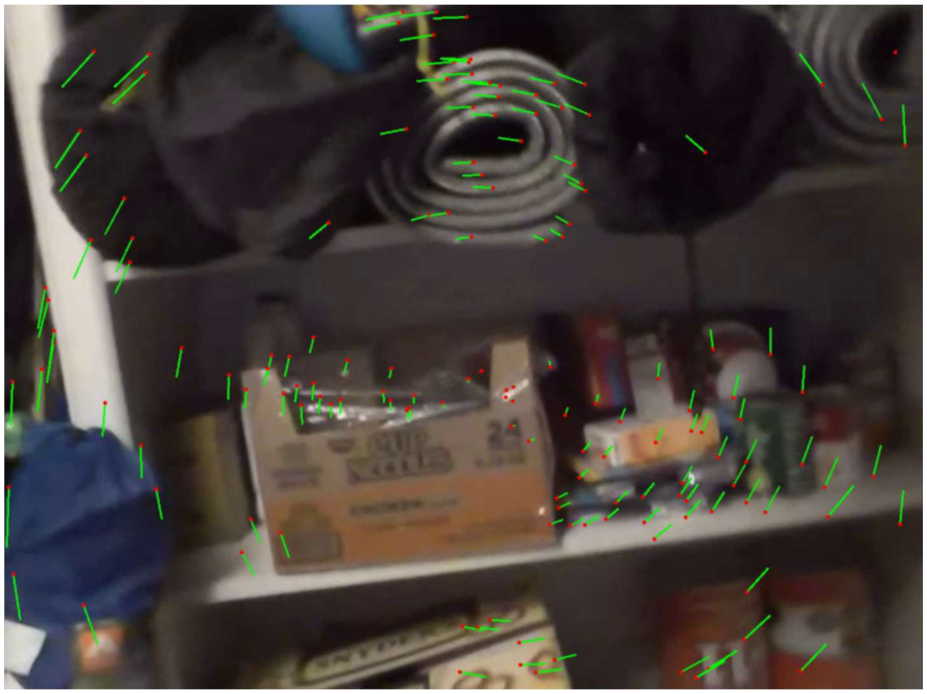
$$\begin{bmatrix} 0.08087 \\ -0.99228 \\ -0.00010 \\ 0.00203 \\ 2.50649 \end{bmatrix}$$

Rotation

Previous Image



Next Image (After 5 Frames)



Parameters

F

```
[[ 0.00000078  0.00000872 -0.00339552]
 [-0.00000869  0.00000077  0.00940939]
 [ 0.00174348 -0.01011769  1.        ]]
```

E

```
[[ 0.04510769  0.62977397 -0.09694109]
 [-0.64861197  0.09218581  0.75332567]
 [ 0.00405758 -0.76995031  0.04916544]]
```

R

```
[[ 0.997432   -0.06796399  0.02259003]
 [ 0.06925211  0.99565733 -0.06221443]
 [-0.01826358  0.06361907  0.99780712]]
```

