

CMPEN 454, Project 2, Fall 2018

Xi Li
Qi Chang
Eryk Maciej Heyssler
Brian Mata

Introduction

This project was intended to provide us with a hands on experience of the relationship between 2D (2-dimensional) image coordinates and 3D (3-dimensional) world coordinates using the ‘machinery’ of image transformations as discussed in class. Our task was to apply this machinery to a system that involved the projection of 3D coordinates, the conversion of 2D points into 3D viewing rays, and eventually the triangulation of the viewing rays to link the process into a “full circle” and receive the 3D coordinates with which we started. The small, individual tasks seemed simple, as they were mostly, if not purely, computational. The true difficulty in this project was tying the concepts together; putting them to work in an interconnected procedure in order to produce a single, seamless mechanism-- this the heart of the project.

Method

The core of this project rested upon the application of forward projection and backward projection between 3D world coordinates and 2D film coordinates. As described in class, the movement from 3D world coordinates to 2D film coordinates is what is known as forward projection, whereas the movement from 2D film coordinates to 3D world coordinates is called backward projection. Figure 1 demonstrates this relationship in the context of the individual steps taken in each concept:

Pixel location	Film plane to pixels	Perspective projection	World to camera	World point
$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$	$\sim \begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	$\begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: left; width: 45%;"> \swarrow Forward Projection </div> <div style="text-align: right; width: 45%;"> \nwarrow Backward Projection </div> </div>				

Figure 1. Forward and backward projection

Forward Projection

When applying forward projection, we have used the internal matrix and the external matrix to calculate 3D world point into 2D pixel location, which is the position on the matlab figure.

Instead of going through the laborious task of calculating 4 matrices, we used the internal matrix and external matrix that were given in the data, which led to saved time and a more streamlined process. The given data was in the form of a 3 by 4 matrix called Pmat, and a 3 by 3 matrix called Kmat. The process of understanding what each matrix represents is outlined in figure 2.

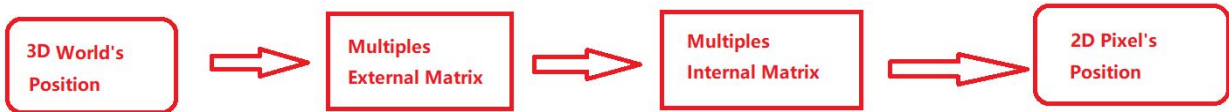


Figure 2. Control flow graph of forward projection

Conceptually, the external matrix represents the projection of points from world coordinate to camera coordinates. This is shown as follows.

$$\underbrace{\begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{M}_{\text{ext}}}$$

The size of this matrix should be 4 by 4 for the coordinate system. However, when the system is not in homogeneous coordinates, the matrix can be 3 by 4 $[R|\tau]$, which contains 3 by 3 rotation information and 3 by 1 translation information. Based on this information, we deduced that **the given matrix Pmat, should be the external matrix.**

Meanwhile, the internal matrix represents the projection of point form camera coordinate to film coordinate.

$$\underbrace{\begin{matrix} \text{Film plane} & & \text{Perspective} \\ \text{to pixels} & & \text{projection} \end{matrix} \begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{M}_{\text{int}}}$$

The size of this matrix should be 3 by 4 for the coordinate system. However, when the system is not in homogeneous coordinates, the matrix can be 3 by 3, which we regardless of the last column. **According to the data, kmat should be the internal matrix.**

When we were trying to decide how the matrices ‘operated’ we checked their values and found these results:

$\mathbf{Pmat2} =$ $1.0\text{e}+03 \cdot$ $\begin{bmatrix} -0.0008 & -0.0006 & 0.0000 & 0.1377 \\ -0.0001 & 0.0001 & -0.0010 & 0.8055 \\ 0.0006 & -0.0008 & -0.0002 & 7.3365 \end{bmatrix}$	$\mathbf{Kmat2} =$ $1.0\text{e}+03 \cdot$ $\begin{bmatrix} 1.5578 & 0 & 0.9760 \\ 0 & 1.5578 & 0.5628 \\ 0 & 0 & 0.0010 \end{bmatrix}$
---	--

Figure 3. Pmat and Kmat of camre vue2

We observed that Pmat2 is a 3 by 4 which would contain the rotation information and shift information. We also saw that the matrix Kmat2 has same structure as:

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

and it should be zero on the M12 and M21 after two matrices multiplied.

