CMPEN 454 Project 2

Jasmine Khalil, Hailey Bickel, Madeleine Hill

Results for Task 3.1 - Understanding Pinhole Camera Model Parameters

- The internal parameters are the focal length (foclen), principal point (prinpoint), aspect ratio (aspect ratio), and skew.
- The external parameters are the unit quaternion (orientation), rotation matrix (Rmat), and camera position (position).
- The internal parameters that combine to form the matrix Kmat are the focal length, principal point, aspect ratio, and skew.
- The external parameters that combine to form the Pmat matrix are the rotation matrix (Rmat) and the camera position in world coordinates (position).
- The location of camera 1 is: The location of camera 2 is:

```
Camera 1 Location in World Coordinates:
   1.0e+03 *

   -3.7407
   -2.1432
    5.9685
```

```
Camera 2 Location in World Coordinates:
   1.0e+03 *

    4.2103
   -1.7134
    5.6732
```

We found these locations by multiplying the negative transpose of the respective rotation matrices by the camera's provided position following this equation derived during lecture, $C = -R^T * T$:

$$R(P_W - C) = RP_W - RC = RP_W + T$$
$$\rightarrow -RC = T$$
$$\rightarrow C = -R^T T$$

We verified that the camera's location and the rotation matrix combine in the expected way to yield the appropriate entries in Pmat by running the code in task3_1.mat to construct appropriate Rmat and camera location matrices that match those in the pinhole camera model.

```
% Creating the camera location matrix that we will combine with the rotation matrix to produce Pmat
C = position(:);
camera_location_matrix = [1 0 0; 0 1 0; 0 0 1];
camera_location_matrix = [camera_location_matrix -C; [0 0 0 1]];

% Turning given Rmat into 4x4 matrix
Rmat = [Rmat [0; 0; 0]; [0 0 0 1]];

% Constructing the full projection matrix
P_constructed = [1 0 0 0; 0 1 0 0; 0 0 1 0] * Rmat * camera_location_matrix;
```

CMPEN 454 Project 2

Then, we compared the results between the given and constructed Pmats for both camera views, which gave the correct results:

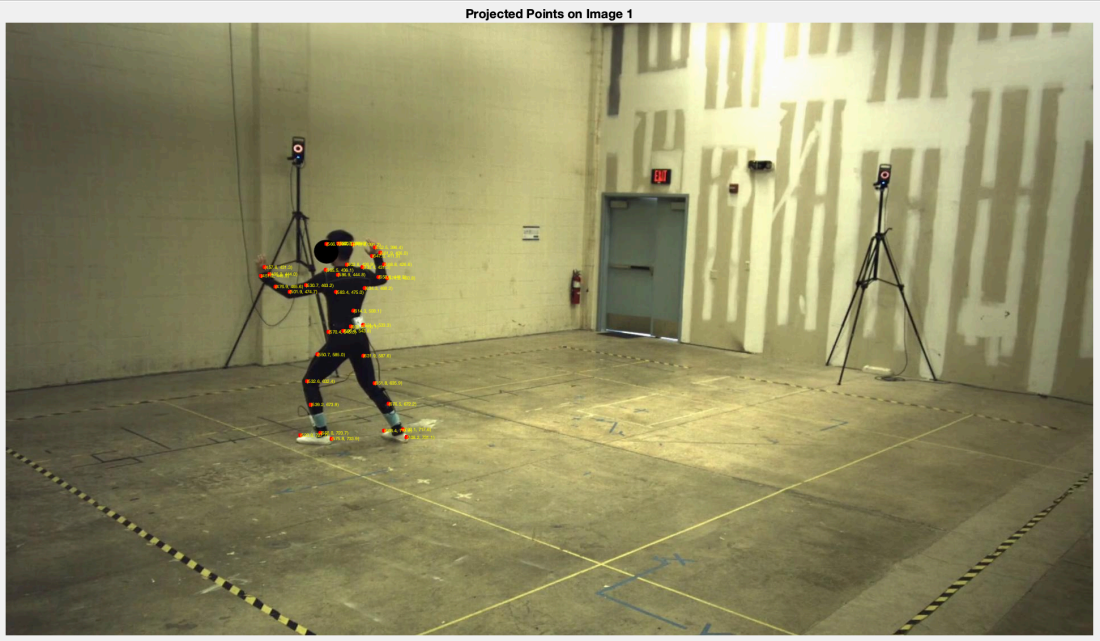
```
Constructed projection matrix P:
   1.0e+03 *

   -0.0007   -0.0007    0.0000    0.4059
   -0.0001    0.0001   -0.0010    0.7375
    0.0007   -0.0007   -0.0002    7.3144

Provided projection matrix Pmat:
   1.0e+03 *

   -0.0007   -0.0007    0.0000    0.4059
   -0.0001    0.0001   -0.0010    0.7375
    0.0007   -0.0007   -0.0002    7.3144

Projection matrix Pmat matches the constructed matrix P_constructed.
```

```
Constructed projection matrix P:
   1.0e+03 *

   -0.0009    0.0005    0.0000    0.9129
    0.0001    0.0002   -0.0010    0.4181
   -0.0005   -0.0008   -0.0002    7.2000

Provided projection matrix Pmat:
   1.0e+03 *

   -0.0009    0.0005    0.0000    0.9129
    0.0001    0.0002   -0.0010    0.4181
   -0.0005   -0.0008   -0.0002    7.2000

Projection matrix Pmat matches the constructed matrix P_constructed.
```

