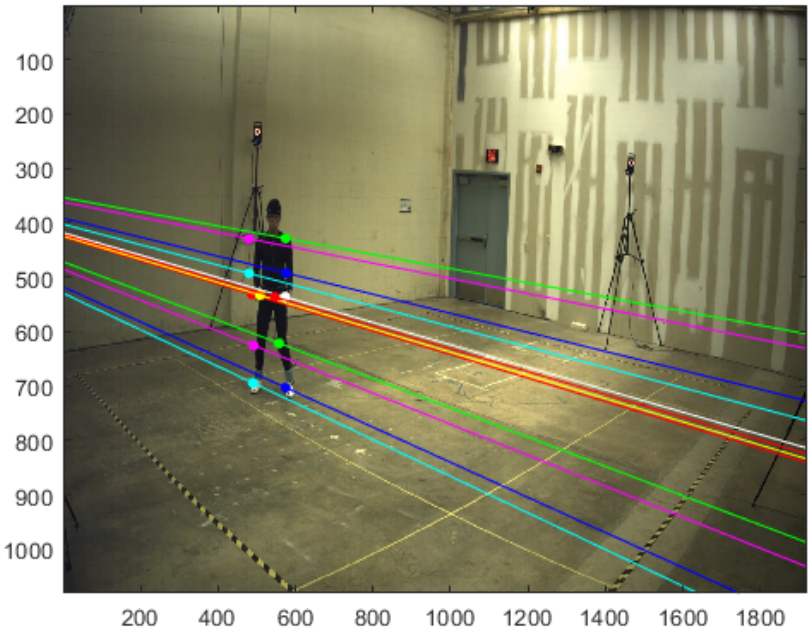
PennState
College of Engineering

CMPEN/EE 454, Project 2, Fall 2020

3D RECONSTRUCTION FROM 2D IMAGE

Project Report

24 October 2020



Submitted to: Robert Collins

Submitted by: Dawang Shen

Zhenghan Fang

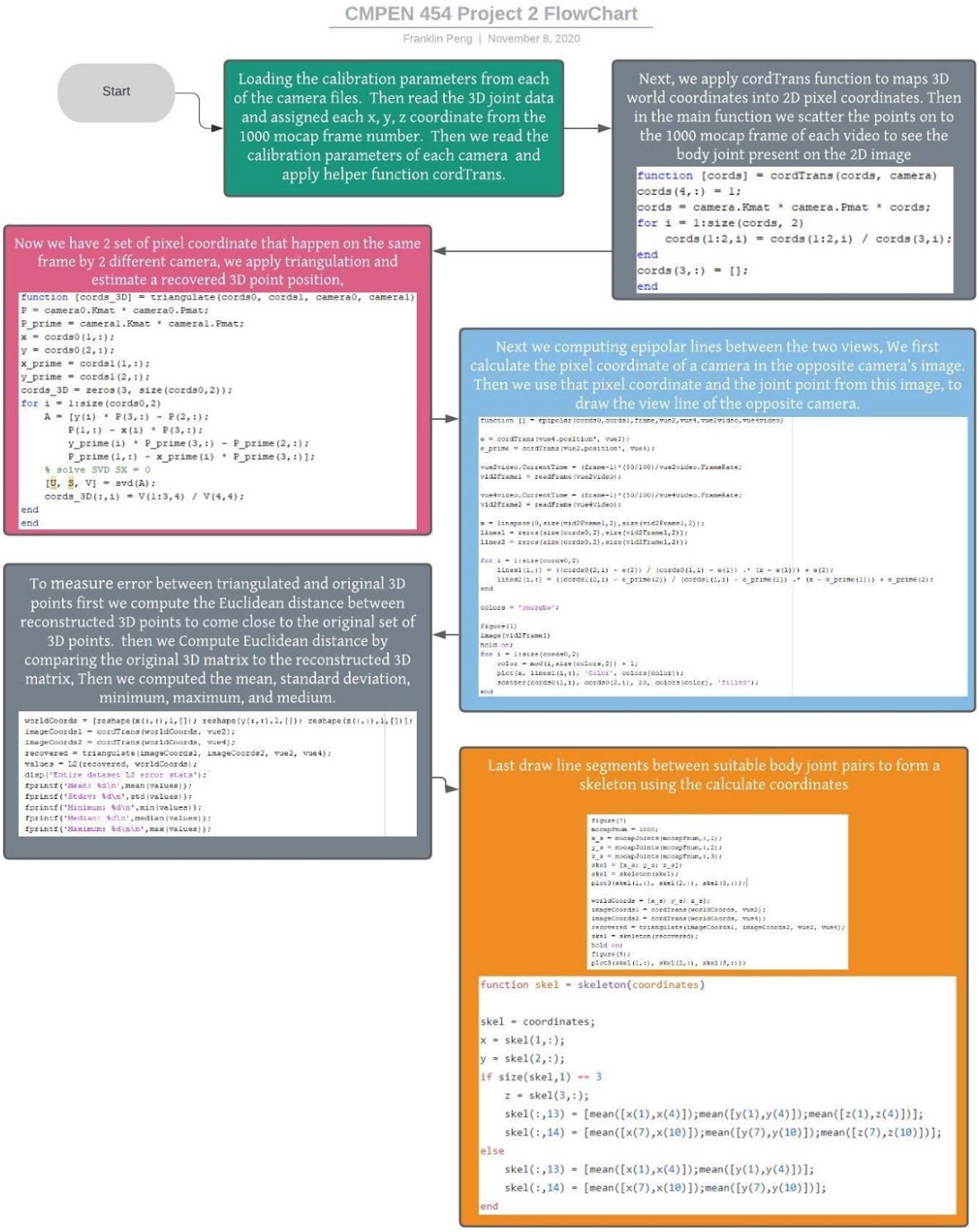
Zekun Peng

TABLE OF CONTENTS

1.0 INTRODUCTION

The goal of this project is to implement a forward (3D point to 2D point) and inverse (2D point to 3D ray) camera projection by performing various image process techniques. This project involved understanding relationships between 2D image coordinates and 3D world coordinates and the chain of transformations that make up the pinhole camera model that was discussed in class. Our tasks were to project 3D coordinates (sets of 3D joint locations on a human body, measured by motion capture equipment) into image pixel coordinates that you can overlay on top of an image, to then convert those 2D points back into 3D viewing rays, and then triangulate the viewing rays of two camera views to recover the original 3D coordinates you started with (or values close to those coordinates). By utilizing the course material and variety of functions from Matlab we were able to create such image processing tools. There are three main functions we implemented, project3DTo2D, reconstruct3DFrom2D, and findEpipolarLines, with all these functional features we are able to make conversion between 3D coordinate and 2D coordinate, and with the help of skeleton functional a fully reperstatable skeleton of the subject can be observed.

2.0 PROBLEM STATEMENT



2.1 OUTLINE

To start the code, we load the calibration parameters from each of the camera files. Then read the 3D joint data and assigned each x, y, z coordinate from the 1000 mocap frame number. Then we read the calibration parameters of each camera.

