CSCE 643 Multi View Geometry CV Final Project Proposal

Yukun Zeng
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77840
Email: yzeng@tamu.edu

I. Introduction of 3D Reconstruction

A. Definition

The essence of an image or a photo is a projection from 3D scene onto a 2D plane[1]. Naturally we can think of that the depths of things are lost in this process as the final image we get from the projection is in 2D, thereby a lot of information is lost during the projection. 3D reconstruction, the reverse process of obtaining 2D images from 3D scenes, is the creation of three-dimensional models from a set of images.

B. Motivation

The applications of 3D reconstruction are penetrated in our daily life. As 2-D imaging has problems of anatomy overlapping with each other and don't disclose the abnormalities, 3-D imaging can be used for both diagnostic and therapeutic purposes. The 3-D models we obtain from reconstruction can be used for operation planning, morphometric studies and has more reliability in orthopedics. Another recent tech trend is the development of 3-D printing, which also requires the model reconstruction to provide input to printers.

II. 3D Reconstruction from Uncalibrated Images

As we mentioned above, due to the lost of information, from a single 2D image, no depth can be computed without additional information. However, this problem becomes feasible when we have multiple images (hopefully focusing on certain region). The classic process is composed of two major steps:

- Match features across multiple images.
- Compute depth information using standard triangulation based on the above feature correspondence.

However, traditional approach like this requires careful calibration, which is a major drawback as it's error-sensitive. There are also other methods proposed but they either requires more than commonly known information or lead to certain drawbacks in results. Therefore, a relative 3D reconstruction approach using uncalibrated images is proposed in [2]. The major advantage of this new approach over other previous ones are two fold:

- It introduces the uncertain 3D information only at the latest stage of the perception process, and only if it is needed.
- It avoids the unstable calibration process, and therefore relies only on the accuracy of the measures in the image during the processing.

The basic idea of our approach for implementation is that it incorporate calibration and reconstruction in the same optimization process by an implementation of parameters estimation theory using Levenberg-Marquardt algorithm. Different from other earlier methods, our reference approach is formulated globally as a least squares estimation method that does not need to first estimate the epipolar geometry and it makes full use of redundancies in multiple images. Throughout this project, we will propose an implementation of the relative 3D reconstruction approach and evaluate the method using reasonable experiments to show its performance.

III. Methodologies

From a high level, the major approaches of this paper can be summarized as follows:

- A relative 3D reconstruction method that incorporates both calibration and reconstruction in a single optimization process. The method enables us to reconstruction 3D scenes even in noise pictures by leveraging the redundancies among multi images. The math behind the reconstruction approach is to use basic camera projection model for establishing least squares estimation based relations among image points, points in the world frame and projection matrices of multi cameras. Then we can leverage nonlinear solvers like Levenberg-Marquardt to solve the least squares estimation problem using the coordinates of multiple points in different images.
- An approach for determining the coplanarity among different points selected from images after we know the epipolar constraint, which can be used when finding reference points and five-point basis.

In the following sections, we will do a detailed reconstruction of the math part for those methods. Same as the paper, we assume a perfect pinhole camera model and noises in image measures are uncorrelated, Gaussian and centered. Though unnecessary, we assume that all points selected for 3D reconstruction exist in all the images we used for brevity of explanation.

A. 3D Reconstruction with Self-Calibration

This section presents the basic math foundations for 3D reconstruction as well as the approach of reconstructing 3D scenes with self-calibration. For the 3D reconstruction problem, we consider the scenario that we have m views of a scene (m photos that share overlappings of the same scene). Among those images, n points have been matched, which means we have $n \times m$ image points in total and as we assumed, we can find occurences of all those points in different pictures. Suppose $\mathbf{M}_i$, represented by $(\frac{x_i}{t_i}, \frac{y_i}{t_i}, \frac{z_i}{t_i})$ in nonhomogeneous form, is the original points set in 3D world frame corresponding the the n points we matched for reconstruction. The ith points projected from $\mathbf{M}_i$ to the camera plane of jth image is $m_{ij} = (u_{ij}, v_{ij})$. $\mathbf{P}_j$ is the 3×4 projection matrix for camera used for jth image.

1) Camera Projection Geometry: With basic knowledge in projection geometry, for homogeneous coordinates, apparently we have the following mathematic equation:

$$\rho_{ij} m_{ij} = \mathbf{P}_j \mathbf{M}_i \tag{1}$$

where ρ_{ij} is an unknown scaling factor that is different for each image point, $i \in [1, \ldots, n]$, $j \in [1, \ldots, m]$

Suppose that $\mathbf{P}_j$ can be expanded as a set of p_{ij}^j, which represents the element on ith row and jth column of $\mathbf{P}_j$, we can rewrite Eq. 1 as follows:

$$u_{ij} = \frac{p_{11}^{(j)} x_i + p_{12}^{(j)} y_i + p_{13}^{(j)} z_i + p_{14}^{(j)} t_i}{p_{31}^{(j)} x_i + p_{32}^{(j)} y_i + p_{33}^{(j)} z_i + p_{34}^{(j)} t_i}$$

$$v_{ij} = \frac{p_{21}^{(j)} x_i + p_{22}^{(j)} y_i + p_{23}^{(j)} z_i + p_{24}^{(j)} t_i}{p_{31}^{(j)} x_i + p_{32}^{(j)} y_i + p_{33}^{(j)} z_i + p_{34}^{(j)} t_i}$$

Fig. 1. Basic Equations from Camera Projection

Both of the above equations are simply a reflection of colinearity between spatial points and their projections on the image plane. One thing to highlight in implementation of this paper is that we assume no knowledge on coordinates of spatial points as well as the projection matrix of all cameras, the only thing known to us is the coordinates of points we selected in each image. Based on the above assumptions we made, we will have a linear equations set of that contains $2 \times n \times m$ equations, $11 \times m$ unknowns in projection matrices (instead of $12 \times m$ as we can define $\mathbf{P}$ up to a scaling factor). Additionally, there are $3 \times n$ unknowns for spatial points $\mathbf{M}_i$. If we have enough images (m) and point pairs (n), a redundant set of equations can be obtained easily.

Another thing worth mentioning here is that the solution for the above equations set should not be unique. Suppose a solution to the above equation system is $\mathbf{P}_j$ and $\mathbf{M}_i$, we can easily find a 4×4 invertible spatial collineation matrix $\mathbf{A}$, so that:

$$\rho_{ij} \mathbf{m}_{ij} = (\mathbf{P}_j \mathbf{A}^{-1})(\mathbf{A} \mathbf{M}_i), i \in [1, \ldots, n], j \in [1, \ldots, m] \tag{2}$$

Therefore, the solution for our basic equation system can only be determined up to a collineation.

