3D Reconstruction using Kinect V2

Computer vision project

Maryam Edalati

Nowadays 3D reconstruction technology is common method in many different areas. For example, in Computer Graphics, Computer Animation, Computer Vision, Medical Imaging, Computational Science, etc. For instance, the lesion information of the patients can be presented in 3D on the computer, which offers a new and accurate approach in diagnosis and thus has vital clinical value [1], in industry very accurate models are used for physical simulations or quality tests and additionally, computer games or visualizations are going to be more and more photo-realistic, so the models must look like real objects.

Introduction

Recently 3D reconstruction technology is used in many domains and there are different tools for reconstruct objects. Tools are good options to do 3D reconstruction but with some limitations like specific hardware which are very expensive. In end of 2010, Microsoft released the Kinect camera which has a depth sensor in addition to the RGB-sensor, it is quite cheap camera which is available for everyone and can be used as input for 3D modelling task. It makes opportunity for developers to use geometric input with using colour pictures. With this impact new methods are developed for gesture control systems or 3D reconstruction.

Kinect camera combines colour image camera with an active depth capturing sensors which finds both colour and spatial information about world scene. For each pixel, it provides both the colour information and the measurement of the distance from the camera to the object that is represented by the corresponding pixel.

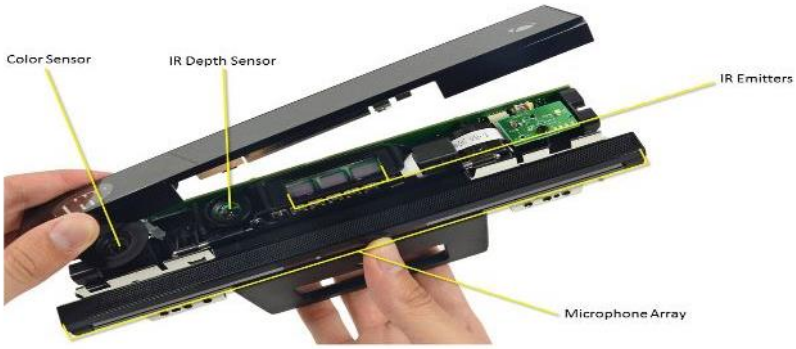
The goal of this project is to create a reliable three-dimensional (3D) model for objects with surfaces using a Kinect v2 camera.

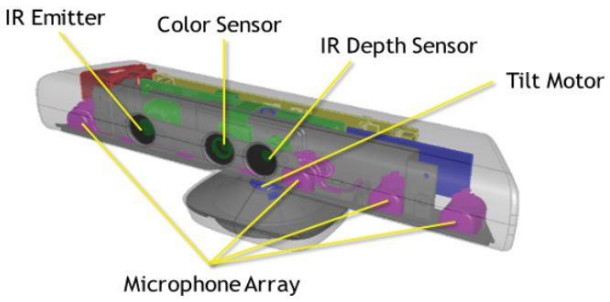
Table of Contents

Kinect V2

History

The second-generation Kinect for Windows v2, based on the Kinect for Xbox One hardware, was first released in 2014. Version 2.0 of the SDK was also released, which supported both the Kinect for Windows v2 and the Kinect for Xbox One, which were internally identical. An adapter cable was required to use the Kinect for Xbox One on a PC [2].





Kinect is a motion sensing input device which is used for Xbox 360 console and also Windows PCs that enables users to control and interact with the application/game without the need to touch a game controller through a natural user interface and using gestures and spoken commands [3].

Technology

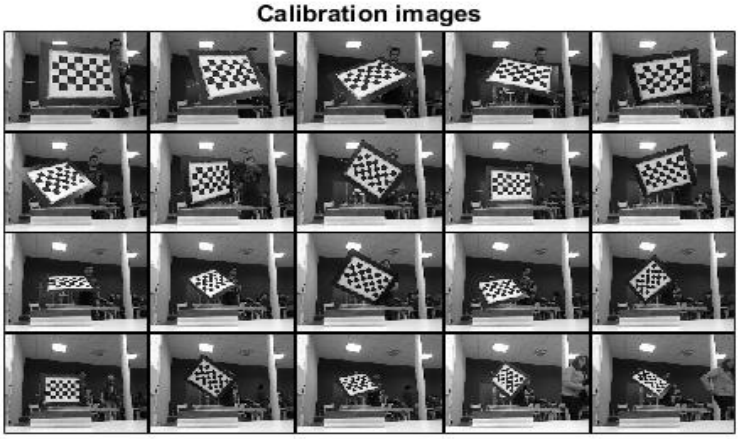
Kinect contains 3D scanner system called Light Coding, special microchip to track the movement of objects and individuals in three dimensions and employs a variant of image-based 3D reconstruction.

Horizontal bar connected to a small base motorized pivot. Designed to be positioned lengthwise above or below the video display consist of an infrared projector, a colour camera, an infrared sensor and Multi-array microphone that enables to acoustic source localization and ambient noise suppression.

The depth sensor captures video data in 3D under any ambient light conditions infrared laser projector and monochrome CMOS sensor.
The Kinect software is capable of automatically calibration based on gameplay based on physical environment. Accommodating for the presence of furniture or other obstacles.
• Kinect can track six people, performing motion analysis for two active players and feature extraction of 20 joints per player

Calibration part

First, we tried to do 3D reconstruction by 20 images in left and right, but some of them pictures even after trying different vale for kc, did not detect exact corner so we had some difficulties in this part for reducing the calibration error that we had mostly with images number 3,4,8,12,11,18. The result of error which is for before and after optimization part is shown in appendix. We did 3D reconstruction with the value w.r.t calibration for 20 images and it was not good. Since calibration error have direct effect on our result because of this error we had some difficulty to align and merge our results, so we tried to remove some pictures that have problem to try different result.