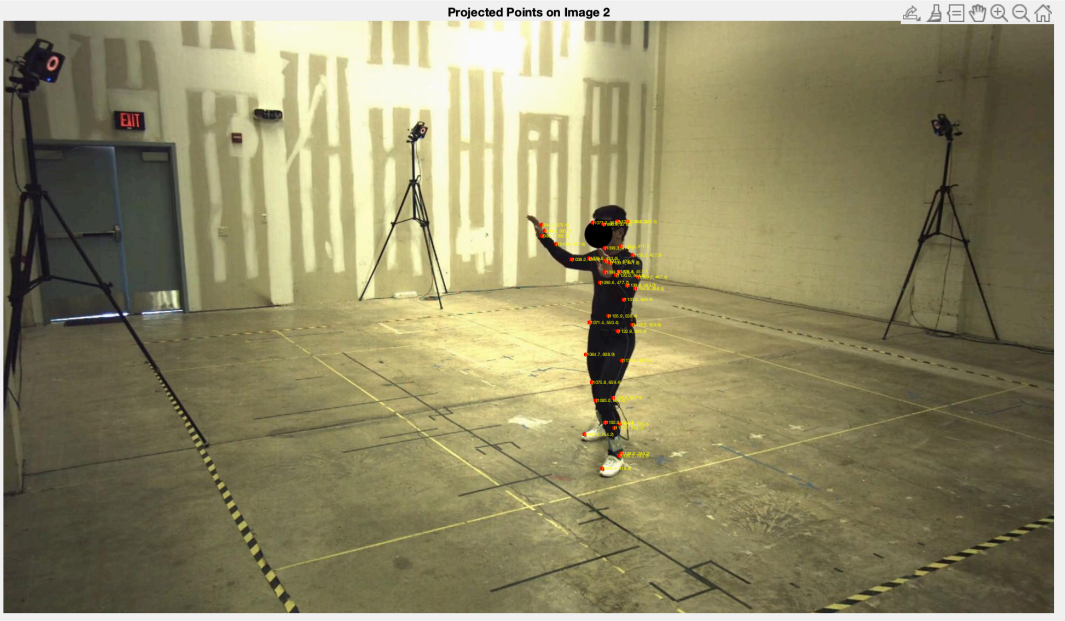
Results for Task 3.2 - Projecting 3D Mocap Points into 2D Pixel Locations

For this task, we referred to the lecture notes on the transformation chain that maps 3D world coordinates into 2D pixel coordinates. To visualize the projected 2D points, we plotted each 2D point pair as red dots with their numeric coordinates in yellow.