Finally, we judged them in few points and decided **Pmat is the external matrix** and **kmat is the internal matrix**.

Elementary Results

By discovering what the given matrices described, we could simply calculate forward projection by doing the calculation: $\mathbf{M}_{\text{internal}} * \mathbf{M}_{\text{external}} * \mathbf{P}_w$.

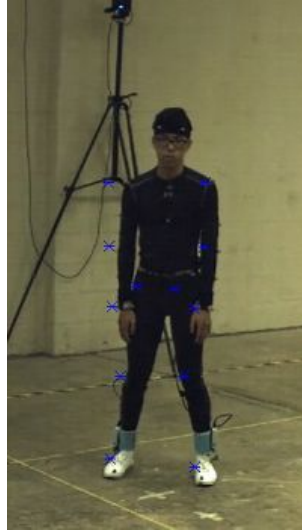


Figure 4. Result of forward projection without nonlinear distortion

However, this initial result was not perfect, since there exists lens distortions that happen during the projection from film coordinate to pixel coordinate. After some research, we found that there are two typical types of distortion: **radial distortion** and **decentering distortion**.

Radial distortion is most visible when taking pictures of vertical structures having straight lines which then appear curved. This kind of distortion appears most visibly when the widest angle (shortest focal length) is selected either with a fixed or a zoom lens. It may be adequately corrected by applying a simple polynomial transformation as discovered by the 19th century mathematician and astronomer Philipp Ludwig von Seidel (1821-1896), in a written published by him in 1856 which requires three constants affecting the image content as a function of the distance from the center and symmetrical about it, hence the name radial distortion. Some typical radial distortion is shown as following:

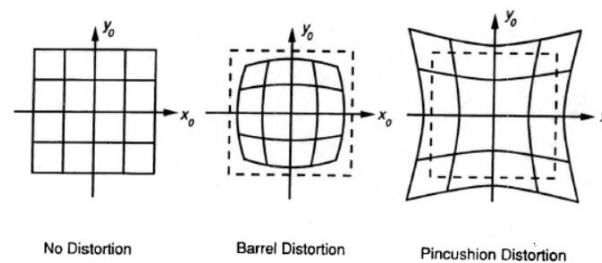


Figure 5. Distortion types

Decentering distortion is frequently caused by improper lens installation, and it results in point displacement between radial and tangential image. However, the decentering distortion is generally only approximately 1/7-1/8 of the radial distortion so some simplified analysis models will ignore it.

In our coding, we assumed that we only have radial distortion and our secondary goal was trying to corrected it.

This kind of distortion only happened in internal matrix. Hence, we used the internal matrix in the data and extracted affine matrix and projection matrix. After that, we have added correlation before we used affine matrix to get the final answer.

$$\begin{array}{ccccc}
 \text{Pixel} & \text{Film plane} & \text{Perspective} & \text{World to camera} & \text{World} \\
 \text{location} & \text{to pixels} & \text{projection} & & \text{point} \\
 \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} & \sim \begin{bmatrix} \pm 1/s_x & 0 & o_x \\ 0 & \pm 1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}
 \end{array}$$

\uparrow
Distortion

Figure 6. Nonlinear distortion in forward projection

In details, we first constructed a projective matrix with the given focal length.

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

To get the affine matrix, we divided the internal matrix Kmat by the projective matrix we just constructed. Then, we pass the the result of multiplying the 3D world position by external matrix and projective matrix to function performRadialDistortion to get the corrected film coordinates. Then we multiply the film coordinates by affine matrix and plot the resulting image points on the frame, as shown in figure 8.

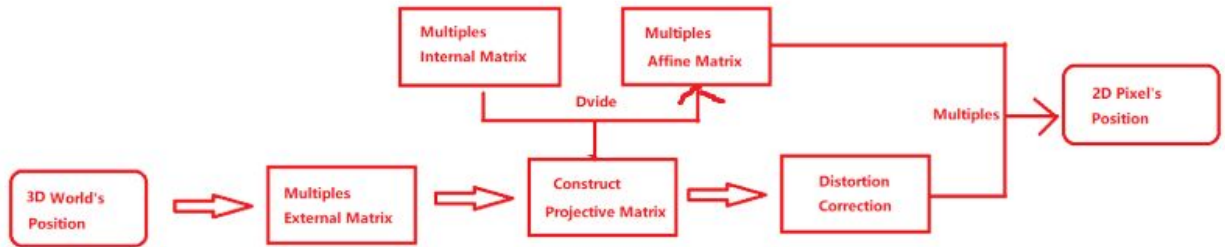


Figure 7. Control flow graph of nonlinear distortion

Secondary Results



Figure 8. Result of forward projection with nonlinear distortion

In some instances, the corrected result was the same as the original result, while other times, the corrected results were in fact more accurate. We noticed that the effect of the radial distortion was strongest around the edges of the image, and weakest near the center of the image where there seemed to be almost no distortion.