Pixel error: [0.39616; 0.41163]

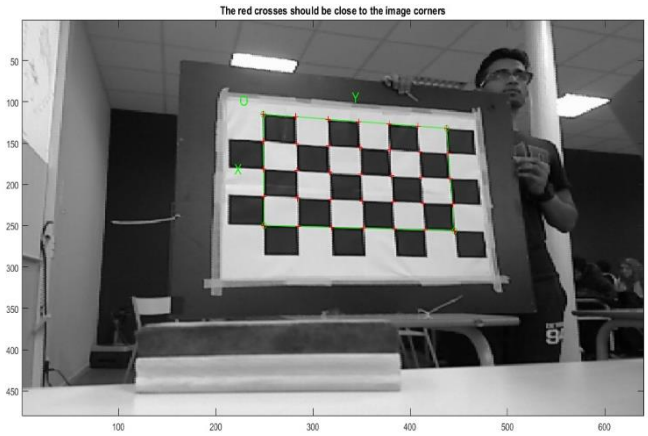
Pixel error after optimization: [0.36916; 0.40066]

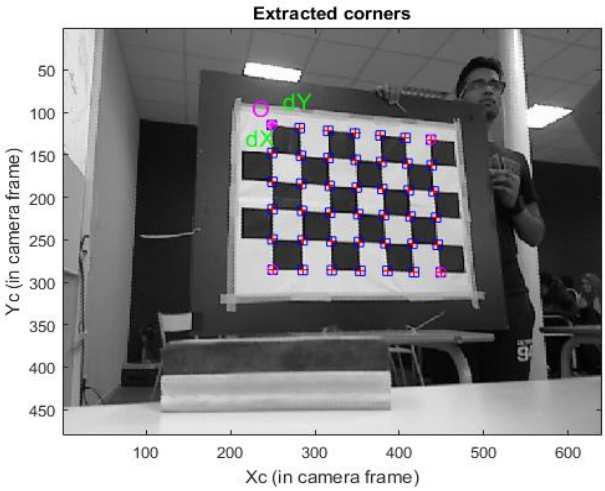
Calibration of left and right images

Firstly, we calibrate the two left and right cameras separately to compute internal parameters of each camera. As we did in the first example of lab session 2 [4], we use the calibration toolbox (calib_gui) for all 14 images by choosing four corners for each image (we removed 3,4,8,11,12,18 images).

1.Read all images with the toolbox (here is for left images since for right images is same procedure.)

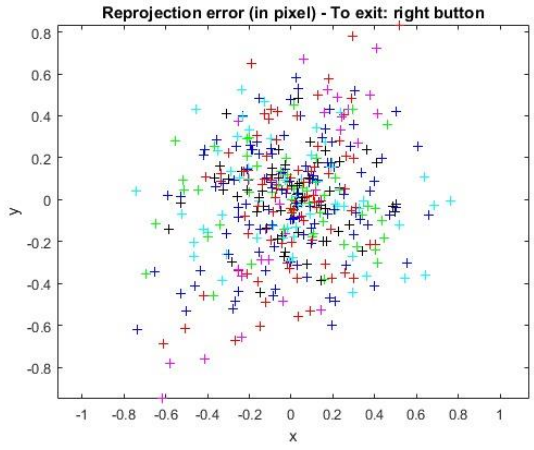
2.Extract corners by choosing the extract grid corner button.

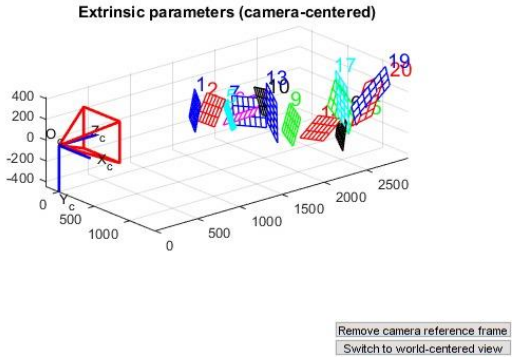


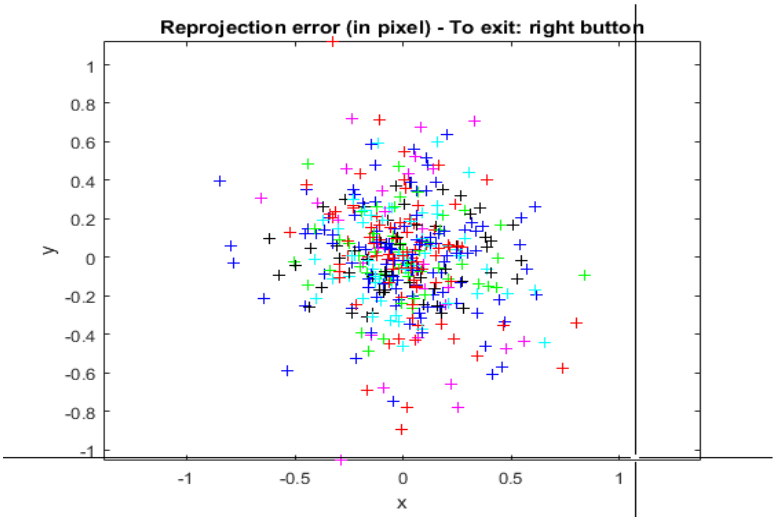


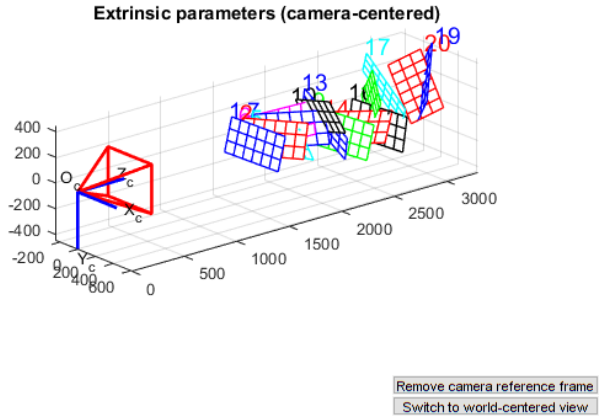
For each image we put window size of the corner finder: wintx=6 and winty=3 and this leads to an effective window of size 11x11 pixels. We select the size of each square in the grid dX=dY=89mm baes on given information by Professor**.**
In some cases, since it did not detect the exact corners we needed to change the value of distortion lense kc between [-1, +1].
After extract corners for all images (first left and next right images), we use calibration button in calib toolbox to do calibration based on all left images and right images seperately. we found the error plot and extrinsic parameters like below:

This part is for left images:







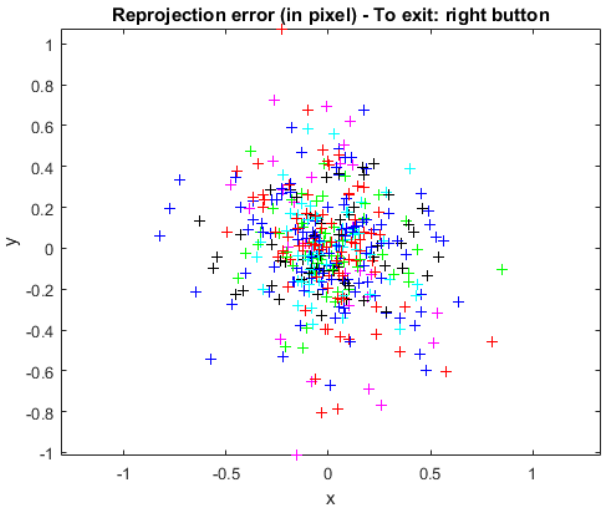


This part is for right images:

Next part is coming from optimization step which we try to minimize error pixel.

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 575.25279   556.80963 ] +/- [ 25.49772   22.63231 ]
Principal point:       cc = [ 234.88928   245.76606 ] +/- [ 23.10223   8.96717 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.29477   -0.19266   -0.01260   -0.09031  0.00000 ] +/- [ 0.13618   0.47702   0.00922   0.02490
Pixel error:          err = [ 0.25631   0.27151 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

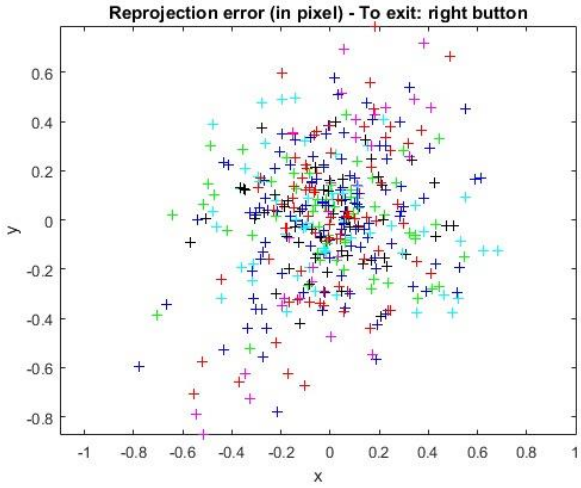
Left part: as you can see the pixel error is reducing significantly.



Right part: as you can see the pixel error is reducing significantly

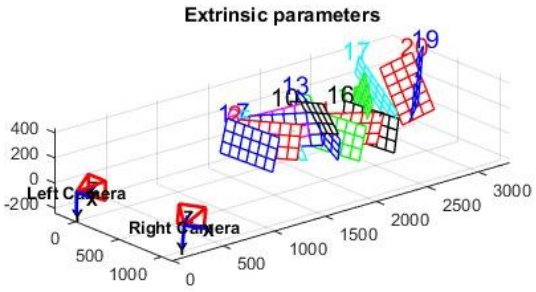
Command Window

```
Initialization of the intrinsic parameters - Number of images: 14


Calibration parameters after initialization:

Focal Length:          fc = [ 496.38876   496.38876 ]
Principal point:       cc = [ 319.50000   239.50000 ]
Skew:             alpha_c = [ 0.00000 ]   => angle of pixel = 90.00000 degrees
Distortion:            kc = [ 0.00000   0.00000   0.00000   0.00000   0.00000 ]

Main calibration optimization procedure - Number of images: 14
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...2
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 560.16449   534.43227 ] +/- [ 26.05968   18.19494 ]
Principal point:       cc = [ 230.88161   260.87561 ] +/- [ 25.92926   9.96698 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ 0.34885   -0.24136   -0.00214   -0.09977  0.00000 ] +/- [ 0.11087   0.15004   0.00661   0.02458
Pixel error:          err = [ 0.26839   0.27818 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).

>>
```



Calibration Stereo

This part is based on the result from calibration part. We found ‘Calib_Results_left’ and ‘Calib_Results_right’ and now we can find the position of camera left with respect to camera right which is done by using the stereo_gui function from calibration toolbox.
In below figure as we can see, the intrinsic parameters of both left and right camera separately displayed as they are computed based on the first example and in the last part extrinsic parameters of the right camera with respect to the left camera is shown.




```
Command Window
Loading the right camera calibration result file Calib_Results_right.mat...


Stereo calibration parameters after loading the individual calibration files:


Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 574.27681   557.13669 ] ± [ 24.26283   21.72149 ]
Principal point:       cc_left = [ 238.42708   245.73389 ] ± [ 22.45032   8.76422 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ 0.30059   -0.19518   -0.01317   -0.08796  0.00000 ] ± [ 0.13079   0.48757   0.00921   0.024


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 562.25454   536.21260 ] ± [ 24.92714   17.32659 ]
Principal point:       cc_right = [ 230.67227   261.58307 ] ± [ 24.52776   9.42083 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ 0.35969   -0.25236   -0.00106   -0.10127  0.00000 ] ± [ 0.10673   0.14612   0.00631   0.02


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ -0.03577   0.72453  -0.12164 ]
Translation vector:           T = [ -883.17935   71.32935  467.33186 ]
fx >>
```

After calibration part and calibration stereo part, we have some information that we will use for 3D reconstruction part.
From left and right cameras calibration:

- Focal length
- Principal Point

From stereo calibration:

- Rotation vector (position of left with respect to right camera)
- Translation vector (position of left with respect to right camera)

Left images:

Rgb_left_fx = 574.276807728040010
Rgb_left_fy = 557.136687864597430;
Rgb_left_cx = 238.427082927314870;
Rgb_left_cy = 245.733893045488630;

Right images:

Rgb_right_fx = 562.254541726260640;
Rgb_right_fy = 536.212604992101550;
Rgb_right_cx = 230.672270128398340;
Rgb_right_cy = 261.583065556063900;

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector: om = [-0.03577 0.72453 -0.12164]

Translation vector: T = [-883.17935 71.32935 467.33186]

Calibrate the two depth cameras with respect of RGB

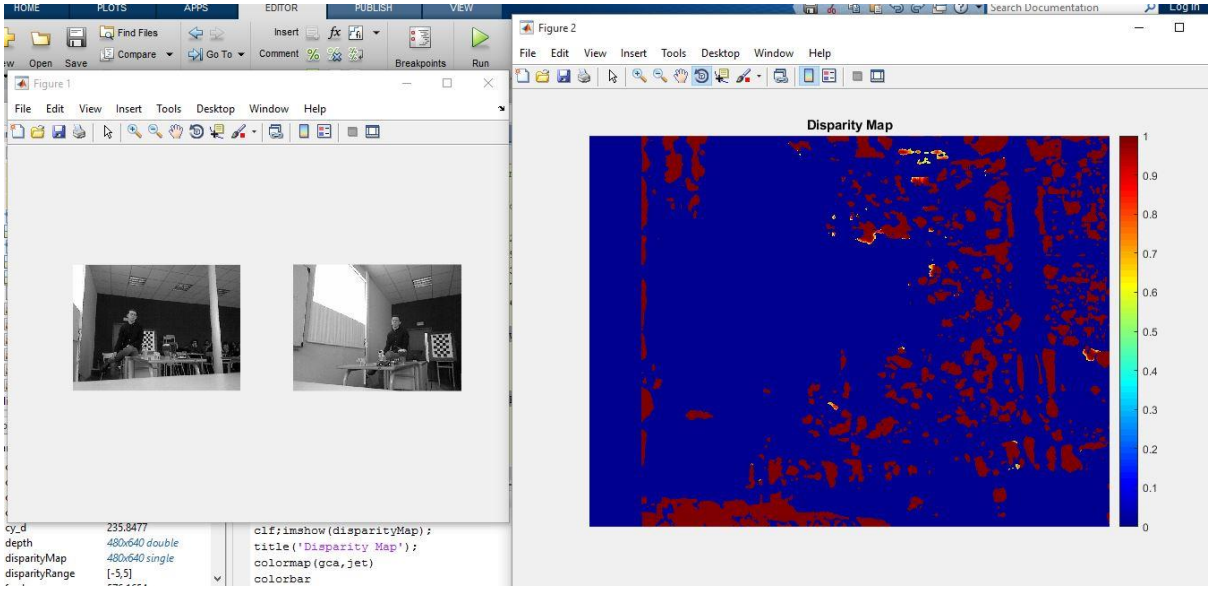
For doing this part we went through this site https://es.mathworks.com/matlabcentral/fileexchange/53439-kinect-2-interface-for-matlab and we download the K2 toolbox for matlab. Then try to run Run compile_cpp_files.m by using Visual Studio 2017 but during its process it aske for Matlab's trial code. Because we do not have Matalb trail code due to use free version of it so we could not able to work on this part.

3D reconstruction

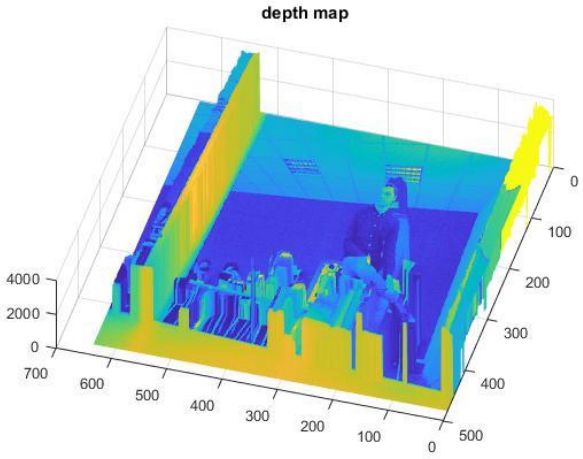
In this part we want to do reconstruction for these images which are taken by the two Kinect v2 cameras. The parameters of left and right cameras are found in calibration part and the position of left camera with respect to right camera is found in calibration stereo part.

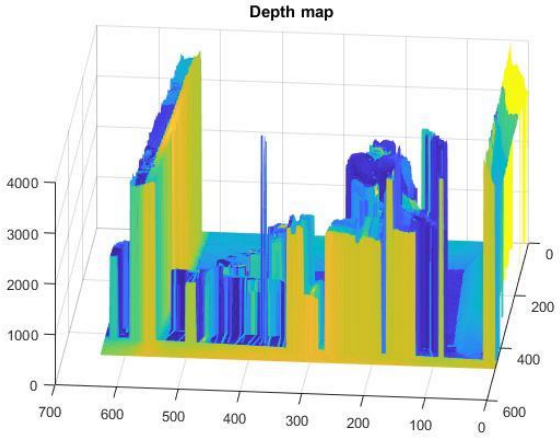
Disparity

Here the disparity map is shown for both image left and right which can be used for finding depth.



And below is shown the depth map for left and right image.