The plotted points line up very well with the person's body, indicating that the projection was accurate. Our function project_3D_to_2D takes in the camera parameters: Kmat, Rmat, and position for each view, uses the number of points (39), and projects each of these points by transforming the 3D point to camera coordinates then from camera to image plane coordinates, and finally normalizes the points to get the final 2D point for all 39 3D mocap points. The visual results are below.



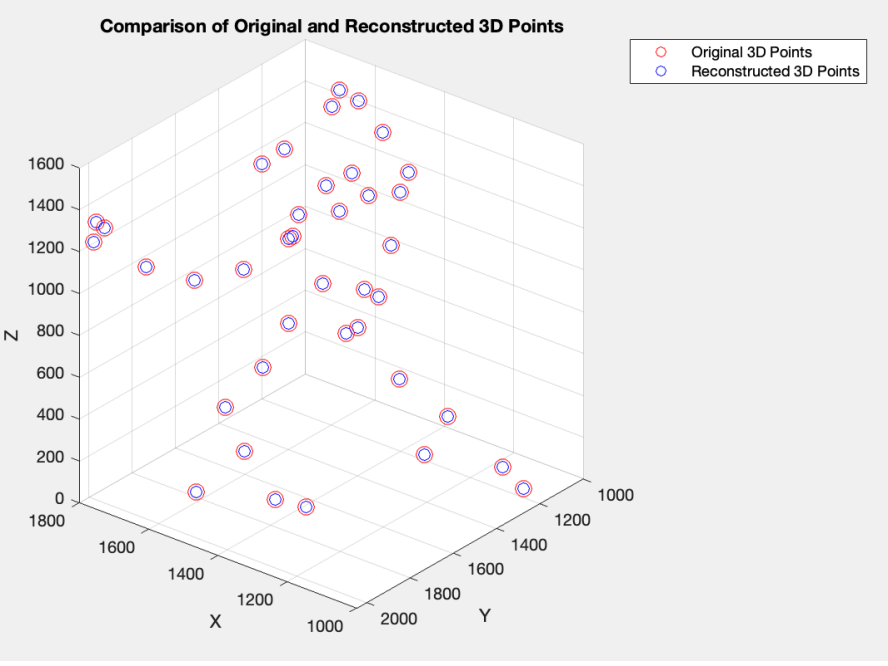
CMPEN 454 Project 2



Results for Task 3.3 - Triangulation to Recover 3D Mocap Points from Two Views

For this task, we referred back to the lecture notes on triangulation. We first converted each 2D point into a ray passing through that point using our get_ray_from_pixel function. Next, we used our triangulate_point function to find the direction of the cross-product of the two rays from each camera view, compute the closest points on each ray, and get the final 3D point as the midpoint between the two closest points.

To ensure that we had the correct result, we applied our result to all 39 mocap points that it projected, and we found that our set of reconstructed 3D points was very close to the original set. We plotted a comparison between the actual 3D points and the reconstructed 3D points, showing that both sets of 3D points are very similar. The resulting mean squared error was also very small.



CMPEN 454 Project 2

The mean squared error calculated between the actual 39 mocap points and the reconstructed mocap points:

```
Mean Squared Error (MSE) between original and reconstructed points:
   2.4554e-24
```

Results for Task 3.4 - Triangulation to Make Measurements about the Scene

For this task, we used a slightly different version of our triangulate_point function where the user selects the points to be triangulated instead of pre-determined points. To do this, we followed a method similar to the plane warp demo on Canvas using ginput(). We also created a function to calculate the equation of the planes for the floor and wall that takes in a 3x3 input where each row represents the coordinates of a 3D point and returns the coefficients for an equation of a plane: Ax + By + Cz + 1 = 0.