Figure 9. Comparison of results with (in red) and without (in blue) nonlinear distortion

Backwards projection

After having applied forward projection to our images, we are now left with two images with world points projected onto them. We want to use this to “reconstruct” the 3D data we were originally given. In order to achieve this, we had triangulate the image coordinates back into

world coordinates. In an ideal scenario, we could simply find the intersection of the two viewing rays, however, if noise presents, the viewing rays would not intersect. To account for that, we implemented the following algorithm to correct for noise in the image.

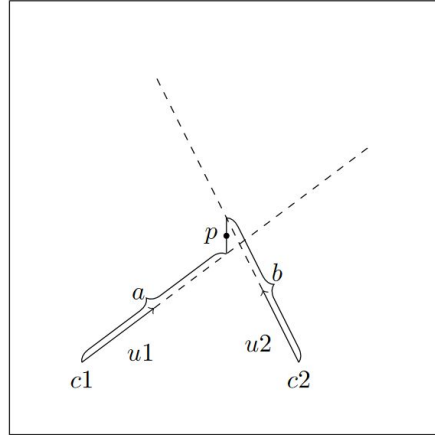


Figure 10. Triangulation with noise

From the lecture, we have learned that

$$P_w = -R^T C + \lambda R^T K^{-1} P_M$$

The first term represents the location of the camera, and the second represents vector pointing out from camera into the world position. Therefore, we can get u , the direction of viewing ray, by normalizing $R^T K^{-1} P_M$. Now we use u_1 and u_2 to denote the viewing rays of camera 1 and camera 2. Since u_1 and u_2 may not intersect, we choose the midpoint of the segment perpendicular to both u_1 and u_2 as the “intersection”. Based on figure 10, we can get the equation:

$$au_1 + du_3 - bu_2 = C_2 - C_1$$

Where u_3 is the cross product of u_1 and u_2 , a , b , d are length of vector u_1 , u_2 and u_3 , C_1 and C_2 are camera positions. Then we can get the a , d and b by rewriting the equation into:

$$[a \ b \ d]^T = [u_1 \ -u_2 \ u_3]^{-1} * [C_2 - C_1]$$

At last, “intersection” point p , or the world point, is evaluated by $\frac{1}{2} * [(C_1 + a*u_1) + (C_2 + b*u_2)]$

The control flow graph of triangulation is shown in figure 11, and the algorithm is implemented in our own function called triangulateDLT.