2.1.1 Analyze the camera parameter

In order to project a 3D point into a 2D pixel location, we need to understand each parameter of the camera.

vue2.Kmat

1	2	3
1.5578e+03	0	976.0397
0	1.5578e+03	562.8225
0	0	1

First, we quickly realize that the Kmat parameter has some unique identity: it is a 3x3 matrix whose last row is [0, 0, 1], and since the first row first column is a number that is greater than 0, we determine that the Kmat parameter is a combination of the intrinsic parameters with the perspective projection matrix.

Film plane to pixels | Perspective projection

$$\begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Therefore the Kmat represents the internal parameters, which include all other internal parameter foclen (focal length), prinpoint (principle point), aspect ratio, and skew.

Then we identify all the other external parameters, including orientation, radial, and position. Then we realized that Rmat is a submatrix of Pmat and when we get 1 when calculating the determinant of Rmat. Thus Rmat is the rotational matrix and Pmat is the combination matrix of External Parameters

vue2.Rmat

1	2	3
-0.7593	-0.6491	0.0468
-0.1381	0.0904	-0.9863
0.6360	-0.7553	-0.1583

```
>> det(vue2.Rmat)

ans =

    1.0000
```

vue2.Pmat

1	2	3	4
-0.7593	-0.6491	0.0468	137.7154
-0.1381	0.0904	-0.9863	805.5227
0.6360	-0.7553	-0.1583	7.3365e+03

After we figure out the camera parameter we can project 3D points into 2D pixel locations. Here we use a little helper function Cordtrans

2.1.2 Project 3D to 2D Function

In the project 3D to 2D functions we just simply follow the Pinhole Camera Model, multiply the Kmat, Pmat, and camera coordinates to calculate the pixel location.