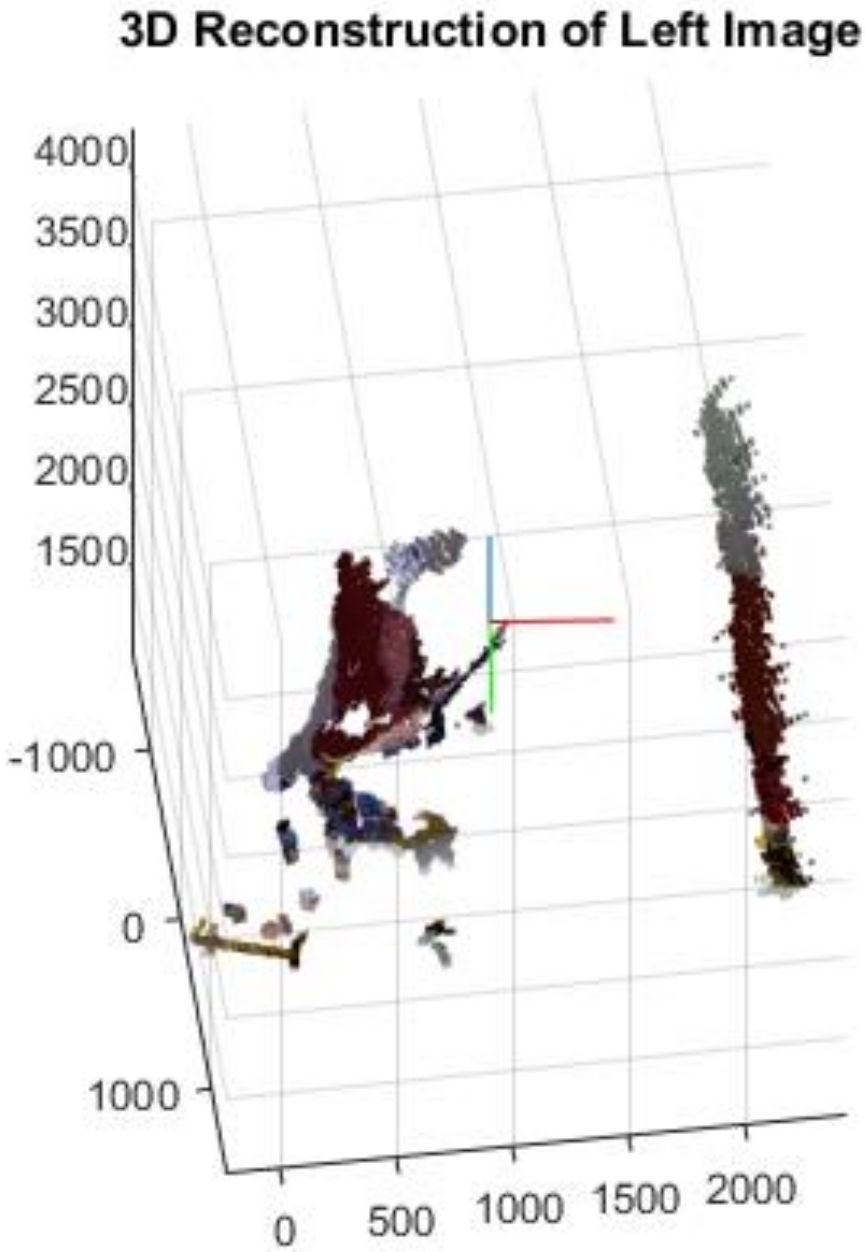


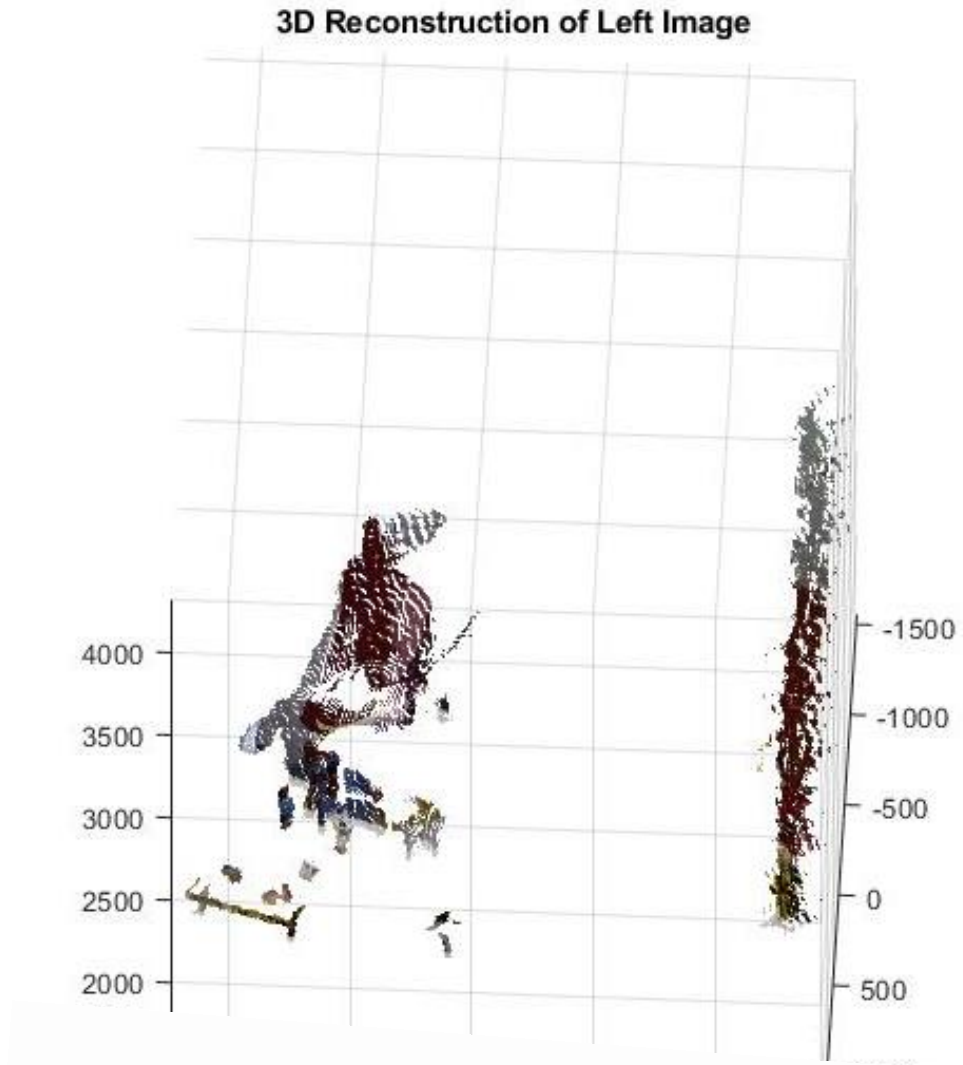


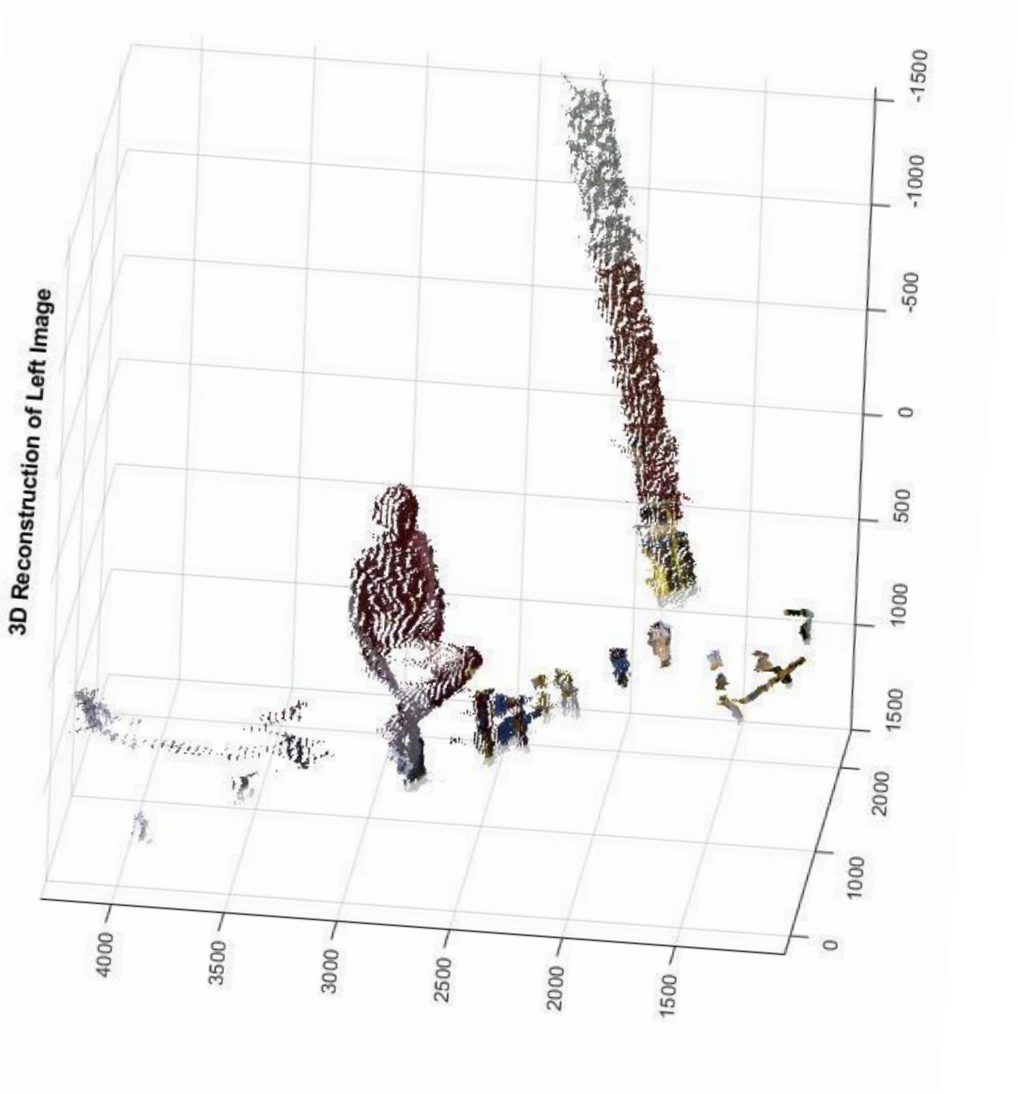
3D reconstruction Result