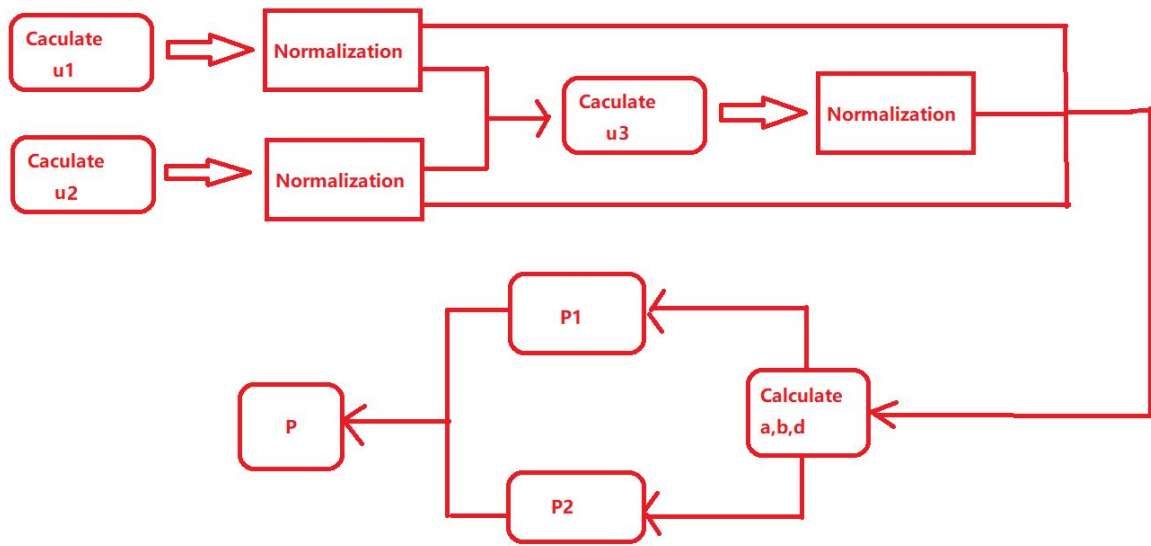


Figure 11. Control flow graph of triangulation with noise

And we also do correction in backward project to see the difference between the reconstructed 3D points. The process is similar to what we do in nonlinear distortion. We pass the film coordinates converted from the distorted image points to the function `performRadialCorrection`. Then we convert the corrected film coordinates back to image points. By triangulation, we reconstruct the 3D world points. The difference between 3D points with and without backward correction is discussed later. The comparison of them are discussed in the section of final results.

Final Results

In our final result, we just want to show the interesting result mentioned in project assignment, which is adding nonlinear distortion in forward but no correction in backward projection. Also, we recorded a movie with this result with the ground truth. In this section, we have provided the quantitative, qualitative analysis and the movie recording. The result after correction were used to show the comparison in qualitative analysis.

1. Quantitative:

Table 1. Errors of reconstructing 3D points with nonlinear distortion in forward projection only

	Right shoul der	Right elbow	Right wrist	Left shoul der	Left elbo w	Left wrist	Righ t hip	Righ t knee	Right ankle	Left hip	Left knee	Left ankl e
Mean	22.11 19	21.14 31	23.15 50	19.67 37	17.4 857	20.25 71	15.7 618	17.7 703	21.15 79	14.4 477	14.0 512	16.7 133
Std	24.43 01	26.55 65	29.14 07	19.12 08	20.6 708	25.33 47	20.3 528	21.5 781	23.32 04	18.1 058	17.5 627	18.7 834
Min	1.750 2	0.388 8	0.221 2	2.871 3	0.95 70	0.389 0	0.26 71	0.05 79	0.324 8	0.26 64	0.02 00	0.43 66
Max	123.8 194	131.7 660	167.5 844	82.38 65	99.8 206	139.9 378	97.4 451	86.5 108	104.0 500	79.0 674	77.9 158	77.4 599
Median	13.27 28	10.16 81	10.36 17	11.54 72	8.38 04	8.697 0	7.54 76	10.7 229	13.27 69	6.63 63	5.78 86	7.98 34

This is the set of data which uses radial correction on the data during the forward projection (3D \rightarrow 2D), but not during the backwards projection (2D \rightarrow 3D). Here we can see that even aligning the motion capture data to the film plane is not enough for accurate reconstruction of the motion capture data as the L^2 error is very large. An interesting thing to note is that the median is less than the mean which implies that the data might be skewed right. It's possible that performing the reconstruction is not good most of the time but that there are a few large errors skewing the data. The variance is also relatively large reinforcing the idea that the data might have many good points as well as many outliers.

Table 2. Errors of reconstructing 3D points with nonlinear distortion in forward projection and correction in backward projection

	Right shoul der	Right elbo w	Right wrist	Left shoul der	Left elbo w	Left wrist	Right hip	Right knee	Right ankle	Left hip	Left knee	Left ankle
Me an	1.213 6E-1 2	1.210 1E-1 2	1.209 8E-1 2	1.227 3E-1 2	1.231 6E-1 2	1.234 7E-1 2	1.226 0E-1 2	1.236 4E-1 2	1.259 2E-1 2	1.234 9E-1 2	1.246 3E-1 2	1.283 5E-1 2
Std	5.651 9E-1 3	5.623 9E-1 3	5.707 9E-1 3	5.738 3E-1 3	5.822 7E-1 3	5.822 0E-1 3	5.670 7E-1 3	5.649 8E-1 3	5.635 3E-1 3	5.719 6E-1 3	5.771 5E-1 3	5.872 2E-1 3
Mi n	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	0.000 0E+0 0	2.131 6E-1 4	0.000 0E+0 0	0.000 0E+0 0	7.105 4E-1 5
Ma x	5.007 4E-1 2	4.616 6E-1 2	5.089 3E-1 2	4.571 6E-1 2	5.220 9E-1 2	5.504 1E-1 2	4.459 3E-1 2	4.828 7E-1 2	4.801 6E-1 2	4.661 5E-1 2	5.115 9E-1 2	5.601 5E-1 2
Me dia n	1.153 0E-1 2	1.142 5E-1 2	1.136 9E-1 2	1.159 4E-1 2	1.159 4E-1 2	1.159 4E-1 2	1.155 2E-1 2	1.159 4E-1 2	1.182 9E-1 2	1.159 4E-1 2	1.164 9E-1 2	1.200 8E-1 2