B. Choosing Reference Points

Due to the fact that we can only determine the solution of the above system up to collineation, we can arbitrarily choose a basis (i.e., a coordinate system) in 3D space. According the definition of basis, 5 algebraically free points can form a basis, i.e., we need 5 points in which any 4 of them are not coplanar. We term the 5 basis points as reference points in rest of this paper and we introduce methods developed in the original paper for automatic selection of reference points in this section.

1) Coplanarity Test: The method for establishing epipolar geometry (the computation of fundamental matrix) has already been proposed in [3]. After that, we can further determine the coplanarity among any four points only through operations in image planes by leveraging properties embodied in the epipolar geometry. Moreover, the reference points we talked above can be easily and even automatically determined from the image once the coplanarity test function is developed.

The computation of fundamental matrix can be done through non-linear optimization method in [4] and we ignore it here for brevity, readers are referred to our previous project report for more details on it. Also, we would like to mention that the computation of fundamental matrix is totally unnecessary in our implemented 3D reconstruction approach, it's only used for better determination of reference points.

Here we assume that the fundamental matrix between two images are well calculated through some method, thus we have a 3×3 fundamental matrix of $\mathbf{F}$ that entains all information needed for epipolar geometry. With that, say we have two images and $\mathbf{F}$ is the fundamental matrix, $\mathbf{m} = (x, y, t)^T$ is a point in image 1, the corresponding epipolar line l' of it in image 2 shall satisfy the constraint $l' = (a', b', c')^T = \mathbf{F}\mathbf{m}$.

Now, consider III-B1 and III-B1, two cases for relationships between two 3D points: Coplanar and Non-Coplanar. In those two figures, we have the projection of 4 points A, B, C, D in two images, referred respectively as a, b, c, d and a', b', c', d'. Dashed lines in two images represent the epipolar lines going through some of those points. According to epipolar geometry, the epipolar line that goes through C must also pass C', vice versa.

In the case of Fig. III-B1, where the four points A, B, C, D are coplanar. The diagonals will surely intersect in a 3D point M that is projected respectively in two images as m and m'. Apparently we know that m and m' should also satisfy the epipolar constaint, i.e., the epipolar line that goes through m in left image must also pass m'.

However, in the case of Fig. III-B1, where A, B, C, D are not coplanar. There is no intersection between two diagonals

since they are not in the same plane, actually in this case, the point m becomes the projection of two 3D points in the space, say these two points are M_1 on line AC and N_1 on line BD. In the other image, similarly we have m' as the projection of both M_2 and N_2.

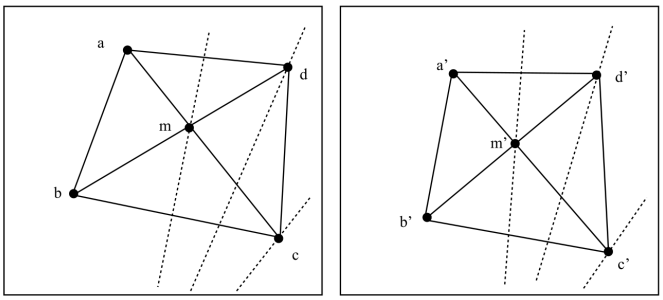


Fig. 2. Properties of Coplanar Diagonal Intersection across Different Images

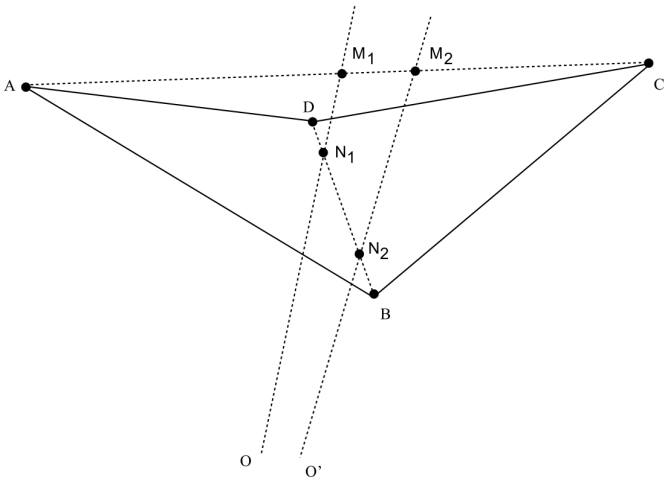


Fig. 3. Non-Coplanar Points and Epipolar Lines

Suppose O, O' are respectively central points of two images. Obviously, if the epipole in the first image does not lay on line ac and bd, that is to say, the center point O' of the second image is not in the plane defined by (ACO) nor in the plane BDO, we will know that 2 view lines Om and $O'm'$ do not intersect, and therefore the points m and m' do not satisfy epipolar correspondence.

The above discussion can be summarized as theorem below:

Theorem: If neither a, b, c nor d are the epipole point of image 2 with respect of image 1, then there exists at least one diagonal intersection m such that m and its corresponding intersection m' satisfy the epipolar constraint if and only if A, B, C, D are coplanar.

Based on which we can easily check the coplanarity of four points and this could be applied in the selection of reference points as we will discuss in the later section.

2) Selecting Five Point Basis with Coplanarity Test: As we have discussed before, in a projective space, we need 5 points in the space to form a basis, any four of which should not be coplanar. Therefore, in this section, we talk about the approaches proposed in the original paper on how to select five point basis using the coplanarity test method presented above.

Two algorithms are provided in the original paper for selecting the reference points, the first one is a greedy algorithm that can be summarized step by step as follows:

- Arbitrarily choose points M_1, M_2 from the 3D world frame, in our case we choose points that have occurences across all images used for 3D reconstruction.
- Choose any spatial point M_3 that is not colinear with M_1M_2, i.e., not on the line M_1M_2.
- Randomly choose M_4 that is not within the plane $M_1M_2M_3$.
- Choose M_5 that is not on any face of tetrahedron M_1, M_2, M_3, M_4, i.e., a point that is not coplanar with any 3 points in the previously selected ones.

Be noticed that the greedy algorithm is mathematically correct in that it guarantees the non-coplanarity of any 4 points in our selection. However, there are always noises in image measurements, the coplanarity should not be the only concern when determining the 5 point basis. For example, if we choose points that are too close to each others, the influences of errors in images will be much more significant than the case that we select points as far from each others as possible. Therefore, the authors improved the algorithm as follows:

- Choose M_1 and M_2 the farthest points pair from one of the image.
- Choose the M_3 farthest point from line M_1M_2.
- Choose M_4 that is farthest away from the plane $M_1M_2M_3$. For distance comparison, we use the projection of line segment from M_4 to the plane on the second image (as shown in Fig. III-B2), instead of the orthogonal distance.
- Sort remaining points according to the maximum distance to any face of tetrahedron M_1, M_2, M_3, M_4, choose for M_5 the point which has the maximum distance.