Here we show different views of right and left and final 3D reconstruction for our project.

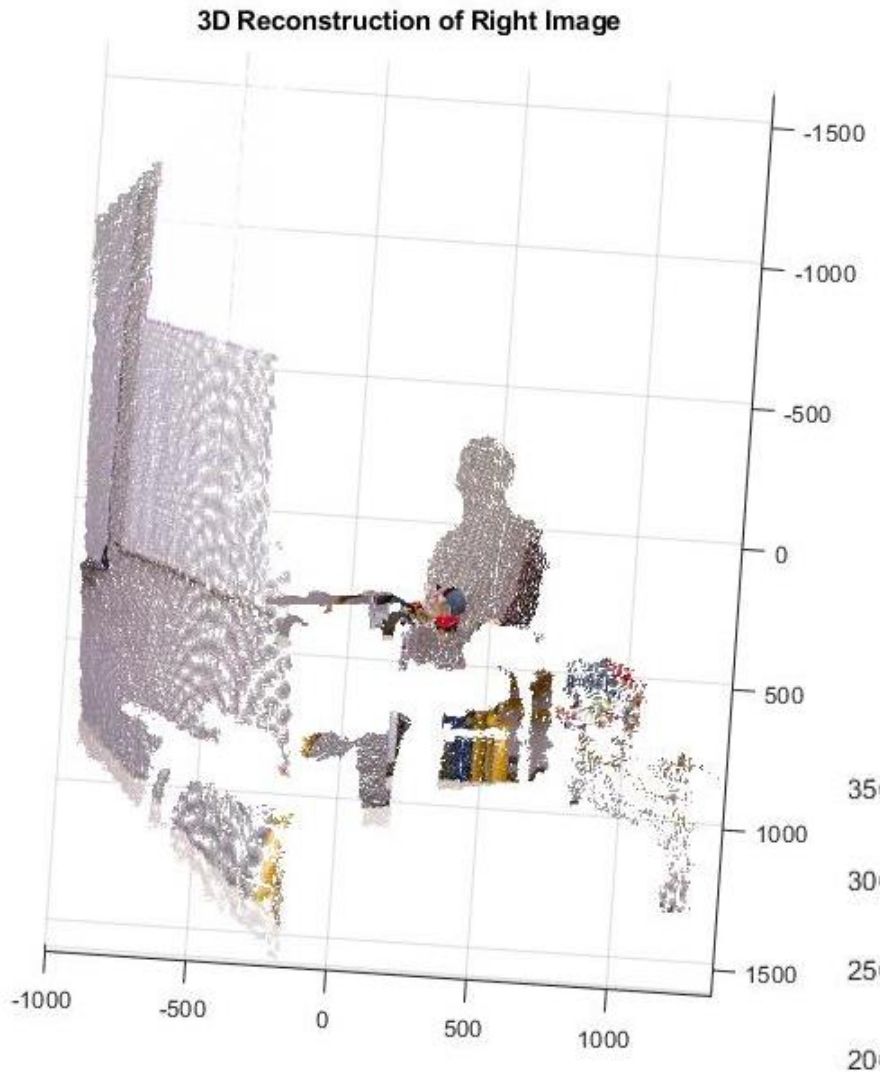
First, it is shown for left image:

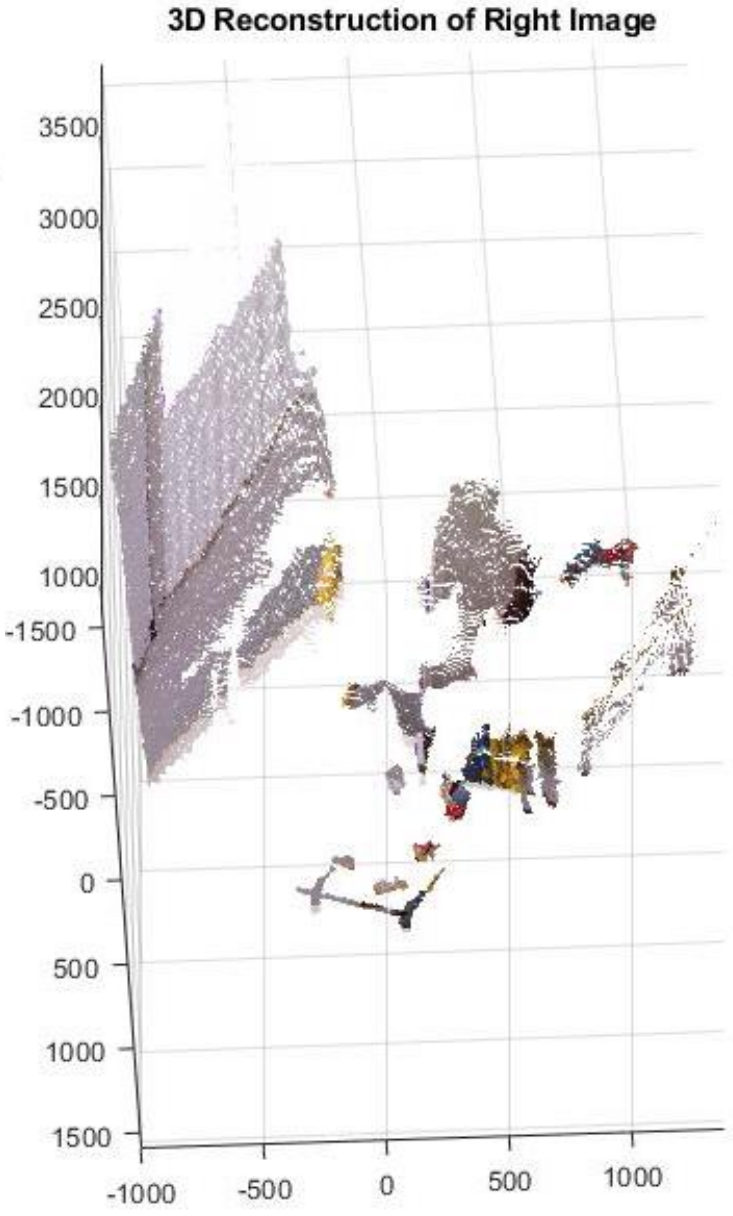


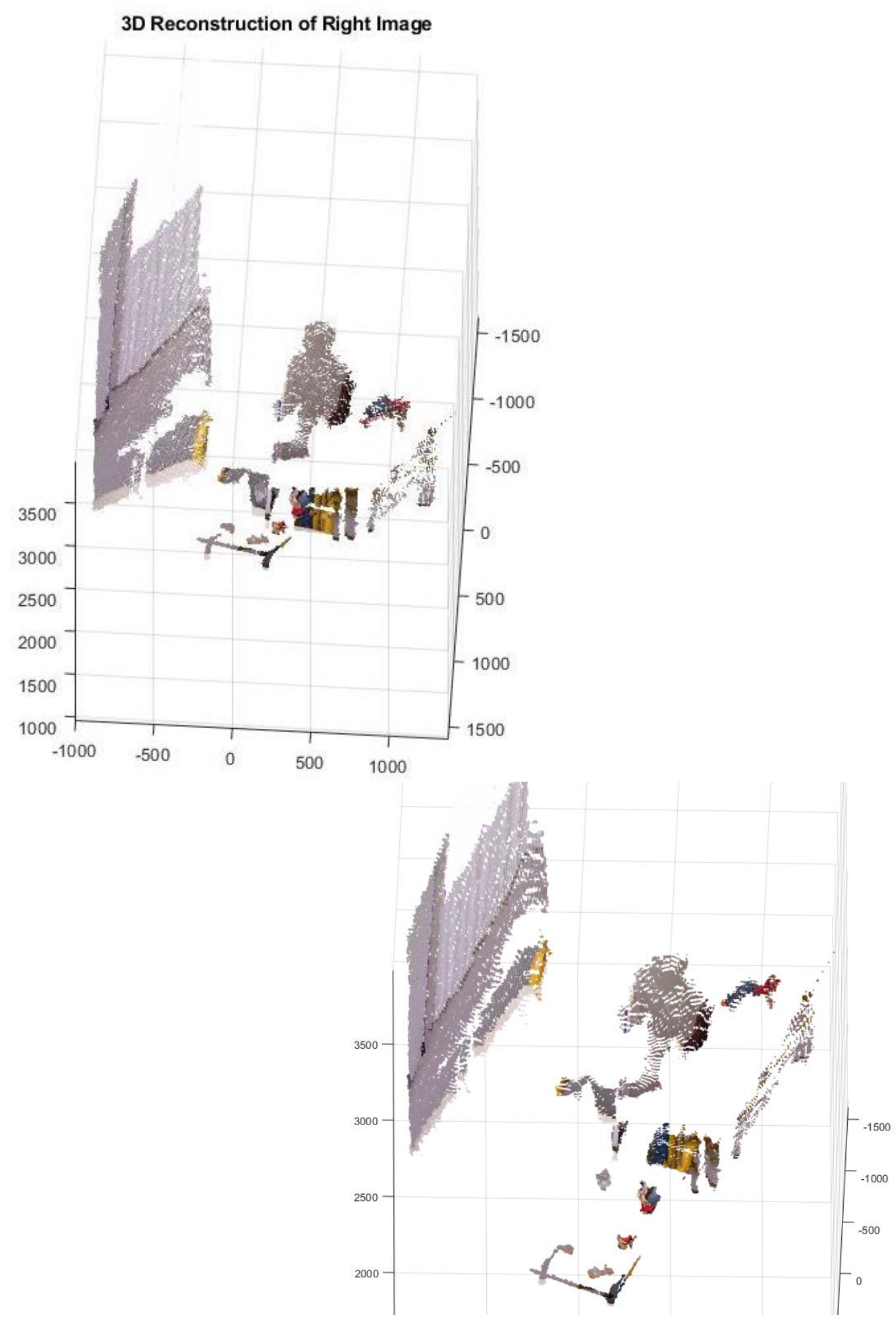

3D Reconstruction of Left Image
4000
3500
3000
2500
2000
-1500
-1000
-500
0
500