T

```
[[0.76938447]
```

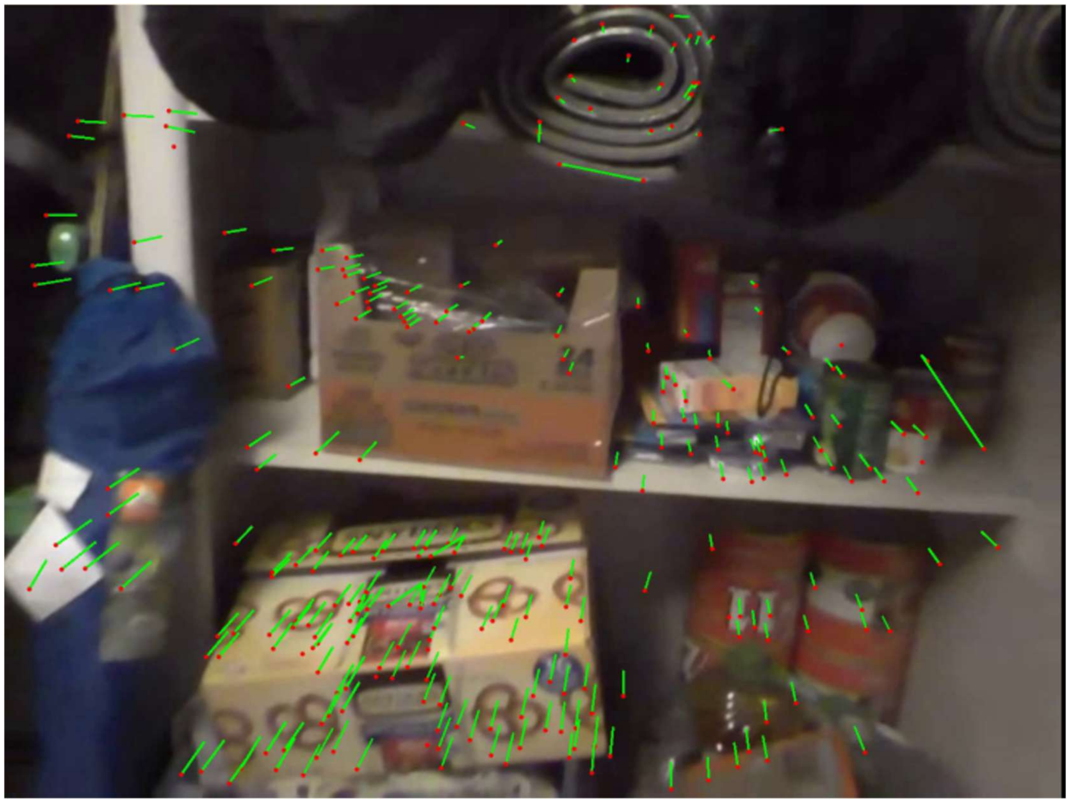
```
 [0.0574867 ]
 [0.636194  ]]
```

As expected, the values away from the diagonal in the rotation matrix are large (or at least larger than the other forms of movement). Unfortunately, the translation matrix is nonzero. This could either be due to error from the algorithm, or error of my own movement of the camera (as it is hard to rotate without also translating the camera).

Expansion

Previous Image

Next Image (After 5 Frames)



Parameters

F

```
[[-0.00000035  0.00038833 -0.06799465]
 [-0.00038767 -0.00000181  0.42684387]
 [ 0.0672344  -0.42580915  1.        ]]
```

E

```
[[-0.00179873  0.65298515  0.07243923]
 [-0.65140275 -0.0000967   0.75511226]
 [-0.0717897  -0.75411007  0.00197093]]
```

R

```
[[ 0.99999387  0.00296184 -0.00186799]
 [-0.00296638  0.99999265 -0.00243091]
 [ 0.00186078  0.00243643  0.9999953 ]]
```

T

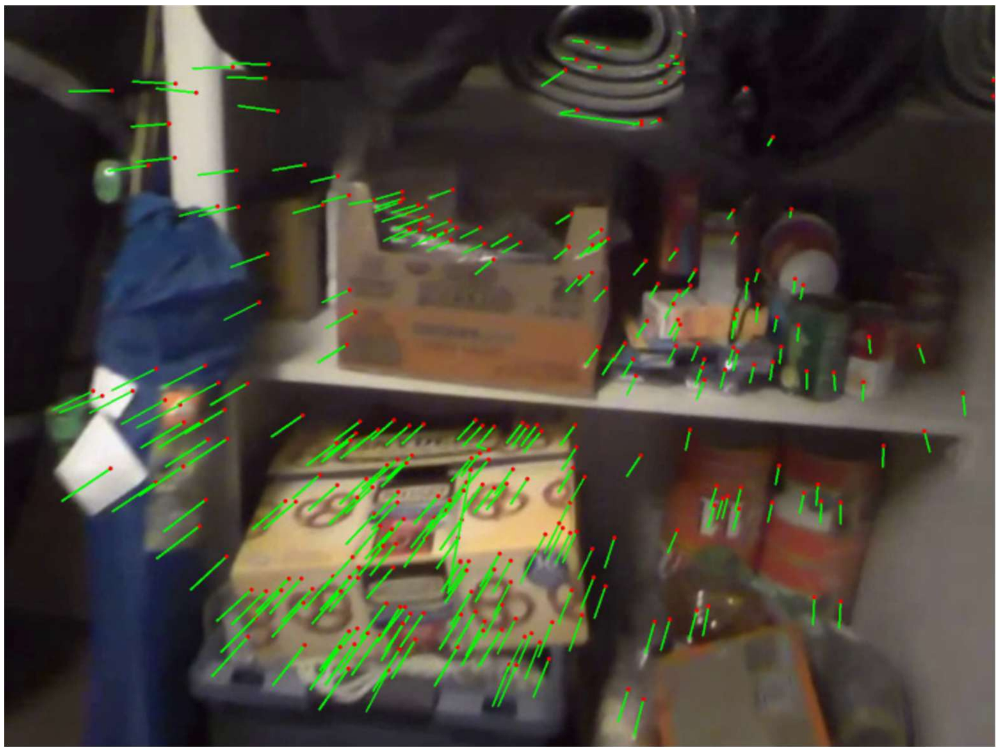
```
[[-0.75389636]
 [ 0.07402649]
 [-0.65280959]]
```