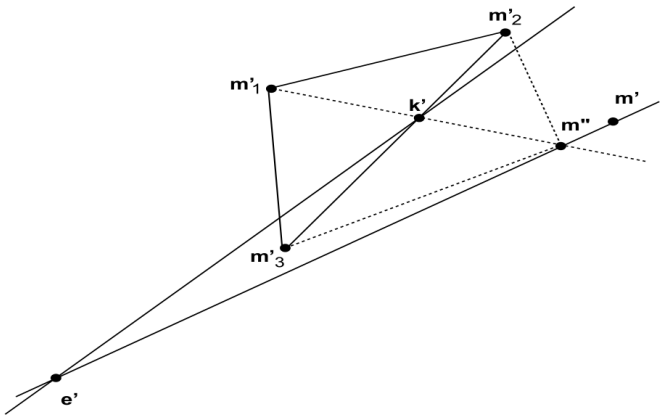


Fig. 4. Distance Metric Used for Selecting Basis

C. 3D Reconstruction with Self-Calibration

So far, we have established the basic equation systems based on camera projection geometry as well as the approach for determining reference points using the coplanarity test method proposed in the paper. In this section, we discuss the approach used in the original paper to reconstruct 3D objects by solving the equation system using nonlinear least square routines.

1) Direct Nonlinear Solution: A natural way of reconstructing based on those equations is to solve them directly. Before that, we can add a constraint $x_i^2 + y_i^2 + z_i^2 + t_i^2 = 1$ for each 3D point since the projective coordinates of the spatial points are only defined up to a constant. Apparently the system is overdetermined and we can leverage standard least squares technique to solve it. In this way, the problem should be formulated as follows:

$$F(x_i, y_i, z_i, t_i, p_{11}^{(j)}, ...p_{34}^{(j)}) = \sum_{k=1}^{2\times m\times n+n} \left(\frac{f_k(u_{ij}, v_{ij}; x_i, y_i, z_i, t_i, p_{11}^{(j)}, ...p_{34}^{(j)})}{\sigma_k} \right)^2$$

over

$$(x_i, y_i, z_i, t_i, p_{11}^{(j)}, ...p_{34}^{(j)}) \quad \text{for} \quad i = 1, \ldots, n, \quad j = 1, \ldots, m,$$

where $f_k(\cdot)$ is either

$$u_{ij} - \frac{p_{11}^{(j)}x_i + p_{12}^{(j)}y_i + p_{13}^{(j)}z_i + p_{14}^{(j)}t_i}{p_{31}^{(j)}x_i + p_{32}^{(j)}y_i + p_{33}^{(j)}z_i + p_{34}^{(j)}t_i}$$

or

$$v_{ij} - \frac{p_{21}^{(j)}x_i + p_{22}^{(j)}y_i + p_{23}^{(j)}z_i + p_{24}^{(j)}t_i}{p_{31}^{(j)}x_i + p_{32}^{(j)}y_i + p_{33}^{(j)}z_i + p_{34}^{(j)}t_i}$$

subject to

$$x_i^2 + y_i^2 + z_i^2 + t_i^2 - 1 = 0 \quad \text{for} \quad i = 1, \ldots, m.$$

Fig. 5. 3D Reconstruction Equation System Formulated as Minimization Problem

Be noticed that σ_k is the standard deviation of each image measure u_{ij}, v_{ij}, which is assumed to distribute normally and uncorrelatedly. We can further transfer the constraint $x_i^2+y_i^2+z_i^2+t_i^2-1=0$ into additional penalty functions (since the constraint won't hold absolutely due to noises, what we can do is only to minimize it, similar to other equations), and the whole problem is an unconstrained least squares problem.

Similar to the projects regarding camera calibration we did before, here we can assume $p_{34}^j = 1, \forall j \in [1, 2, \ldots, n]$. The only variables we know beforehand in the equation system are the image measurements (u_{ij}, v_{ij}). All other parameters are assumed unknown in the paper for estimation. With that, the system finally leads to $n+2\times n\times m$ equations in $11\times m+3\times n$ unknowns.

In our final implementation, we adopted the alternative of F function by minimizing the following system:

The advantages of doing so are as follows:

$$G(x_i, y_i, z_i, t_i, p_{11}^{(j)}, ...p_{34}^{(j)}) = \sum_{k=1}^{2\times m\times n+n} \left(\frac{g_k(u_{ij}, v_{ij}; x_i, y_i, z_i, t_i, p_{11}^{(j)}, ...p_{34}^{(j)})}{\sigma_k} \right)^2,$$

where $g_k(\cdot)$ is either

$$u_{ij}(p_{31}^{(j)}x_i + p_{32}^{(j)}y_i + p_{33}^{(j)}z_i + p_{34}^{(j)}t_i) - (p_{11}^{(j)}x_i + p_{12}^{(j)}y_i + p_{13}^{(j)}z_i + p_{14}^{(j)}t_i)$$

or

$$v_{ij}(p_{31}^{(j)}x_i + p_{32}^{(j)}y_i + p_{33}^{(j)}z_i + p_{34}^{(j)}t_i) - (p_{21}^{(j)}x_i + p_{22}^{(j)}y_i + p_{23}^{(j)}z_i + p_{24}^{(j)}t_i).$$

Fig. 6. Degraded Minimization Problem for 3D Reconstruction

- Degree of nonlinearity of the equation system is greatly reduced.
- The Jacobian matrix of function g_k is especially nicely reduced.
- Faster convergence when using nonlinear least squares routines.

However, such a alternative also transformed the real Euclidean distance into an algebraic distance so that the error function is degraded.

D. From Projective to Affinity/Euclidean

As we can infer from the above sections, with the 5 reference points as basis in 3D space, the solution we acquired by solving the system provides both the projective shape (defined up to a collineation) of objects and the projection matrices of each camera. However, so far there is no metric information implied in our projective reconstruction and only projective properties are preserved. Similarities between current projective space and 3D world include but are not limited to:

- Aligned points remain aligned.
- Coplanar points remain coplanar.
- Conics are transformed into conics.
- A circle may be represented by an hyperbole.

As we have learnt from previous projects and courseworks, pure projective shape can be transformed into affine or Euclidean space, which requires additional information about the affinity and Euclidean space. For transformation to affinity, we have to determine a collineation $\mathbf{A}$, a 4×4 matrix that brings the canonical basis $\mathbf{e}_i, \forall i \in [1, \ldots, 5]$ to any 5 points like follows:

$$\mathbf{a}_i = (a_{i1}, a_{i2}, a_{i3}, a_{i4})^T = \mathbf{A}\mathbf{e}_i \tag{3}$$

All point correspondences in two views are shown in Fig. III-D.