Pixel location | Film plane to pixels | Perspective projection | World to camera | World point

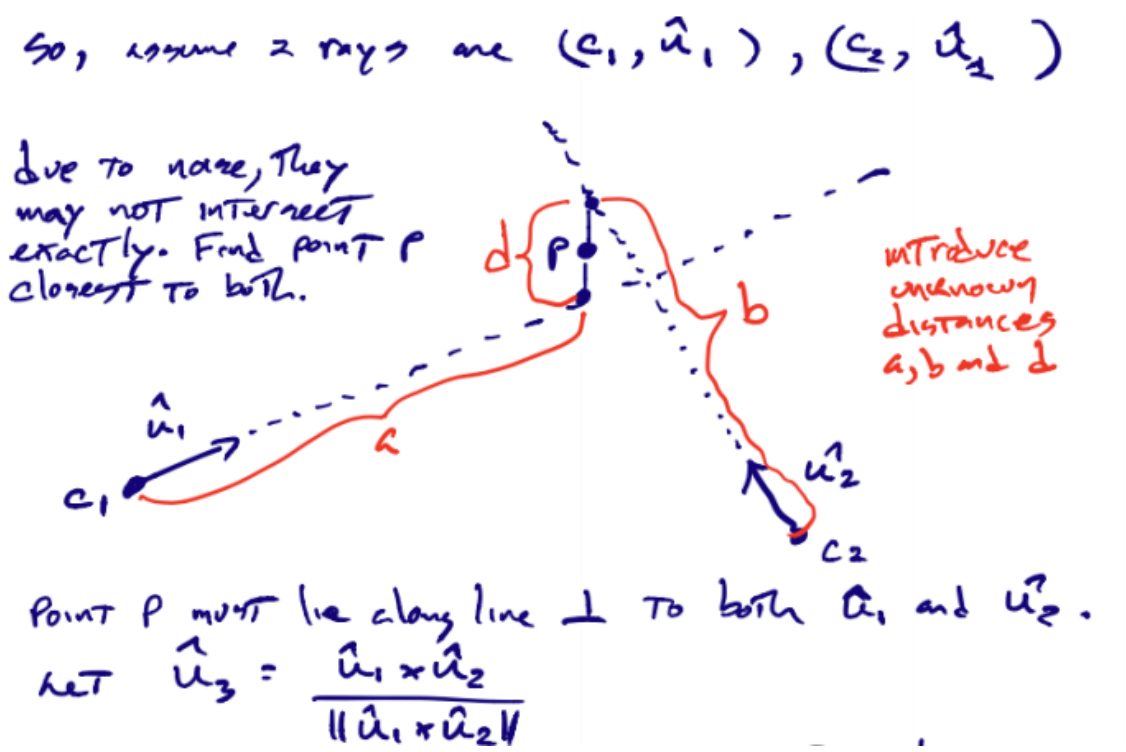
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$$

2.1.3 Triangulation back into a set of 3D scene points(reconstruct3DFrom2D)

We now have two sets of corresponding 2D pixel positions in the two camera views. We must triangulate each of the 12 pairs of 2D points to estimate the recovered 3D point position. To do this, We did some research and decided to use a method presented by Carnegie Mellon University:

> Kitani, Kris. “Triangulation 16-385 Computer Vision (Kris Kitani) Carnegie Mellon University.” *16-385 Computer Vision, Spring 2017*, 2017, www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf.

In his lecture, Kitani states that if we know a given set of (noisy) matched points {x_i, y_i}and two camera matrices {P1, P2}, we can estimate the 3D point since there are two correspondences. However, since the points are usually noisy we are usually unable to directly compute a point from the calculation, but we can still use the best fit to estimate the location.



In the lecture, the slide also mentions that we can find a point P which lies along the perpendicular line to both points. Thus it means if we compute the cross product of the two camera matrices in the Pinhole Camera Model, we are able to get a new matrix of the perpendicular line.