Here, we see that the movement nondiagonal terms in the rotation matrix are small. Also, the z-value in the translation vector is negative (suggesting that the camera is moving away from the scene). Once again, there seems to be a slight error in the x-translation, which probably occurred from user error in how I moved the camera.

Contraction

Previous Image

Next Image (After 5 Frames)



Parameters

F

```
[[-0.00000014  0.00012529 -0.02013027]
 [-0.00012499  0.00000027  0.12285768]
 [ 0.01977656 -0.12640419  1.        ]]
```

E

```
[[-0.00223276  0.69656023  0.09283563]
 [-0.70932625  0.00233342  0.69835846]
 [-0.0951597  -0.71152666  0.00106176]]
```

R

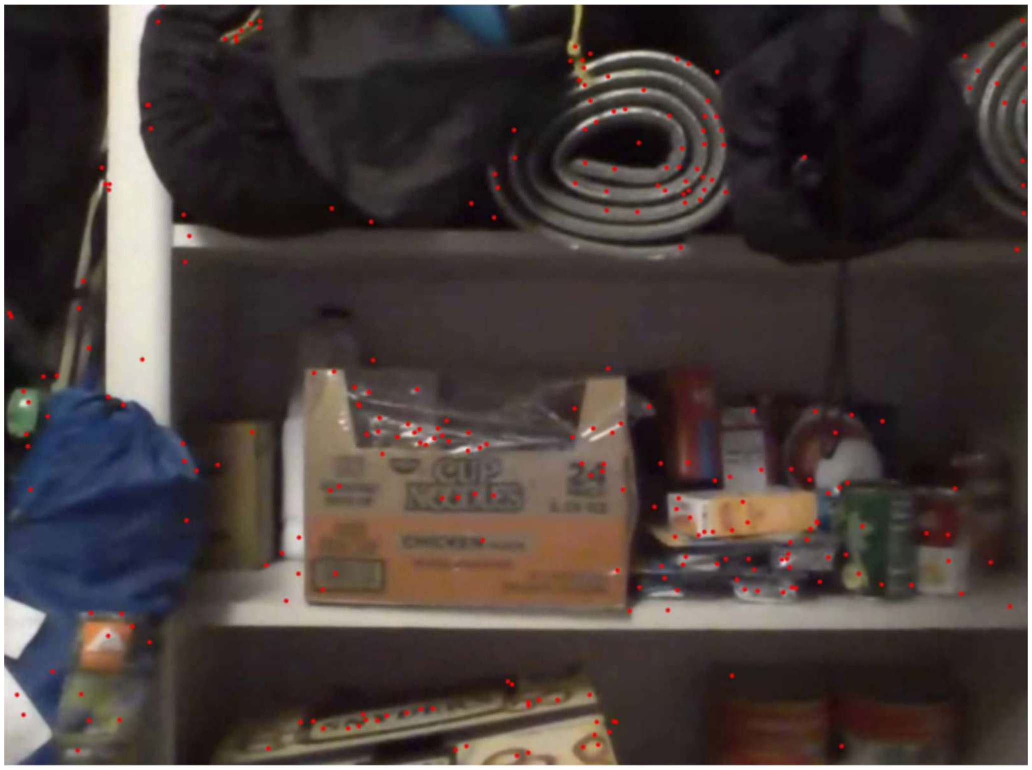
```
[[ 0.99982603  0.00072077  0.01863848]
 [-0.00064636  0.9999918  -0.00399778]
 [-0.01864121  0.00398503  0.9998183 ]]
```

```
T

[[ 0.71146356]

 [-0.0956362 ]

 [ 0.69618483]]
```