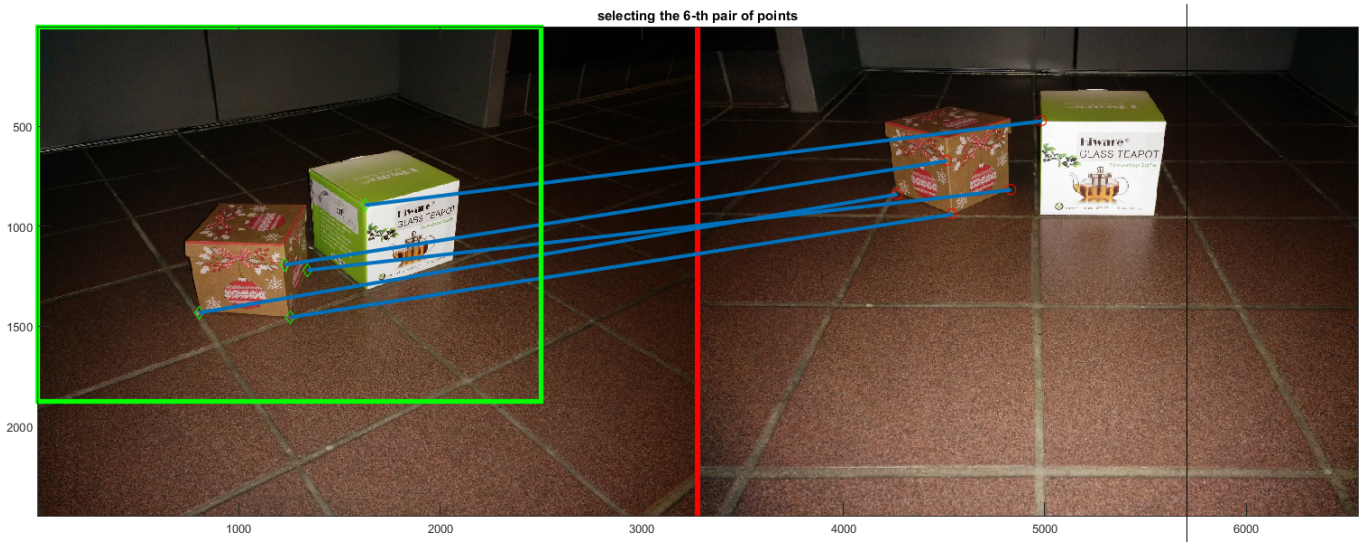


Fig. 7. Reference Point Pairs Used for Reconstruction

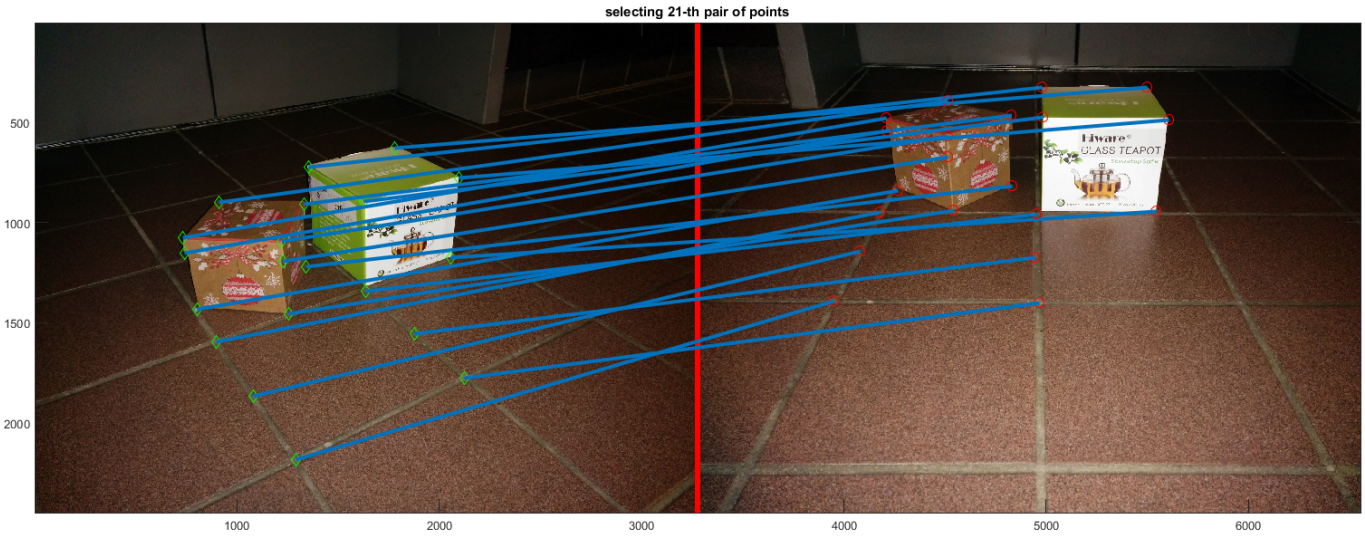


Fig. 8. All Point Correspondences Used for Reconstruction

Assuming that the five points here are only affinely known to us, four of which can be assigned the standard affine coordinates, the coordinates of the fifth point must be affine coordinates with respect to these five points. Therefore, we can assume the following coordinates for the 5 points:

$$(1,0,0,1)^T, (0,1,0,1)^T, (0,0,1,1)^T, \\ (1,1,1,1), (\alpha,\beta,\gamma,1)^T \tag{4}$$

In short, we need to acquire the affine information and incorporate them into α, β, γ for initialization of the 5 reference points so that we can obtain the collineation that transforms a pure projective shape into an affine shape by solving the equation system.

For transformation to usual Euclidean space, similar to other approaches we have to know the exact coordinates of 5 reference points in 3D space, with which we can compute the corresponding collineation that transforms a pure projective shape into an usual Euclidean shape. Another method for acquiring Euclidean reconstruction is to simply use the actual coordinates when initializing the nonlinear least squares equation system.

IV. Performance Evaluation

In this section, we measure the performance of our implemented approach through a series of experiments. At first, we introduce how to initialize the parameters used for minimization, and then we talk about the setup of our experiment. Furthermore, we present both the graphical and numerical results we obtained in running the implemented method.

A. Data Initialization

Though the convergence does not depend too much on the initial starting points, it is suggested in the original paper that we use the following data for initializing the five reference points:

$$(0,0,0,1)^T, (1,0,0,1)^T, (0,1,0,1)^T, (0,0,1,1)^T, (1,1,1,1)^T \tag{5}$$

It is worthmention that though the result is not quite sensitive on reference points initialization, a totally wrong setup might make the equation system diverge. Strangely, the initialization of other normal points and projection matrices seem to influence the results a lot. According to the paper we adopted:

$$\begin{pmatrix} 0.1 & 0.1 & 0.1 & 20 \\ 0.1 & 0.1 & 0.1 & 20 \\ 0.1 & 0.1 & 0.1 & 1 \end{pmatrix} \tag{6}$$

for initializing projection matrices and:

$$(0.5, 0.5, 0.5, 1) \tag{7}$$

for points other than those reference points. Also recall that we can initialize the reference points with affine/Euclidean information to directly obtain affine/Euclidean shape.