With this data, we corrected for radial distortion both ways. As we can see, L^2 error is now on the magnitude of 10^{-12} which is much better than before when we had not corrected for radial distortion in the backwards projection. As we can see, the median is still less than the mean which again implies a skewed data with largely good points and a few bad outliers. Even still, we managed to bring even the scale of the error way down and this is a very good reconstruction.

Table 3. Errors of all joint pairs (independent of locations)

Mean	Std	Min	Max	Median
215.3539	242.9873	6.7489	1.0595e+03	111.4107

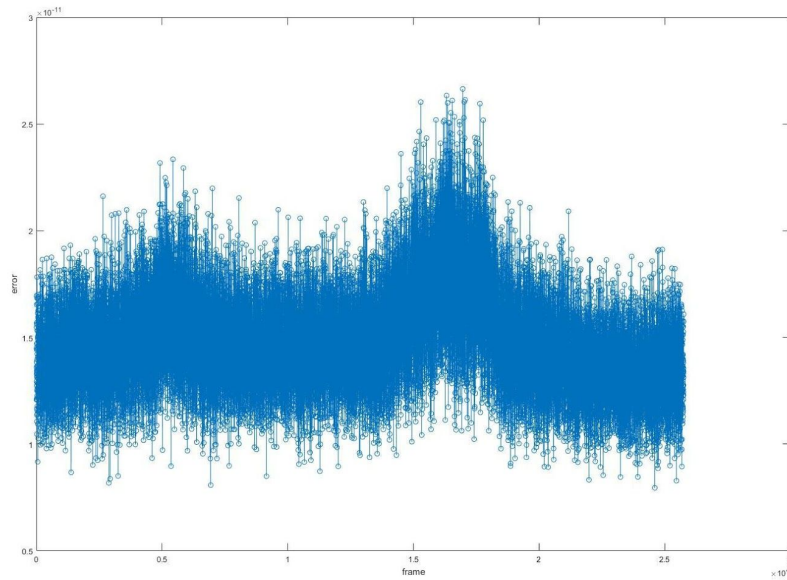


Figure 12. Error per frame of reconstructing 3D points with nonlinear distortion in forward projection and correction in backward projection

Here we can see that error is fairly consistent throughout with a noticeable spike sometime towards the center. This graph also illustrates the idea that data is fairly spread and that there are many outliers.

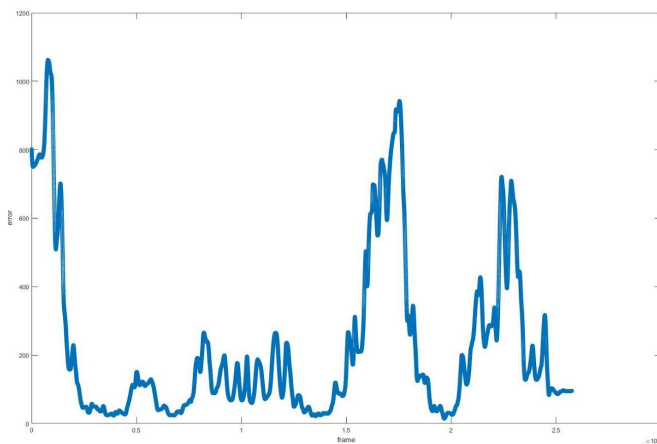


Figure 13. Error per frame of reconstructing 3D points with nonlinear distortion in forward projection only

Here we can see that the error is way down from the uncorrected dataset. However, there is still a large jump in errors somewhere around the middle. Perhaps there is something analogous to a *blindspot* in the camera system. Somewhere where small discrepancies in the location of the dancer might affect the location of his projection disproportionately. And while the dancer stands in this blindspot, the data is prone to large errors.

2. Qualitative:

Here we select frame No.1000 to do the visualization. Aligning the skeleton to the dancer reveals the how accurate the projection system is at reconstruction.