Concatenate the 2D points from both images

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In the end, Kitani states that to solve the homogeneous linear system we can either find the eigenvector corresponding to the smallest eigenvalue of (A^T)A or we can use

Singular value decomposition to return the column of V corresponding to the smallest singular value as the result.

Minimize $\|\mathbf{A}\boldsymbol{x}\|^2$

subject to $\|\boldsymbol{x}\|^2 = 1$

can be written as…

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} ||\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2$$

suppressing constraint from notation for simplicity

due to orthonormality

$$||\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2 = ||\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2$$

just rotates contents and doesn't scale it so magnitude is unchanged

can be written as …

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} ||\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2$$

due to orthonormality

$$||\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2 = ||\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2$$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} ||\mathbf{\Sigma}\mathbf{V}^\top \boldsymbol{x}||^2$$

substitute $\boldsymbol{y} = \mathbf{V}^\top \boldsymbol{x}$

from orthonormality $||\mathbf{V}^\top \boldsymbol{x}||^2 = ||\boldsymbol{x}||^2$

$$\hat{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}} ||\mathbf{\Sigma}\boldsymbol{y}||^2$$

if diagonals are sorted in decreasing order:

$$\boldsymbol{y} = [0, 0, \cdots, 1]^\top$$

From substitution we know that:

$\boldsymbol{y} = \mathbf{V}^\top \boldsymbol{x}$ $\quad$ $\boldsymbol{x} = \mathbf{V}\boldsymbol{y}$

Therefore:

$\boldsymbol{y}$ is the last column of $\mathbf{V}$

Therefore in our code, we use SVD to compute the 3 matrices and return the last column of the V unitary matrix.

2.1.4 Measure error between triangulated and original 3D points

For this project, we choose Euclidean distance as our equation.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

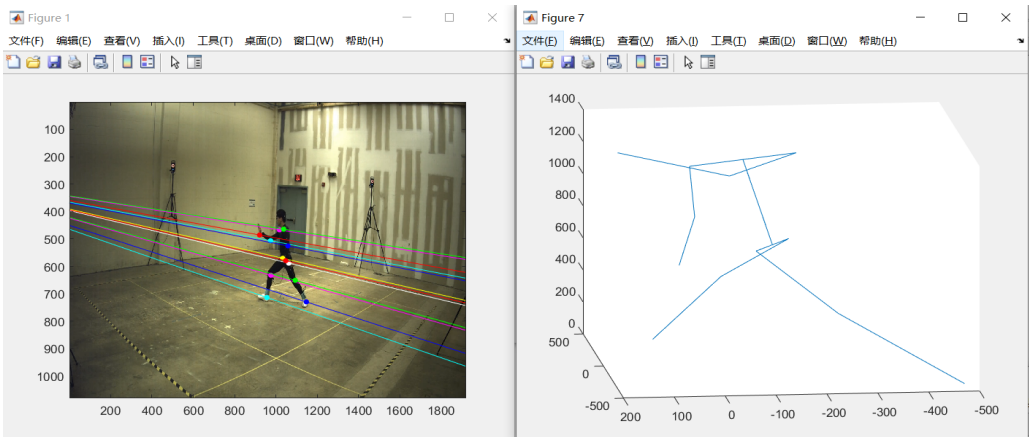
To measure the error between triangulated and original 3D points first we compute the Euclidean distance between reconstructed 3D points to come close to the original set of 3D points. Then we compared the original 3D matrix to the reconstructed 3D matrix and computed the mean, standard deviation, minimum, maximum, and medium.