B. Experiments

In our experiments, we used an android smartphone as the camera. Since the smartphone is incorporated with auto-focus, its focal lengths and other properties keep changing when taking photos, this would better illustrate the advantage of our approach as we don't need calibration before reconstruction.

As shown in Fig. III-D, the reference points we choose in our experiments are selected to satisfy the non-coplanarity constraint we mentioned above.

We refer readers to the appendix for more numerical data about point correspondences.

The result we get from 3D reconstruction in comparison with the original 3D points is shown in Fig. IV-B.

Overall, the reconstructed 3D points are located closely to the original ones. But be noticed that this result is actually acquired after some rotation and translation (the original reconstructed points look like a mess but the shape is nearly the same). Furthermore, we tried to apply this approach to other circumstances but got terrible results that we cannot make full sense. The result comparison between last project (reconstruction after calibration) and our final project is provided in 10. So far, we still cannot make sense of this strange reconstruction results, the only doubt I have is that there might be something wrong with the choice of reference points, since in the last project the only non-coplanar points from both wall and floor plane actually lies in a plane very close to wall plane, which means the influence of noises might be significant in this case.

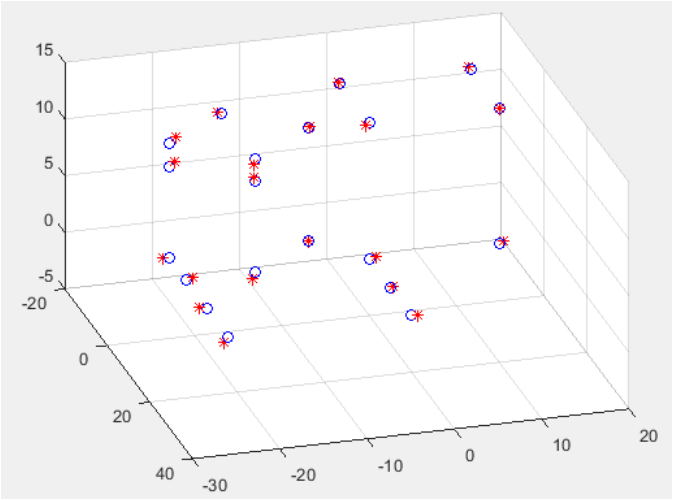


Fig. 9. Results Comparison between Reconstructed 3D Points and Original Ones

V. Conclusion

In this final project, we reconstructed and implemented the paper [2] from scratch. The approach implemented here has several advantages over previous work:

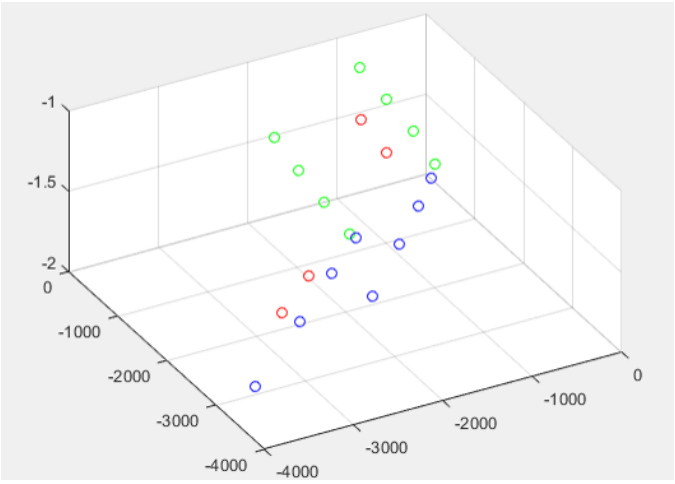
- This approach requires no pre knowledge of the camera, i.e., no camera calibration is needed before 3D reconstruction.
- A novel method for simultaneously calibrating camera and reconstructing the 3D objects is proposed leveraging standard nonlinear least squares estimation, the redundant use of data further reduced overall errors in the reconstruction process.
- Two automatic reference point selection routines based on coplanarity test is developed together with the method by leveraging the nice properties after epipolar geometry information is obtained.

Be noticed that we didn't implement the coplanarity test part due to time limitations, however we did follow the algorithm procedure in the experiments while selecting reference point for 3D reconstruction and guaranteed that:

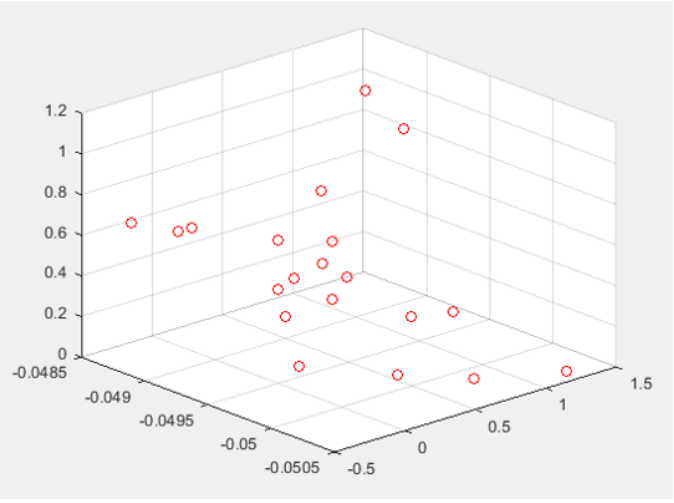
- No four of the reference points we choose are coplanar.
- The distance (not necessarily real orthogonal distance) between any ref point and the faces of tetrahedron formed by other ref points is maximized.

The first guarantee ensured that we don't get degenerated solutions in the estimation process while the second one greatly reduces the influence of measurement errors in the final 3D reconstruction results.

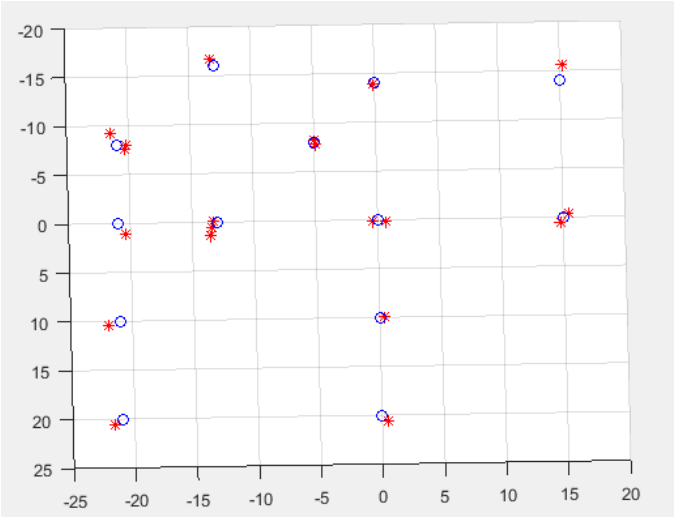
Through our experiments, we saw that the results are good, we did a little comparison with previous project we did and it turned out that the new approach yields similar accuracies while it does not even require camera calibration.