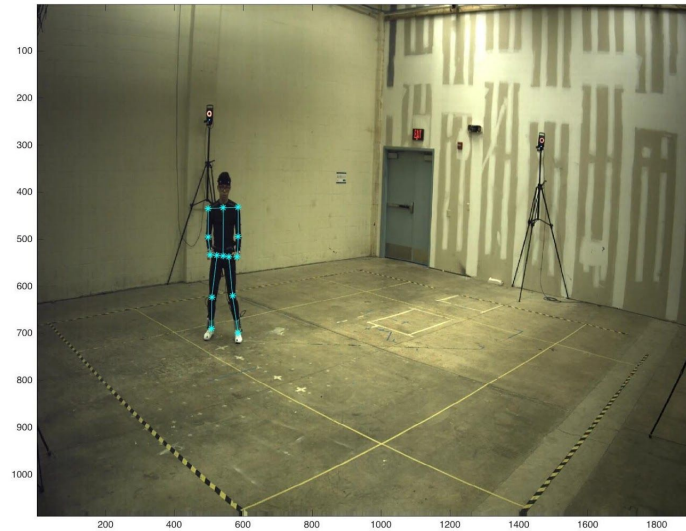


Figure 14. 2D projection in camera vue2 (with nonlinear distortion)

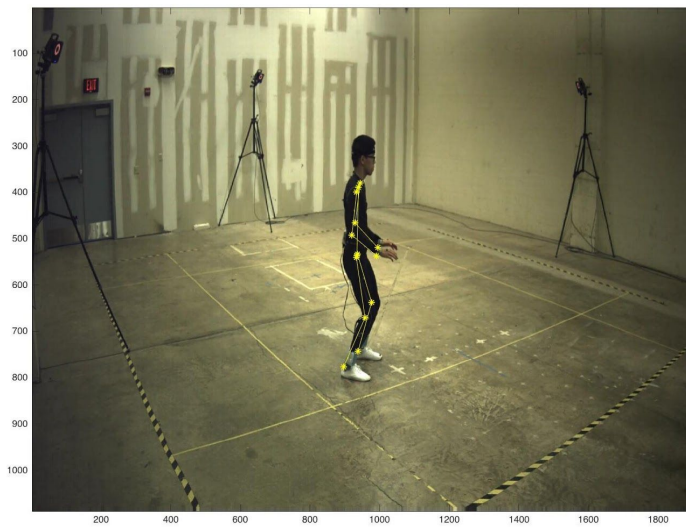


Figure 15. 2D projection in camera vue4 (with nonlinear distortion)