3D locations of 3 points on the floor:

```
floor_points =

   1.0e+03 *

   -1.6079    1.9557    0.0060
    2.2527    1.9852   -0.0247
    2.3343   -2.2732   -0.0085
```

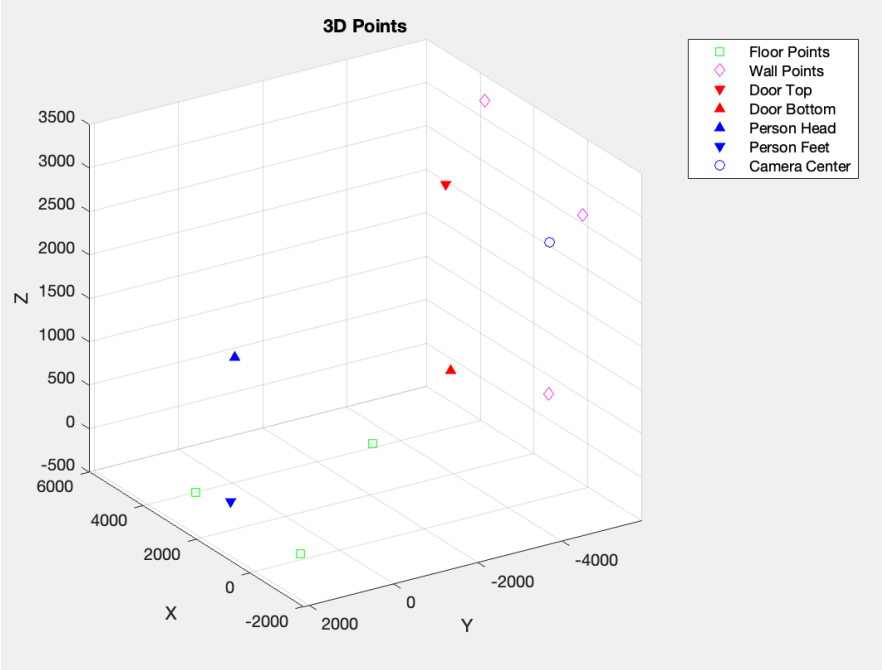
This is where we had some trouble. The 3D points found seem accurate because all of the Z values are relatively small, so the points must all lie close to the Z=0 plane compared to the X=0 and Y=0 planes. However, when we calculated the plane equation for the floor, we ended up with this result:

```
Floor plane equation:
1.032520e-02x + 5.628039e-03y + 1.828790e+00z + 1 = 0
```

The floor plane is not the Z=0 plane.

To try to understand why this was happening, we plotted all of the 3D coordinates of all the points we selected to see if there was something wrong with our coordinate system, but all of the points accurately reflect the actual scene in the images. We also thought this could be because of poorly selected points, so we repeated the process multiple times with different combinations of points on the floor and took extra care when picking corresponding points in the second camera view, but the result hardly changed.

CMPEN 454 Project 2



From the 3D reconstruction of the scene above, all the points look like they are accurately located relative to the actual scene.

3D locations of 3 points on the wall:

```
wall_points =

   1.0e+03 *

    3.2458   -5.5056    3.3711
   -0.4545   -5.4808    2.7700
    0.9169   -5.5476    0.4395
```

Plane equation of the wall plane:

```
Wall plane equation:
-2.295205e-07x + 1.807108e-04y + -1.464643e-06z + 1 = 0
```

The doorway is 2146.36 units tall:

```
Height of the doorway: 2146.36 units
```

The person is 1632.45 units tall:

```
Height of the person: 1632.45 units
```

3D location of the center of the camera:

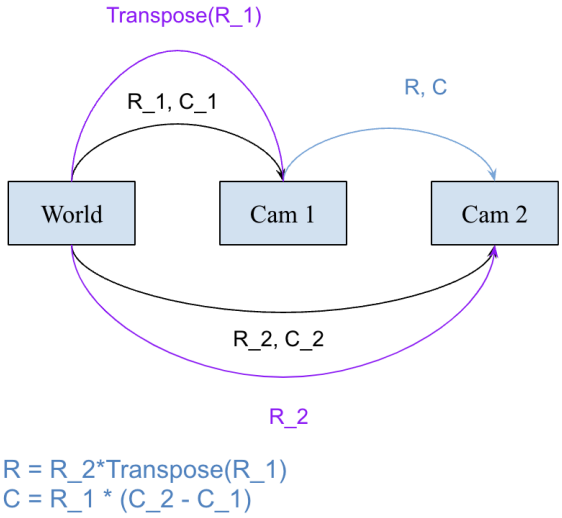
```
3D location of the tripod camera: (-46.96, -4938.97, 2439.29)
```