(a) Reconstruction Results from Last Project



(b) Reconstruction Results using Approach Implemented in This Paper



(c) Projection of Reconstructed and Original Points on xy Plane for Comparison

Fig. 10. Reconstruction Results using Approach Implemented in This Paper

VI. Discussions

In this section, we will focus on the disadvantages and possible improvements that can be made for the implemented method as the advantages have already been highlighted in previous sections. There are several possible sources of inaccuracy when applying the uncalibrated reconstruction approach:

- Imperfect point correspondence: this is the natural first thought of inaccuracy source. In our experiments, we select points manually from the images. The reason of doing so is that we found though the *c*orner function in MATLAB performs well in finding corners on high-contrast checkerboard, it acts bad when applied to our measured images (which is captured in dark environments with flash light). Additionally, the corner function is totally separate from our method, i.e., it has no idea about the reference points selection and spatial coplanarity. Therefore, we select point correspondences manually to ensure good choice of ref points. However, the manual approach on the other hand creates a lot of errors since we can hardly identify the ideal points only by hand and the point correspondence is just an approximation.
- Location accuracy in the image measurements: aside from the errors caused by manual selection, the camera also generates a lot of measurement errors, especially in the dark-light environment.
- Another source of inaccuracy is the lens distorsion. Similar to the original paper, we assume throughout this paper that the noise on the measures is uncorrelated, Gaussian and centered. Lens distorsion introduces correlated noise which is obviously not centered. However, the good news is that we can easily remove the lens distortion using previous implemented approach.

Though disadvantages still exist, overall, the implemented approach can yield very good results while requiring very few

knowledge for the camera system and 3D ground truth.

VII. Future Work

Due to time limitations, the approach we implemented in this project is still far from mature. Also we actually didn't do much on numerical analysis of reprojected errors, most of the results we obtained are graphical ones. Additionally, though the coplanarity test is applied in our point selection process, we didn't implement it in code, this should also be left as future work. I'm about to catch up the flight right now, thanks for your understanding for the inconvenience I caused and hope you have a nice summer!

References

[1] Wikipedia, "3D reconstruction from multiple images," https://en.wikipedia.org/wiki/3D_reconstruction_from_multiple_images.

[2] R. Mohr, L. Quan, and F. Veillon, "Relative 3d reconstruction using multiple uncalibrated images," *The International Journal of Robotics Research*, vol. 14, no. 6, pp. 619–632, 1995.

[3] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *International journal of computer vision*, vol. 8, no. 2, pp. 123–151, 1992.

[4] W. H. Press, B. P. Flannery, and S. A. Teukolsky, "Wt vetterling numerical recipes in c," *Cambridge University Press*, vol. 1, p. 988, 1988.

Appendix

A. Numerical Data

The point correspondences from images in the last project are:

$$\begin{pmatrix} 664.3 & 75.34 \\ 676.2 & 562.6 \\ 685.0 & 1040.0 \\ 723.5 & 1467.0 \\ 1262.0 & 142.2 \\ 1268.0 & 546.7 \\ 1271.0 & 935.2 \\ 1276.0 & 1305.0 \\ 883.2 & 1005.0 \\ 871.3 & 559.4 \\ 791.5 & 1534.0 \\ 948.2 & 1610.0 \\ 1137.0 & 1706.0 \\ 1359.0 & 1808.0 \\ 1265.0 & 1375.0 \\ 1427.0 & 1426.0 \\ 1616.0 & 1493.0 \\ 1832.0 & 1556.0 \\ 1498.0 & 1301.0 \\ 1646.0 & 1340.0 \end{pmatrix} \tag{8}$$

$$\begin{pmatrix} 830.2 & 518.0 \\ 836.1 & 916.1 \\ 818.4 & 1305.0 \\ 821.3 & 1661.0 \\ 1407.0 & 451.1 \\ 1386.0 & 948.0 \\ 1368.0 & 1413.0 \\ 1336.0 & 1843.0 \\ 1203.0 & 1381.0 \\ 1223.0 & 932.0 \\ 821.3 & 1738.0 \\ 646.9 & 1789.0 \\ 451.7 & 1849.0 \\ 229.9 & 1910.0 \\ 1268.0 & 1900.0 \\ 1085.0 & 1973.0 \\ 880.5 & 2059.0 \\ 641.0 & 2161.0 \\ 1596.0 & 2028.0 \\ 1436.0 & 2114.0 \end{pmatrix} \tag{9}$$

The point correspondences found in this image are as follows:

$$\begin{pmatrix} 1256.0 & 1455.0 \\ 804.0 & 1433.0 \\ 1341.0 & 1220.0 \\ 1229.0 & 1193.0 \\ 1628.0 & 894.2 \\ 732.3 & 1075.0 \\ 1220.0 & 1093.0 \\ 898.0 & 1596.0 \\ 1355.0 & 722.2 \\ 1292.0 & 2184.0 \\ 2124.0 & 1777.0 \\ 2098.0 & 776.5 \\ 1780.0 & 627.2 \\ 2057.0 & 1175.0 \\ 1637.0 & 1342.0 \\ 1332.0 & 907.8 \\ 911.4 & 898.7 \\ 741.3 & 1152.0 \\ 1081.0 & 1867.0 \\ 1878.0 & 1555.0 \end{pmatrix} \tag{10}$$

$$\begin{pmatrix} 2016.0 & 939.4 \\ 1743.0 & 839.9 \\ 2307.0 & 817.3 \\ 1990.0 & 672.5 \\ 2460.0 & 473.4 \\ 1685.0 & 477.9 \\ 1990.0 & 568.4 \\ 1654.0 & 948.5 \\ 2455.0 & 324.0 \\ 1426.0 & 1392.0 \\ 2442.0 & 1401.0 \\ 3082.0 & 486.9 \\ 2974.0 & 324.0 \\ 3019.0 & 944.0 \\ 2428.0 & 957.5 \\ 2303.0 & 464.3 \\ 1990.0 & 391.9 \\ 1690.0 & 541.2 \\ 1555.0 & 1143.0 \\ 2419.0 & 1175.0 \end{pmatrix} \tag{11}$$