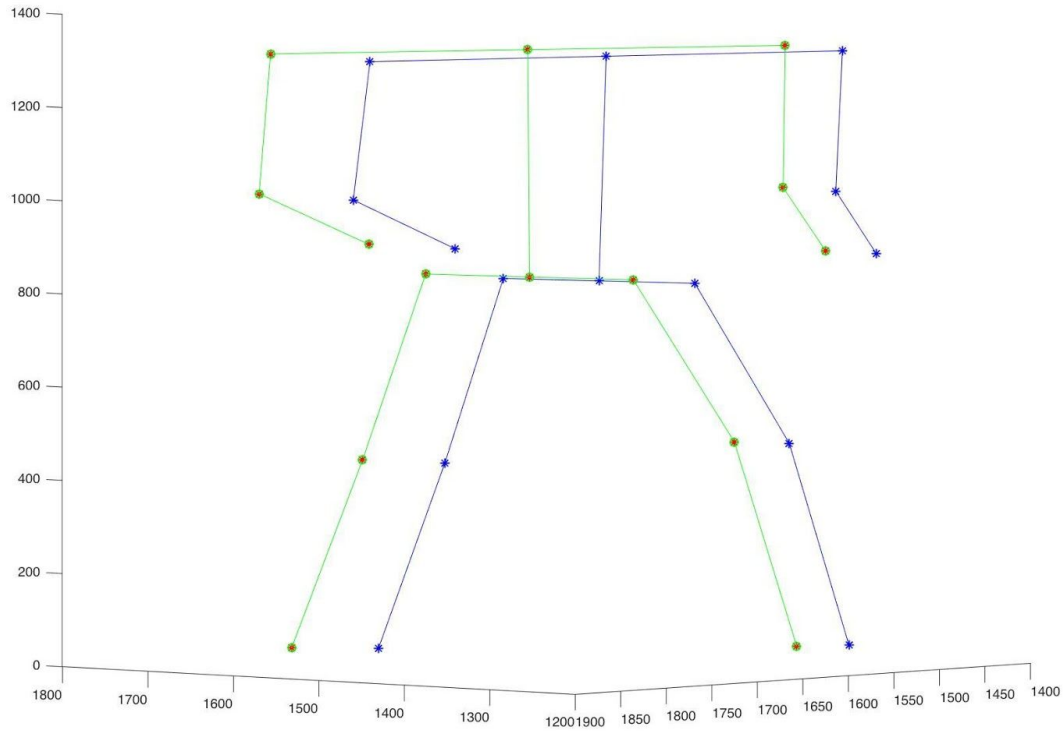


Figure 16. Reconstructed 3D skeleton (input 3D world points in green, 3D reconstructed points with nonlinear distortion in forward projection in blue, 3D reconstructed points with nonlinear distortion in forward projection and correction in backward projection in red)

In figure 16, we can see the problem that if we used nonlinear distortion in forward projection only, the reconstructed 3D points will result in a shift (blue) from the input data (green). To avoid the error caused by the distortion, we add correction in backward projection (red).

Table 4. Error between world points and 3D reconstructed points with nonlinear distortion in forward projection and correction in backward projection

Right shoulder	Right elbow	Right wrist	Left shoulder	Left elbow	Left wrist	Right hip	Right knee	Right ankle	Left hip	Left knee	Left ankle
7.876 5E-13	8.507 5E-13	2.542 1E-13	9.646 6E-13	1.042 0E-12	8.879 2E-13	8.507 5E-13	8.824 5E-13	1.053 9E-12	1.271 1E-12	1.345 2E-12	7.563 9E-13

Max: 1.3452e-12

Min: 2.5421e-13

In the same way as before, the corrected data is very accurate. The L^2 error of the dataset is generally on the scale of 10^{-13} - 10^{-12} . Strangely though, locations where there was the most error, left knee, left hip, and right ankle, seem to come from the left side and further down on the dancer. Accordingly, the right wrist is also surrounded by low error regions. In any case, the errors are very small and likely contributed to by rounding errors and floating point error.

Table 5. Error between world points and 3D reconstructed points with nonlinear distortion in forward projection only

Right shoulder	Right elbow	Right wrist	Left shoulder	Left elbow	Left wrist	Right hip	Right knee	Right ankle	Left hip	Left knee	Left ankle
93.8048	88.1517	79.5040	52.1781	47.3057	44.8088	71.3911	76.1080	81.2352	55.4625	49.1600	47.3145