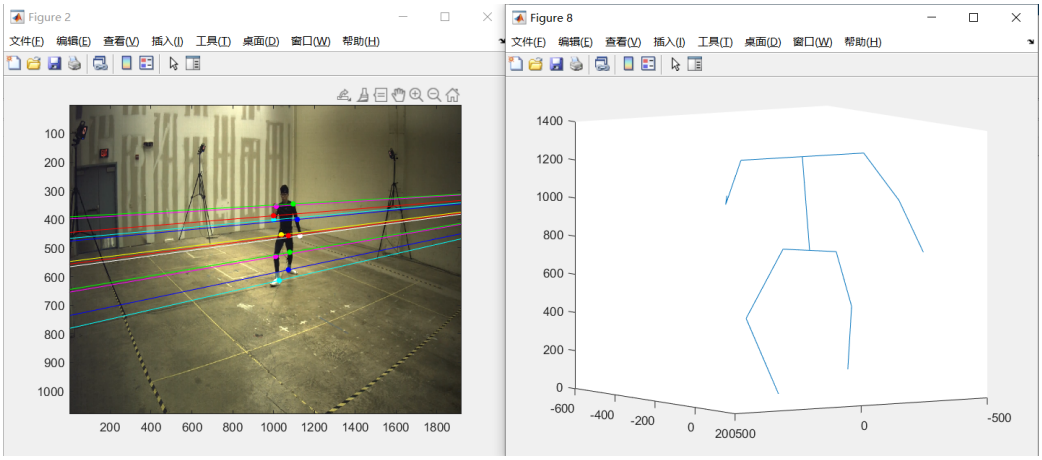
2.1.5 Computing epipolar lines between the two views

To compute the epipolar line between the two cameras, we decide to use the method that determines the epipoles in each view from the given camera calibration data for each projected mocap point in the image, then draw the line containing that point and the epipole for that image.

3.0 EXPERIMENTAL OBSERVATIONS

In this project, we are required to use frame 1000 to get the epipolar line and the skeleton. Therefore, we tried to use different frames to compare if we still get the result which we expect. The following two graphs are the result which we choose frame = 20000:





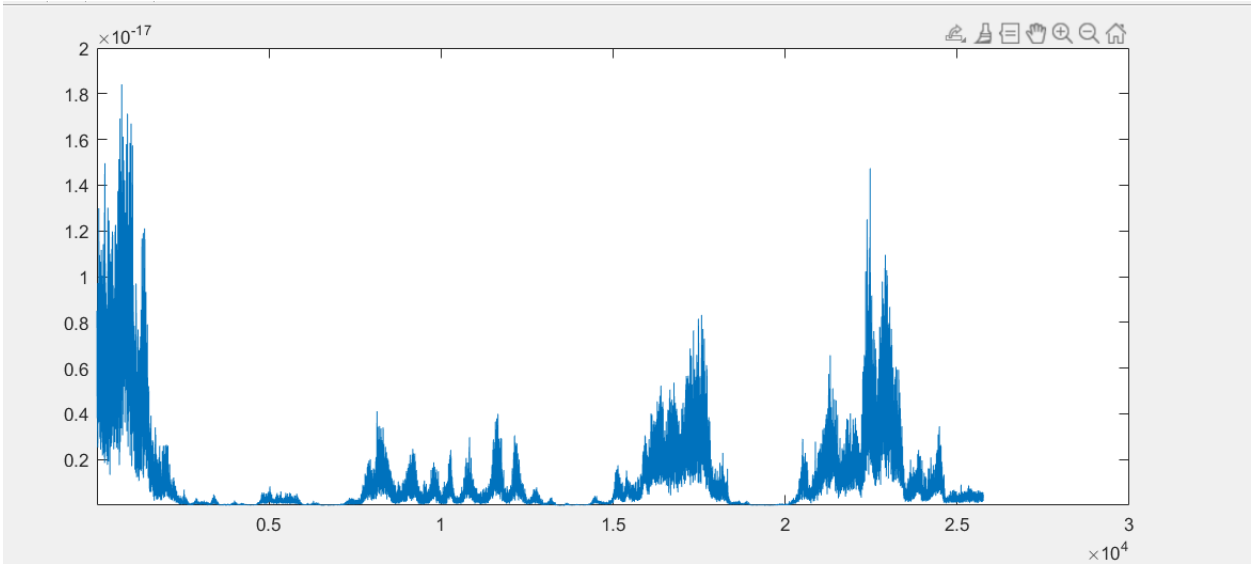
After comparing those input skeleton and output skeleton, we still get the result which we want and the error is still very close to 0. This project did give us a new perspective to understand how the stereo system in the camera projection actually works.