Results for Task 3.5 - Fundamental Matrix from Known Camera Calibration Parameters

From some mathematical derivation, we computed the relative rotation between the two camera views to be $R = R_2 * R_1^T$ and the location of camera2 with respect to the coordinate system of camera1 to be $C = R1 * (C2 - C1)$ where C2 and C1 are the positions of camera two and camera one, respectively, with respect to the world coordinate system. Then, we formed the skew matrix using our results from T:

```
T_skew = [0 -C(3) C(2);
       C(3) 0 -C(1);
    -C(2) C(1) 0];
```

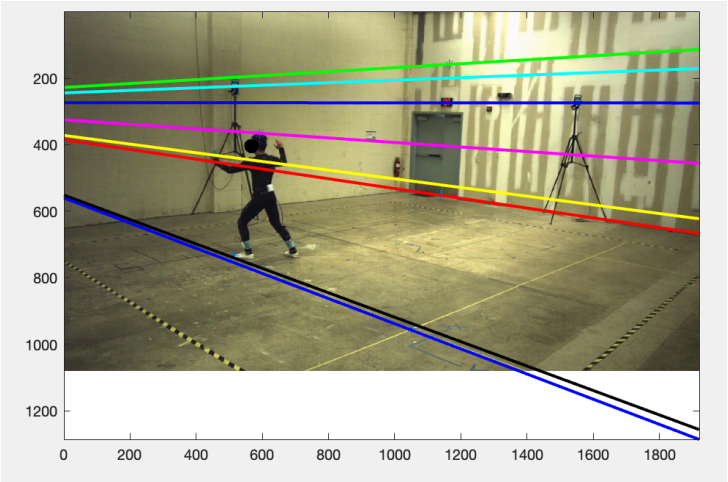
Then, we calculated the essential matrix: $E = R * T_{skew}$ from which we calculated the fundamental matrix by multiplying E by the Kmat matrices from both views: $F = inv(K2)' * E * inv(K1)$. Below is a visual explanation of the equations we developed for R and T.

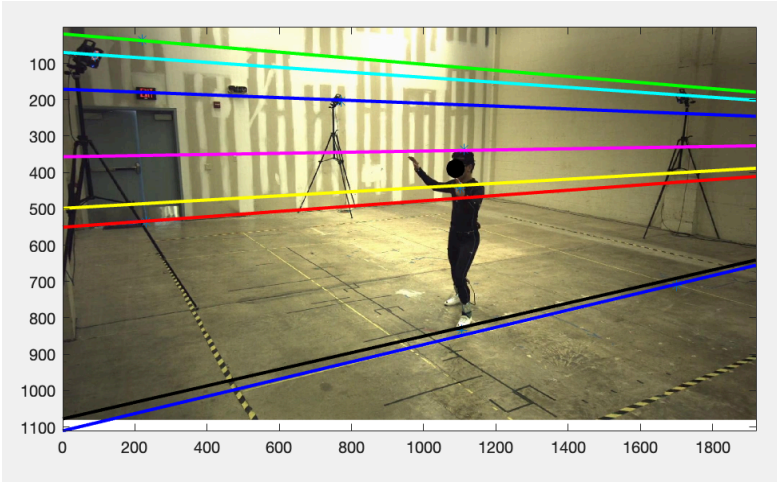


For consistency, we chose the same points in tasks 3.5 and 3.6. (We got the coordinates of the points we clicked on for 3.6 and used them in 3.5).

Below are the resulting plots from mapping points in images 1 and 2 into epipolar lines in images 2 and 1, respectively. The epipolar lines in both definitely seem to accurately capture where the other camera is in

each view (the epipoles). We implemented the equations for the epipolar lines: $l2 = F * P_1$, $l1 = P_2' * F$ into our code and used the same plotting code from the eight point algorithm code on Canvas.