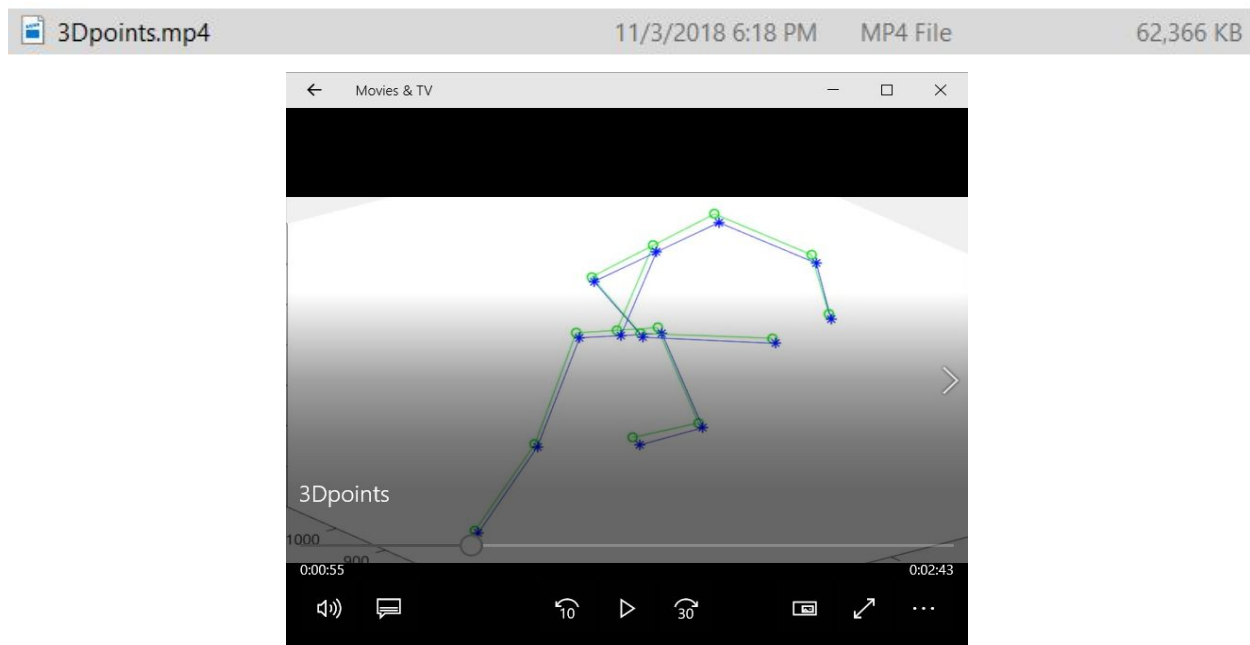
Max: 93.8048

Min: 44.8088

The data here results from uncorrected linear distortion. It explicitly shows the L^2 error between world points and 3D constructed points, and, as observed previously, this uncorrected data tends to be rather erroneous. At first glance, the difference between the Max and the Min (~ 94 to ~ 45), approximately 50 in magnitude, seems substantial. However, in the grand scheme of things, this magnitude is insignificant because the errors are of the same scale.

3. Movie Recording







We make a movie comparing original and reconstructed skeletons. The movie is named as “3Dpoints.mp4” and runs for 2 minutes and 43 seconds.



Algorithm Efficiency

The MatLab® profiler is a built in tool that measures where code takes up the most time upon running it-- it is essentially a tool that can be used to improve efficiency. As can be observed from the figure below, the bulk of the time spent running the code (60.8%) was dedicated to the line 168, corresponding to “*waitbar (mocapFnum/N)*”. This of course is representative of frame computation for each frame, so it is not surprising that this takes up the majority of the time, as it constitutes the actual parsing of the mocap dataset. The second most significant resource users were the calls to our triangulation function, which took up a total of 24.2% of total runtime. We believe this would be very difficult to improve on (without an incredibly deep understanding of geometry), as this is a computational result. The total time spent is approximately 1 minute, which we estimate to be decent. However, given more substantial data, our implementation could possibly be much slower.

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
168	<code>waitbar (mocapFnum/N) ;</code>	26214	38.077 s	60.8%	
228	<code>[predicted, distances] = trian...</code>	25758	8.083 s	12.9%	
235	<code>[predicted, distances] = trian...</code>	25758	7.077 s	11.3%	
181	<code>ImagePoints2 = bsxfun(@rdivide...</code>	25758	0.812 s	1.3%	
70	<code>vue2video.CurrentTime = (mocap...</code>	1	0.583 s	0.9%	
All other lines			7.977 s	12.7%	
Totals			62.610 s	100%	

Concluding Remarks

It was very interesting to see the final 3D reconstruction juxtaposed against the original motion capture data. We were able to see the differences between our reconstruction and the “true” data, which help put into perspective the weaknesses of our methods. We can imagine that in order to get really accurate results, it would help to have more than two cameras to weed out some of the noise that you can’t really see with just two images.

What differed in this project in comparison to the first project is that we were given extra information in the form of the two matrices, Pmat and Kmat. With this addition, our methodology was slightly more constrained, as we had to first “reverse-engineer” the process by figuring out what the matrices represented. From there, we were able to use our knowledge of camera projection to build the project around the given information.

One of the first challenges we faced was the issue of radial distortion. Suddenly, we were introducing a non-linear issue to our linear system. Figuring out exactly where to apply the given correction functions was a bit challenging. Through creative thinking (and admittedly, a bit of guesswork), we were able to fix this issue by extracting the affine matrix and projection matrix from the internal matrix, and then manipulating that to yield our results. The distortion correction that we applied greatly improved the results we were getting. From there, we had to reconstruct the given data back into 3D coordinates. This required a bit of mathematical prowess, though we were able to push through.

We had a total of three meetings to discuss the project, during which everyone was present. During the first meeting, as a group, we decided what the P_{mat} and K_{mat} matrices represented, and discussed a rough outline of how to approach the problem. By the second meeting, we all had rough drafts of code that we assimilated to create one cohesive file that was able to function as a whole. The second meeting, our discussion revolved around correcting for distortion, as well as a mathematical structure for triangulation. During the last meeting, everyone contributed what they had in terms of code, and we wrote the report together.