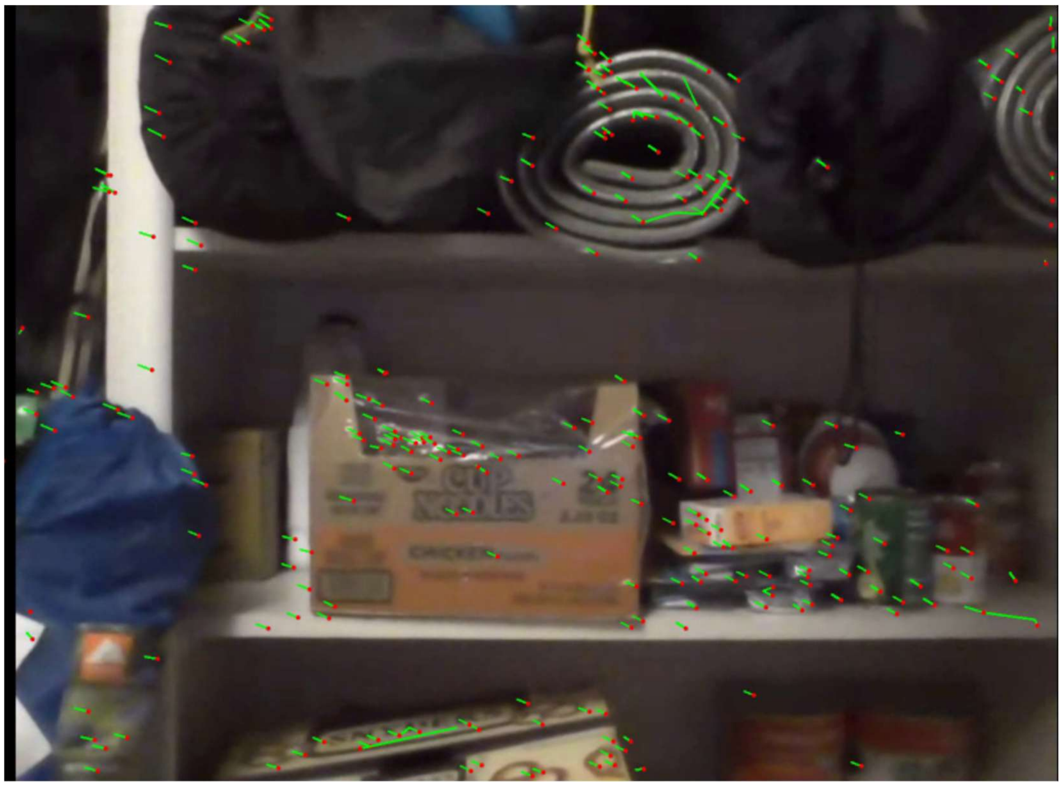
Here, we see that the movement nondiagonal terms in the rotation matrix are small. Also, the z-value in the translation vector is positive (suggesting that the camera is moving towards from the scene). Once again, there seems to be a slight error in the x-translation, which probably occurred from user error in how I moved the camera.

Translation (X-Direction)

Previous Image



Next Image (After 5 Frames)



Parameters

F:

```
[[-0.00000003 -0.00036362  0.16387009]
[ 0.00036345  0.00000058 -0.46191728]
 [-0.16626546  0.46653242  1.       ]]
```

E

```
[[-0.00180592 -0.56302466  0.18338051]
 [ 0.56672159  0.00398524 -0.80247434]
 [-0.18566684  0.80605962  0.00360049]]
```

R

```
[[ 0.99998159 -0.00127527  0.00593237]
 [ 0.00124061  0.99998216  0.00584224]
 [-0.00593971 -0.00583477  0.99996534]]
```