The fundamental matrix calculated from our mathematical derivations and code for this task is:

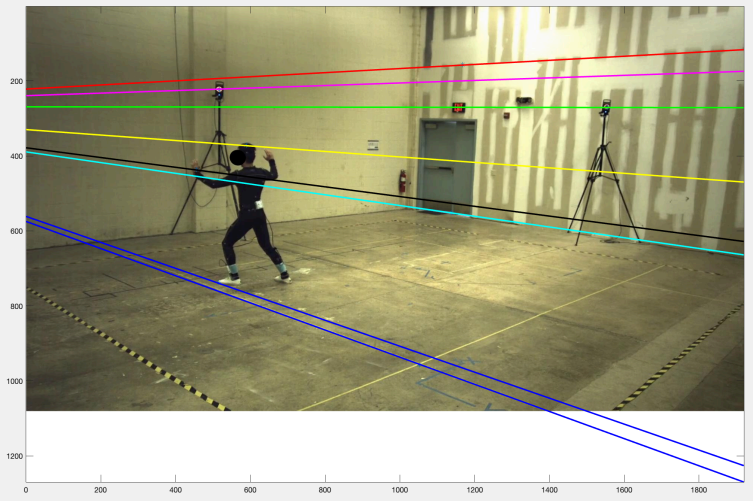
```
Fundamental Matrix for task 3.5:

F =

   1.0e+03 *

   -0.0000    0.0000   -0.0006
    0.0000    0.0000    0.0019
   -0.0005   -0.0069    1.5304
```

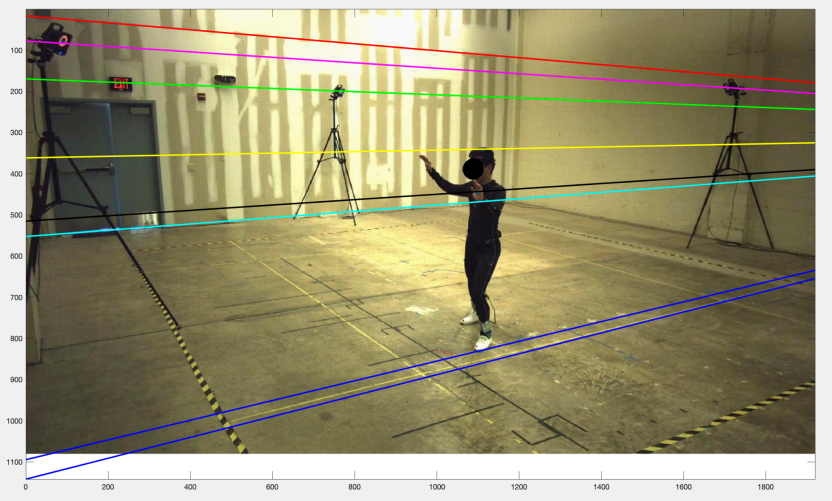
<u>Results for Task 3.6</u> - Fundamental Matrix Eight-Point Algorithm with and without Hartley Preconditioning

1. **With Hartley Preconditioning:**

   Using the eight-point algorithm we demo'ed in class, we adjusted the code to calculate the fundamental matrix from user-selected points (as we did in task 3.4) instead of pre-determined points. We then selected 8 points spread across the two image views, and the Fundamental matrix was calculated to be:

```
>> task3_6_1
-0.000291366 0.00743117   -2.31493
0.00991994 0.00222317    6.87352
   -1.9025   -26.6075     6061.5
```