B. Code Section

1) Fake 3D Point Simulator: This is a subroutine that can be used to generate image projection from 3D points for testing reconstruction algorithm before applying into real image.

```
% fake 3D points
X=[10 0 0 1;
    0 10 0 1;
    0 0 10 1;
    0 0 0 1;
    10 10 10 1;

    10 10 0 1;
    10 0 10 1;
    0 10 10 1;
    -1 8 9 1;
    -2 -5 -3 1;

    -1 6 2 1;
    2 -1 -3 1
    7 3 1 1
    9 1 2 1
    -4 -5 3 1]';
% count of image points
n = 15;
% plot original points in 3D
figure (1)
scatter3(X(1, 1:n), X(2, 1:n), X(3, 1:n), 'ro');
hold on;
% fake the first projection matrix
rx1=0; ry1=1 * pi; rz1=1 * pi;
    Rx1=[1 0 0;
         0 cos(rx1) -sin(rx1);
         0 sin(rx1)  cos(rx1)];
    Ry1=[cos(ry1) 0 sin(ry1);
         0 1 0;
         -sin(ry1) 0 cos(ry1)];
    Rz1=[cos(rz1) -sin(rz1) 0;
         sin(rz1)  cos(rz1) 0;
         0 0 1];
    % rotation matrix
    R1=Rz1*Ry1*Rx1;

    tx1=250; ty1=500; tz1=1000;
    % translation matrix
    T1=[tx1;ty1;tz1];
    % rotation and translation
    RT1=[R1 T1;
         0 0 0 1];
    % intrinsic matrix
    K1=[1 0 1 0;
         0 1 1 0;
         0 0 1 0];
    P1=K1*RT1;
    % project fake 3D pts to image plane
    x1 = P1*X;
    % homogenize
```

```
53      x1 = bsxfun (@rdivide, x1, x1(3,:));
54      % the second projection matrix
55      rx2=0; ry2=0; rz2= pi;
56      Rx2=[1 0 0;
57           0 cos(rx2) -sin(rx2);
58           0 sin(rx2)  cos(rx2)];
59      Ry2=[cos(ry2) 0 sin(ry2);
60           0 1 0;
61           -sin(ry2) 0 cos(ry2)];
62      Rz2=[cos(rz2) -sin(rz2) 0;
63           sin(rz2)  cos(rz2) 0;
64           0 0 1];
65      % rotation matrix
66      R2=Rz2*Ry2*Rx2;
67
68      tx2=500; ty2=500; tz2=1000;
69      T2=[tx2;ty2;tz2];
70      % arbitrary rotation and translation
71      RT2=[R2 T2;
72           0 0 0 1];
73      % camera intrinsic
74      K2=[1 0 1 0;
75           0 1 1 0;
76           0 0 1 0];
77      % projection matrix
78      P2=K2*RT2;
79      x2 = P2*X;
80      % homogenize
81      x2 = bsxfun (@rdivide, x2, x2(3,:));
82  % u_ij is the x coordinate of jth point in ith image
83  u = [x1(1, :);
84       x2(1, :)];
85  v = [x1(2, :);
86       x2(2, :)];
87  % how many images
88  m = 2;
89  % how many point correspondences
90  n = 15;
91  % initialize the minimization input
92  P_init = [0.1 0.1 0.1 20 0.1 0.1 0.1 20 0.1 0.1 0.1];
93  x_0 = [P_init P_init];
94
95  x_s = [10 0 0 0 10 (0.5*ones(1, 10))];
96  y_s = [0 10 0 0 10 (0.5*ones(1, 10))];
97  z_s = [0 0 10 0 10 (0.5*ones(1, 10))];
98  x_0 = [x_0 x_s y_s z_s];
99
100 [err] = cost_function(x_0);
101
102 %options.Algorithm = 'levenberg-marquardt';
103 options = optimoptions('lsqnonlin','Algorithm','levenberg-marquardt'...
104 ,'TolX',10e-10,'TolFun',10e-10);
105 x = lsqnonlin(@cost_function, x_0, [], [], options);
106 [final_cost] = cost_function(x);
107
108 X = x(1, (m*11+1) : (m*11+n));
```

```
Y = x(1, (m*11+n+1):(m*11+2*n));
Z = x(1, (m*11+2*n+1):(m*11+3*n));
% scatter the reprojected pts
figure (1)
scatter3(X(1, 1:n), Y(1, 1:n), Z(1, 1:n), 'b*')
hold on;
```

```
close all
clear all
clc

im1 = imread('pics/left2obj.jpg');
im2 = imread('pics/right2obj.jpg');

seperate_line_width=25;
seperate_line=zeros(2448,seperate_line_width,3);
seperate_line(:,:,1)=255;
%im1(1876:2500, 1:2500) = zeros((2500 - 1875), 2500);
im_com=[im1 seperate_line im2];

figure_handler=figure(1);
image(im_com)
hold on
set(gcf,'Units','centimeters', 'Position',[0 3.5 50 15],'Renderer','painters','color'
    ,'white');
% all odd points we selected are supposed to be in the first figure
pts_img_1 = [];
% all even points we selected are supposed to be in the second figure
pts_img_2 = [];
if exist('points1.mat','file')>0
    load('points1.mat');
    for j=1:length(x1)
        im1(floor(y1(j))-10:floor(y1(j))+10,floor(x1(j))-10:floor(x1(j))+10,1:2)=255;
        im1(floor(y1(j))-10:floor(y1(j))+10,floor(x1(j))-10:floor(x1(j))+10,3)=0;
        im2(floor(y2(j))-10:floor(y2(j))+10,floor(x2(j))-10:floor(x2(j))+10,1)=0;
        im2(floor(y2(j))-10:floor(y2(j))+10,floor(x2(j))-10:floor(x2(j))+10,2)=250;
        im2(floor(y2(j))-10:floor(y2(j))+10,floor(x2(j))-10:floor(x2(j))+10,3)=150;
        A(2*j-1:2*j,:)=[  0,    0,    0, -x1(j), -y1(j), -1,  y2(j)*x1(j),  y2(j)*y1(
            j),  y2(j);
                        x1(j), y1(j), 1,   0,      0,      0, -x2(j)*x1(j), -x2(j)*y1(
                            j), -x2(j)];
        plot(x1(j),y1(j),'gd','MarkerSize',8);
        plot(x2(j)+(2500+seperate_line_width),y2(j),'ro','MarkerSize',8);
        line([x1(j),x2(j)+(2500+seperate_line_width)],[y1(j),y2(j)],'linewidth',3);
    end
else
    i=0;
    while i<40
        title( ['selecting the ',num2str(floor(i/2)+1),'-th pair of points'])
        j=floor(i/2)+1;
        if mod(i,2)==0
            h=rectangle('Position',[0.01 0.01 25 18.75]*100,'EdgeColor','g','
                LineWidth',4);
            [x1(j),y1(j)]=ginput(1);
            if x1(j)<=2500
                plot(x1(j),y1(j),'gd','MarkerSize',8)
```

```
                    pts_img_1 = [pts_img_1;