4.0 QUANTITATIVE RESULTS

4.1 Table of 12 joint error value

	Mean	Standard Deviation	Minimum	Median	Maximum Distance
Right shoulder	1.017327e-19	3.105695e-19	7.888609e-31	1.676624e-20	9.538681e-18
Right Elbow	1.101763e-19	3.118945e-19	1.097940e-25	1.524107e-20	6.473228e-18
Right Wrist	1.161862e-19	3.123229e-19	2.178834e-26	1.495297e-20	6.531115e-18
Left Shoulder	7.961800e-20	2.153479e-19	7.815294e-27	1.302832e-20	5.077533e-18
Left Elbow	8.028671e-20	2.293042e-19	8.535475e-27	1.050413e-20	4.829420e-18
Left Wrist	9.648185e-20	2.842600e-19	4.090244e-27	1.090551e-20	9.291231e-18
Right Hip	8.478055e-20	2.554924e-19	1.616849e-26	1.187546e-20	5.475463e-18
Right Knee	9.069420e-20	2.611611e-19	1.539856e-26	1.209223e-20	5.952429e-18
Right Ankle	9.796018e-20	2.931349e-19	1.090521e-26	1.145720e-20	8.334312e-18
Left Hip	7.595843e-20	2.264355e-19	1.797341e-26	9.806524e-21	5.605303e-18
Left Knee	7.087689e-20	2.171939e-19	1.230820e-27	7.456308e-21	4.400264e-18
Left Ankle	7.057043e-20	2.270063e-19	3.317002e-26	5.788178e-21	4.230645e-18
Overall Results	8.961020e-20	2.650311e-19	7.888609e-31	1.110152e-20	9.538681e-18

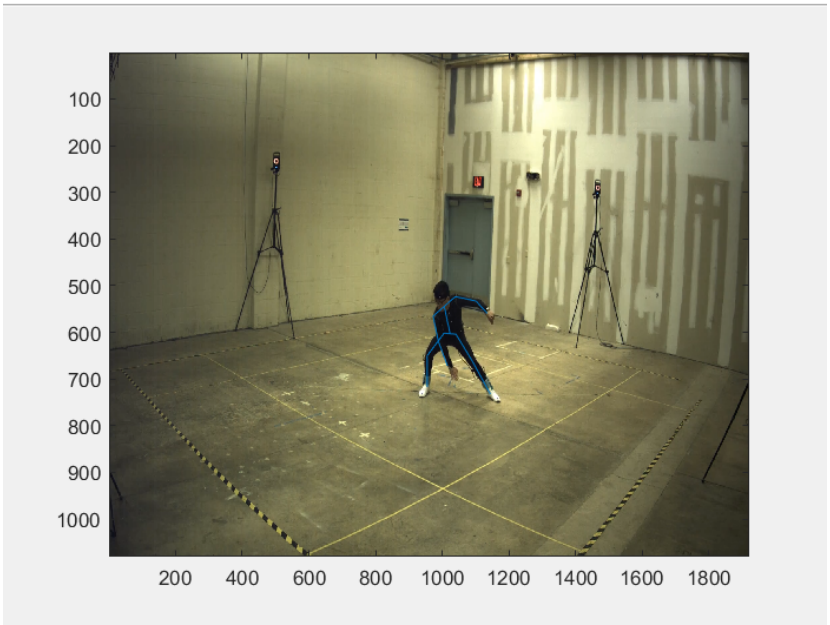
4.2 Total error across all frames of the sequence:



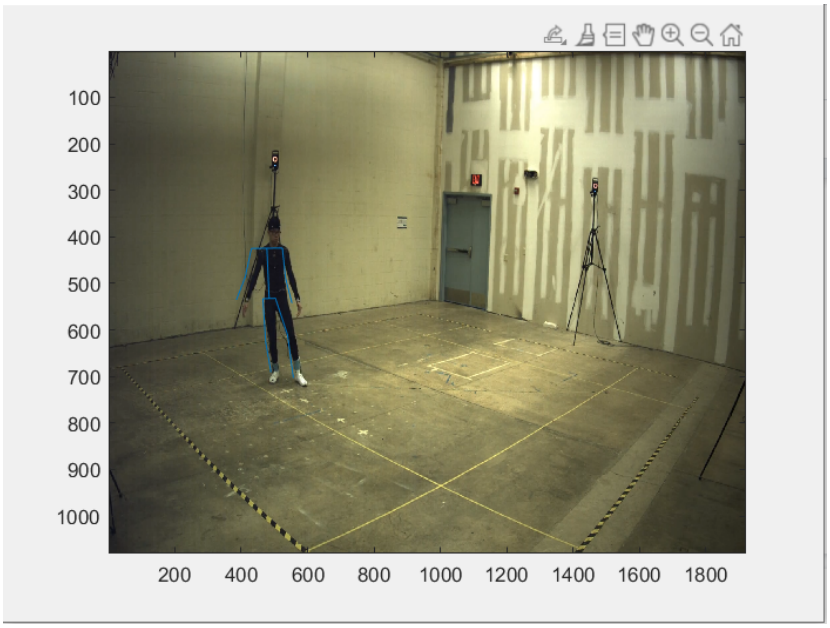
In this graph, we can see there are notable peaks and valleys in that, the whole graph seems like a wave. The extreme value appears during 0-0.25 and 2-2.5.