T

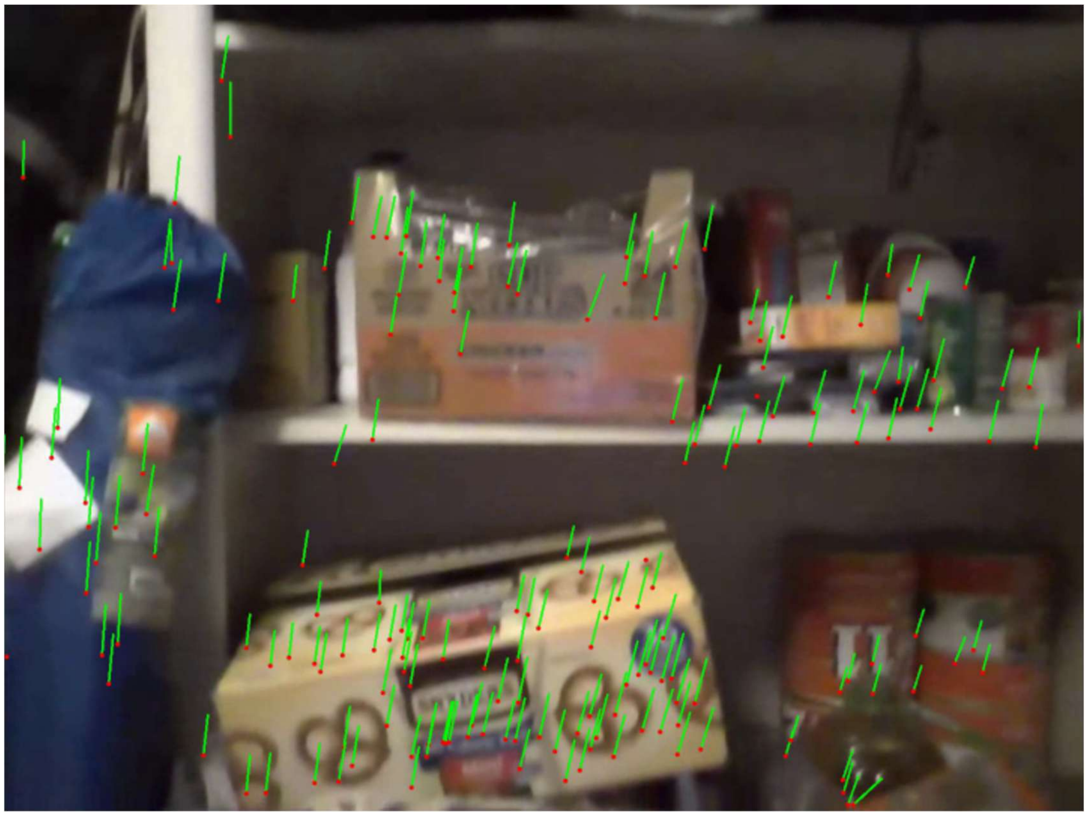
```
[[0.80583594]
 [0.18667   ]
 [0.5619455 ]]
```

Here, we see that the movement nondiagonal terms in the rotation matrix are small. Also, the x-value in the translation vector is positive (suggesting that the camera is to the right). However, there seems to be a slight error in the z-translation, which probably occurred from user error in how I moved the camera.

Translation (Y-Direction)

Previous Image

Next Image (After 5 Frames)



Parameters

F

```
[[ 0.00000018  0.00003741 -0.19389809]
 [-0.00003995 -0.00000104 -0.03408294]
 [ 0.19509634  0.03699467  1.        ]]
```

E

```
[[ 0.00744709  0.12666755 -0.96376235]
 [-0.13079735 -0.00281129 -0.23583451]
 [ 0.96060612  0.24395375  0.00711909]]
```

R

```
[[ 0.99991271  0.00967525  0.00899805]
 [-0.00961617  0.99993207 -0.00658653]
 [-0.00906116  0.00649943  0.99993782]]
```

T

```
[[-0.23465294]
 [ 0.96294664]
 [ 0.13293518]]
```

Here, we see that the movement nondiagonal terms in the rotation matrix are small. Also, the y-value in the translation vector is positive (suggesting that the camera is up). However, there seems to be a slight errors in the x and z translations, which probably occurred from user error in how I moved the camera.

In all cases, it should be noted that the translation can only be given up to a scale factor. We do not know how the units transfer over to real measurements.