                        x1(j), y1(j)]
                    i=i+1;
                else
                    disp('please select point inside the green box');
                end
            else
                h=rectangle('Position',[25+seperate_line_width/100 0.01 18.74 24.99]*100,
                    'EdgeColor','g','LineWidth',4);
                [x2(j),y2(j)]=ginput(1);
                x2(j)=x2(j)-(2500+seperate_line_width);
                if x2(j)>=0
                    plot(x2(j)+(2500+seperate_line_width),y2(j),'ro','MarkerSize',8);
                    line([x1(j),x2(j)+(2500+seperate_line_width)],[y1(j),y2(j)],'
                        linewidth',3);
                    pts_img_2 = [pts_img_2;
                        x2(j), y2(j)]
                    i=i+1;
                else
                    disp('please select point inside the green box');
                end
            end
        % im_com=[im1 seperate_line im2];
            delete(h);
            %im_com=[im1 seperate_line im2];
            %image(im_com)
            title( ['selecting ',num2str(floor(i/2)+1),'-th pair of points'])
        end
    end

%save('points.mat', 'pts_img_1', 'pts_img_2');
```

```
close all; clear all; clc;

% load point correspondences into variable pts_img_1 and pts_img_2
load('points_5.mat');

% number of pictures
global m n u v;
m = 2;
% number of pts
n = 20;
pts_img_1 = pts_img_1;
pts_img_2 = pts_img_2;
% pts in images
u = [];
u = [u; pts_img_1(:, 1)'; pts_img_2(:, 1)'];
v = [];
v = [v; pts_img_1(:, 2)'; pts_img_2(:, 2)'];

P_init = [0.1 0.1 0.1 20 0.1 0.1 0.1 20 0.1 0.1 0.1];

% initializing Projection matrices and reference points as well as ordinary
% points
x_0 = [P_init P_init];
x_s = [0 -8 -8 0 0];
```

```
x_s = [x_s (0.5 * ones(1, 15))];
y_s = [-12 -20 -4 -12 0];
y_s = [y_s (0.5 * ones(1, 15))];
z_s = [0 0 0 6 12.5];
z_s = [z_s (0.5 * ones(1, 15))];

% x_s = 0.5 * ones(1, 20);
% y_s = 0.5 * ones(1, 20);
% z_s = 0.5 * ones(1, 20);

%t_s = ones(1, 20);
x_0 = [x_0 x_s y_s z_s];
[cost] = cost_function(x_0);
options = optimoptions('lsqnonlin','Algorithm','levenberg-marquardt'...
,'TolX',10e-10,'TolFun',10e-10);
x = lsqnonlin(@cost_function, x_0, [], [], options);
[final_cost] = cost_function(x);
X = x(1, (m*11+1) : (m*11+n));
Y = x(1, (m*11+n+1):(m*11+2*n));
Z = x(1, (m*11+2*n+1):(m*11+3*n));
% scatter the pts on the wall using green markers
scatter3(X(1, 1:n), Y(1, 1:n), Z(1, 1:n), 'r')
hold on;



% X_2 = [20 10 0 -8 -8 -8 -16 0 0 0 -8 -8 20 10 0 0 -14 0 0 -14];
% X_2 = [X_2;
%     -21 -21 -21 -21 -21 -21 -13 -13 -13 -13 -5 -5 0 0 0 0 0 15 15 15];
% X_2 = [X_2;
%     0 0 0 0 8 10 0 8 10 10 0 10 0 0 0 12 12 0 12 12];
% scatter3(X_2(1, 1 : 20),X_2(2, 1 : 20),X_2(3, 1 : 20), 'b')

% compute standard deviation
% x_0 = [p_11, ..., p_34; ...
% u_11, ... u_1n;
% ...;
% u_m1, ... u_mn;
% v_11, ... v_1n;
% ...
% v_m1, ... v_mn]

% function t = t_ij(x_0, i, j)
%     global m n;
%     base = m * 11 + m * n *3;
%     index = base + (j - 1) * n + i;
%     t = x_0(index);
% end
```

```
function [cost] = cost_function(x_0)
    cost = [];
    global m n u v;
    x_j = 0;
    y_j = 0;
    z_j = 0;
    t_j = 1;
```

```
    % error of g function
    for i = 1 : m
        p_31 = p_ij(x_0, 3, 1, i);
        p_32 = p_ij(x_0, 3, 2, i);
        p_33 = p_ij(x_0, 3, 3, i);
        p_34 = 1;
            p_11 = p_ij(x_0, 1, 1, i);
            p_12 = p_ij(x_0, 1, 2, i);
            p_13 = p_ij(x_0, 1, 3, i);
            p_14 = p_ij(x_0, 1, 4, i);
        p_21 = p_ij(x_0, 2, 1, i);
        p_22 = p_ij(x_0, 2, 2, i);
        p_23 = p_ij(x_0, 2, 3, i);
        p_24 = p_ij(x_0, 2, 4, i);

       % in image i, the jth point
        for j = 1 : n
            x_j = x_i(x_0, j);
            y_j = y_i(x_0, j);
            z_j = z_i(x_0, j);
            t_j = 1;
           % j is the numbering of point, i is the numbering of img
           % function g()
            tmp_cost_u = u(i, j) * (p_31 * x_j + p_32 * y_j + p_33 * z_j + p_34 * t_j)
                - (p_11 * x_j + p_12 * y_j + p_13 * z_j + p_14 * t_j);
            tmp_cost_v = v(i, j) * (p_31 * x_j + p_32 * y_j + p_33 * z_j + p_34 * t_j)
                - (p_21 * x_j + p_22 * y_j + p_23 * z_j + p_24 * t_j);
            cost = [ cost (tmp_cost_u * tmp_cost_u) (tmp_cost_v * tmp_cost_v) ];
        end
    end
    % uncomment here if not using Euclidean initialization
    % error of the constraint
%     for i = 1 : n
%         cost = [cost (x_j*x_j + y_j*y_j+z_j*z_j+t_j*t_j - 1)];
%     end
end

% get the coordinates of p_ij in m th image
function p = p_ij(x_0, i, j, m)
    index = (m - 1) * 11 + (i - 1) * 4 + j;
    p = x_0(index);
end

% get the coordinates of x_ij, which is the ith pt in jth image
function x = x_i(x_0, i)
    global m n;
    base = m * 11;
    index = base + i;
    x = x_0(index);
end

function y = y_i(x_0, i)
    global m n;
    base = m * 11 + n;
    index = base + i;
    y = x_0(index);
```

```
62 end
63
64 function z = z_i(x_0, i)
65     global m n;
66     base = m * 11 + n *2;
67     index = base + i;
68     z = x_0(index);
69 end
```