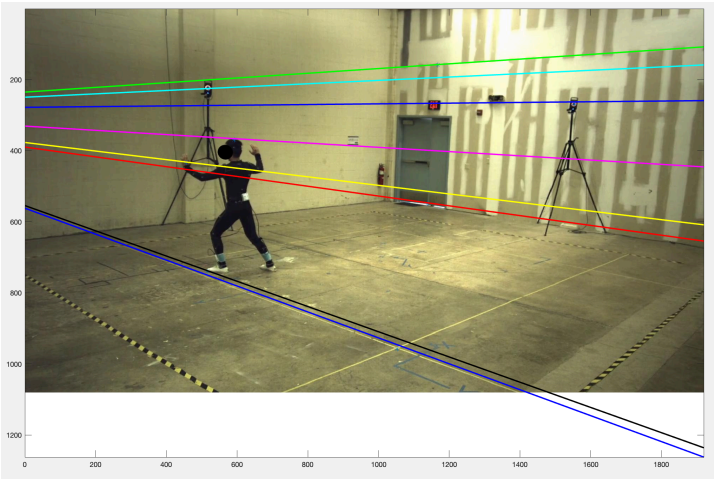


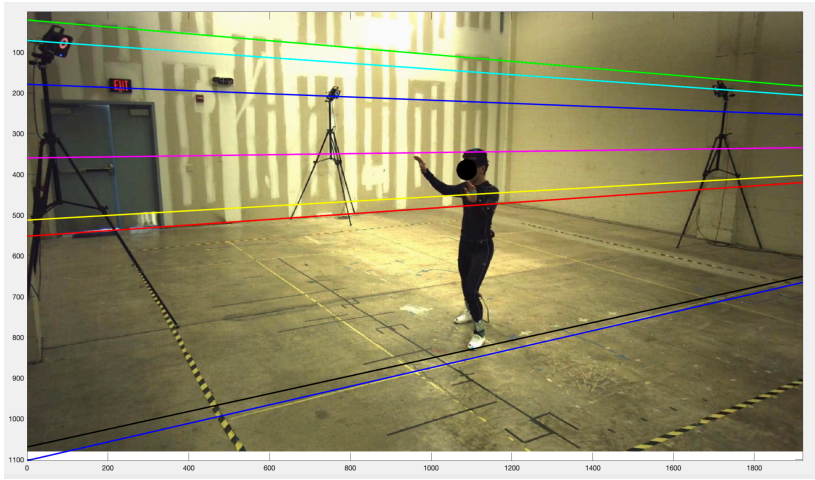


The epipolar lines definitely seem reasonable and accurately match those from task 3.5. Looking closely, there are slight gaps between the points and the epipolar lines that are supposed to go through them, indicating that there still is some slight error - this can be improved by carefully selecting points in both views.

2. Without Hartley Preconditioning:

Next, we modified the eight-point algorithm code not to do Hartley preconditioning by removing the effects of Hartley's normalization procedure where instead of using nx1, ny1, nx2, ny2, we used x1, y1, x2, y2 to build matrix A without normalization. The second thing we adjusted was to remove the part of the code that does the 'unnormalization' of F. We then ran the code using the same points collected from the previous step (eight-point with Hartley preconditioning) and the same input images. Below are the epipolar lines plotted and the fundamental matrix printed:





```
>> task3_6_2
F matrix (no normalization):
-0.000546601   0.0122392    -3.90872
 0.0162005   0.0036418     11.2454
  -3.19866    -43.5228      9999.9
```

The epipolar geometry still looks reasonable, but there are slightly larger gaps between the epipolar lines and the points selected compared with the epipolar geometry from the eight-point algorithm with Hartley preconditioning.

Results for Task 3.7 - Quantitative Evaluation of Estimated F Matrices

Using the set of 39 accurate 2D point matches from task 3.2, where one pair of those coordinates is (x1,y1) in image 1 and (x2,y2) in image 2, we computed the epipolar line in image 2 from (x1,y1) and computed the squared geometric distance of the point (x2, y2) from that line using the following equation:

$$Squared\ Geometric\ Distance\ =\ (ax\ +\ by\ +\ x)^2 / (a^2\ +\ b^2)$$

Repeating this process by mapping (x2,y2) in image 2 into an epipolar line in image 1 and measuring the square distance of (x1,y1) to that line, we accumulated all of these distances over the 39 3D point matches to get the SED error.

For each of the fundamental matrices from tasks 3.5, 3.6 (with Hartley preconditioning), and 3.6 (without Hartley preconditioning), we called our function symmetric_epipolar_distance which returns the SED error. Our results are summarized below.