5.0 QUALITATIVE RESULTS

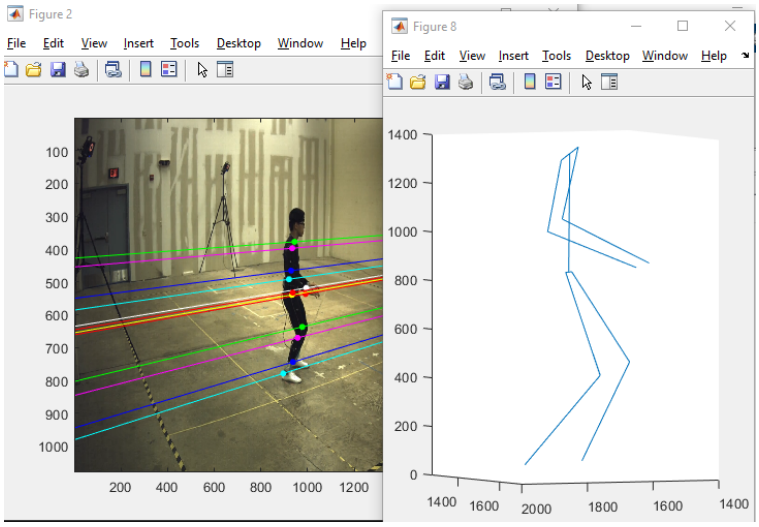
5.1 Minimum Error Frame :



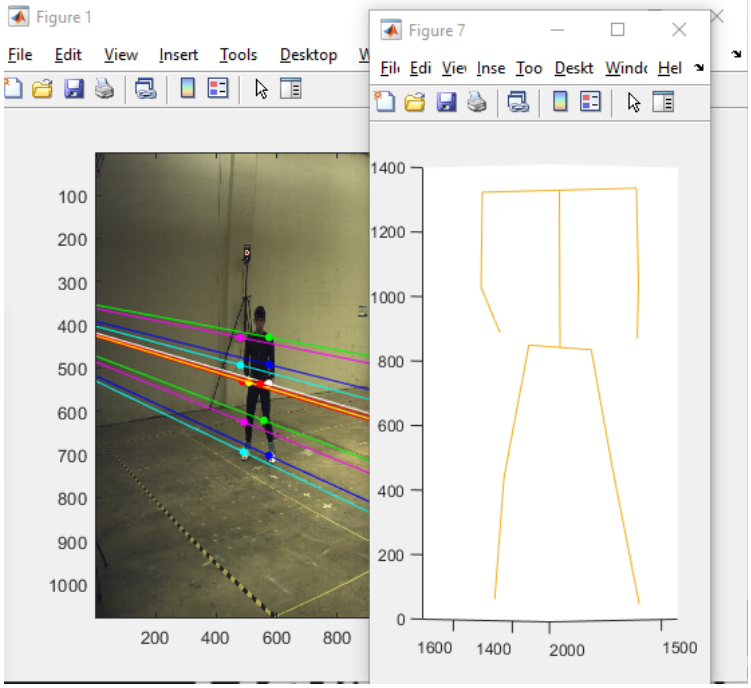
5.2 Maximum Error Frame:



5.3 Input Skeleton:



5.4 Output Skeleton:



6.0 ALGORITHM EFFICIENCY

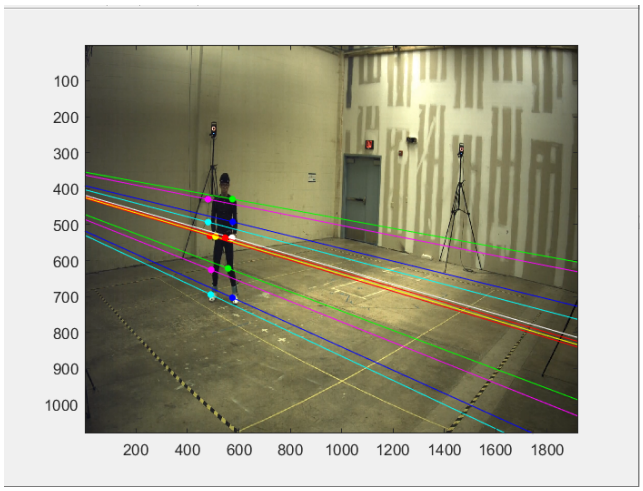
6.1 Profile Viewer:

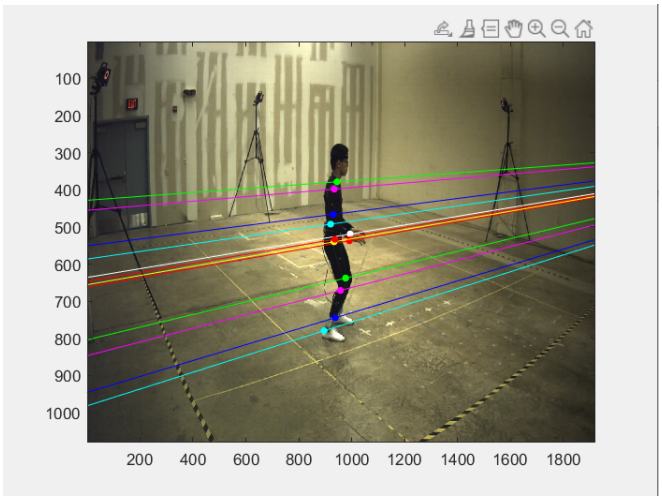
函数名称	调用次数	总时间(秒)	自用时间* (秒)	总时间图 (深色条带 = 自用时间)
main	1	17.373	4.003	
triangulate	25773	6.969	6.969	
cordTrans	51552	3.388	3.388	
IVideoReader>IVideoReader.cacheFrame	14	1.982	0.004	
VideoReader>VideoReader.cacheFrameTargetImpl	14	1.978	0.002	
VideoReader>VideoReader.cacheNextFrame	22	1.959	0.002	
InputStream>InputStream.read	39	1.957	0.016	
IVideoReader>IVideoReader.set.CurrentTime	6	1.950	0.002	
VideoReader>VideoReader.readFrameAtPosition	6	1.941	0.002	
VideoReader>VideoReader.readFrameAtTime	6	1.939	0.006	
Stream>Stream.wait	14	1.879	0.390	
epipolar	1	1.046	0.047	
VideoReader>VideoReader.VideoReader	2	0.360	0.000	
IVideoReader>IVideoReader.IVideoReader	2	0.359	0.001	
VideoReader>VideoReader.initReader	2	0.357	0.001	
VideoReader>VideoReader.VideoReader	2	0.340	0.005	
…ler.ToolbarController>@(e,d)obj.handleMouseMotion(e,d)	4	0.319	0.001	
…Controller>DesktopToolbarController.handleMouseMotion	4	0.318	0.003	
InputStream>@(obj)obj.getDataAvailable()>0	20280	0.312	0.054	
Stream>Stream.isDeviceDone	20283	0.288	0.288	

This is what we get from the Profile Viewer in Matlab, we can see our program's total running time is 17.37s, the function reconstruct3DFrom2D uses 6.9s seconds and the function project3DTo2D uses 3.38s and which is called most (51552 times).

7.0 TWO-VIEW EPIPOLAR LINES VISUALIZATION

We first calculate the pixel coordinate of a camera in the opposite camera's image.Then we use that pixel coordinate and the joint point from this image, to draw the view line of the opposite camera.





8.0 CONTRIBUTOR

8.1 Dawang Shen

- Code and report.

8.2 Zekun Peng

- Report Sections: entire Section 2 on outline and function approach
- Use lucidchart to create Flow Chart

8.3 Zhenghan Fang

- Coding & Report
- Editing and Revising