1. SED for F from Task 3.5:

```
Symmetric Epipolar Distance (SED): 0.000000
```

2. SED for F from Task 3.6 with Hartley Preconditioning:

```
Symmetric Epipolar Distance (SED): 9.454294
```

3. SED for F from Task 3.6 without Hartley Preconditioning:

```
Symmetric Epipolar Distance (SED): 12.919607
```

As expected, our SED error for F from task 3.5 was very small compared with the error of either of the F matrices computed using the eight-point algorithm. Removing the preconditioning seemed to have a significant effect on the SED error because with preconditioning, the SED error = 9.5. Without preconditioning, SED error = 12.9 - approximately a difference of 3.4.

CMPEN 454 Project 2

Extra Credit 1 - Modify F for Cropped Views:

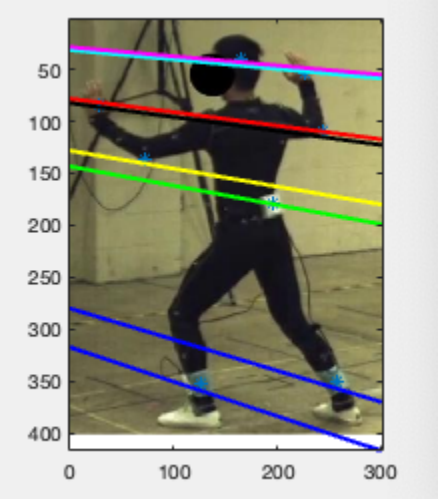
The equation for the fundamental matrix above is: $F = inv(K2)' * E * inv(K1)$ and because the images are now cropped, the Kmat matrices for both camera views have changed, changing F. Firstly, we played around with the cropping parameters: x_start, y_start, width, height for each of the images such that the images focus tightly around the person. Next, we set the points to be user inputted as in tasks 3.4 and 3.6.1. Then, we called our processing_cropped_views function to adjust the Kmats as:

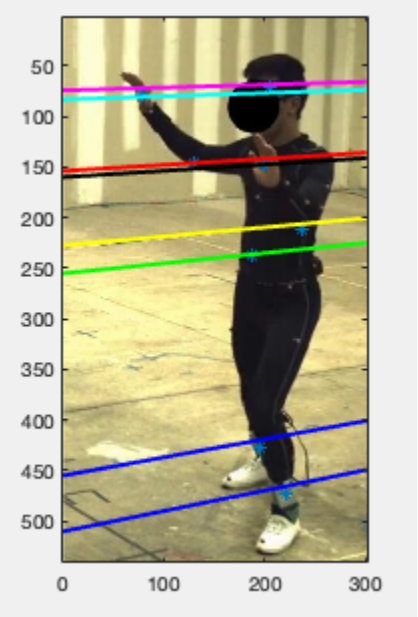
```
K1_cropped = K1_original;
K1_cropped(1, 3) = K1_original(1, 3) - x1_start;
K1_cropped(2, 3) = K1_original(2, 3) - y1_start;

K2_cropped = K2_original;
K2_cropped(1, 3) = K2_original(1, 3) - x2_start;
K2_cropped(2, 3) = K2_original(2, 3) - y2_start;
```

R and T in this task have not changed, so E has also not changed. Using the equation for F and the updated Kmat matrices, we printed the new fundamental matrix to be:

```
Updated Fundamental Matrix for Cropped Views

F_cropped =

   -0.0001    0.0020    0.0362
    0.0026    0.0003    3.1317
    0.2383   -4.9820 -100.5067
```

Using the same code for calculating and plotting the epipolar geometry as in the previous tasks, we plotted the epipolar lines, which definitely look accurate relative to the positions of the camera and the previous epipolar line plots:





CMPEN 454 Project 2

Project Contributions:

- Jasmine: 3.1,3.2,3.3,3.4,3.5,3.6,3.7, extra credit 1 coding tasks, and written report.
- Madeleine: helped with the written report and report structure.
- Hailey: helped with 3.1, 3.2, 3.3, 3.4, and worked on the written report.