3D Reconstruction of Left Image
4000
3500
3000
2500
2000
1500
0
1000
2000
1500
1000
500
0
-500
-1000
-1500

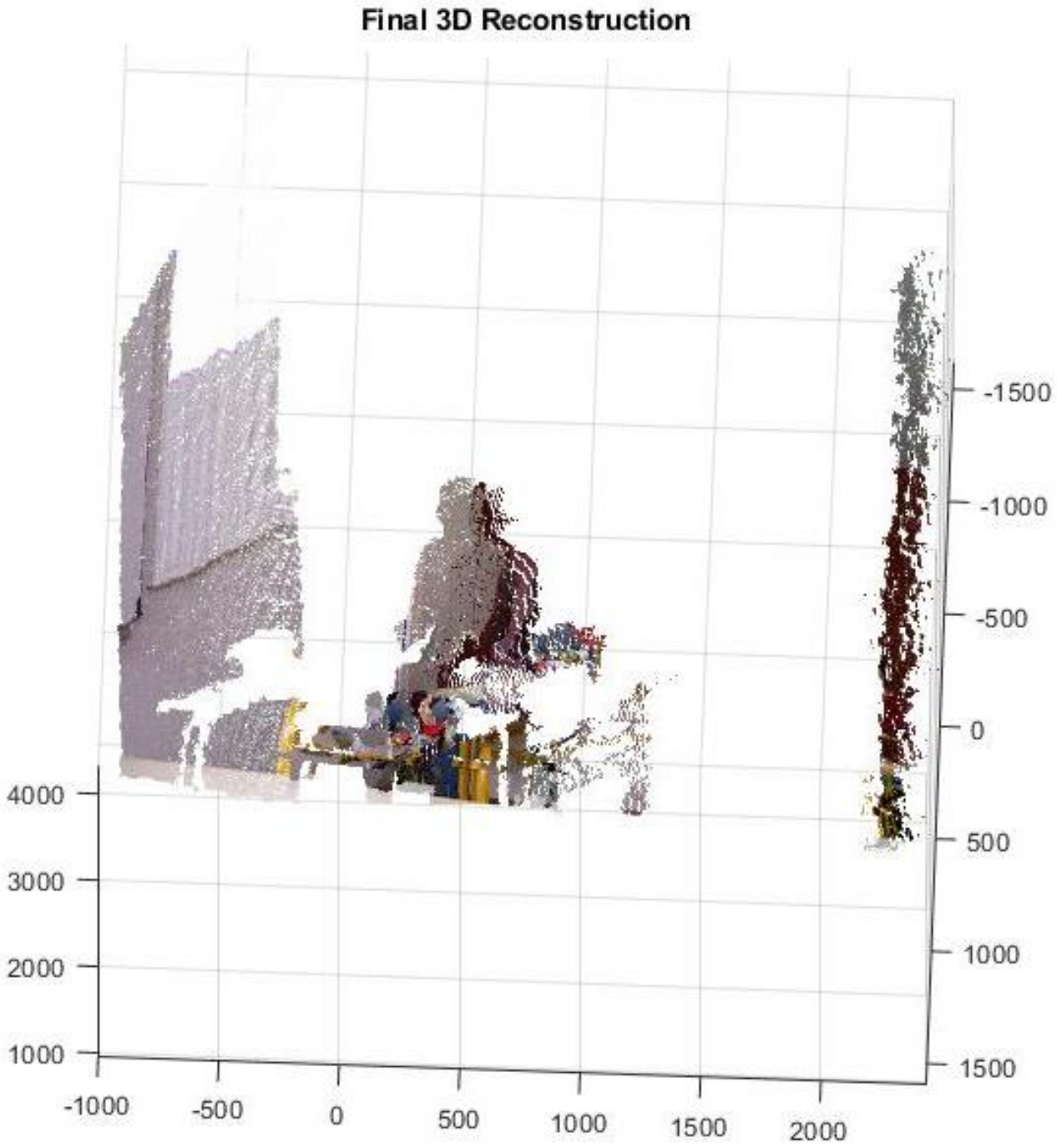
First, it is shown for right image:

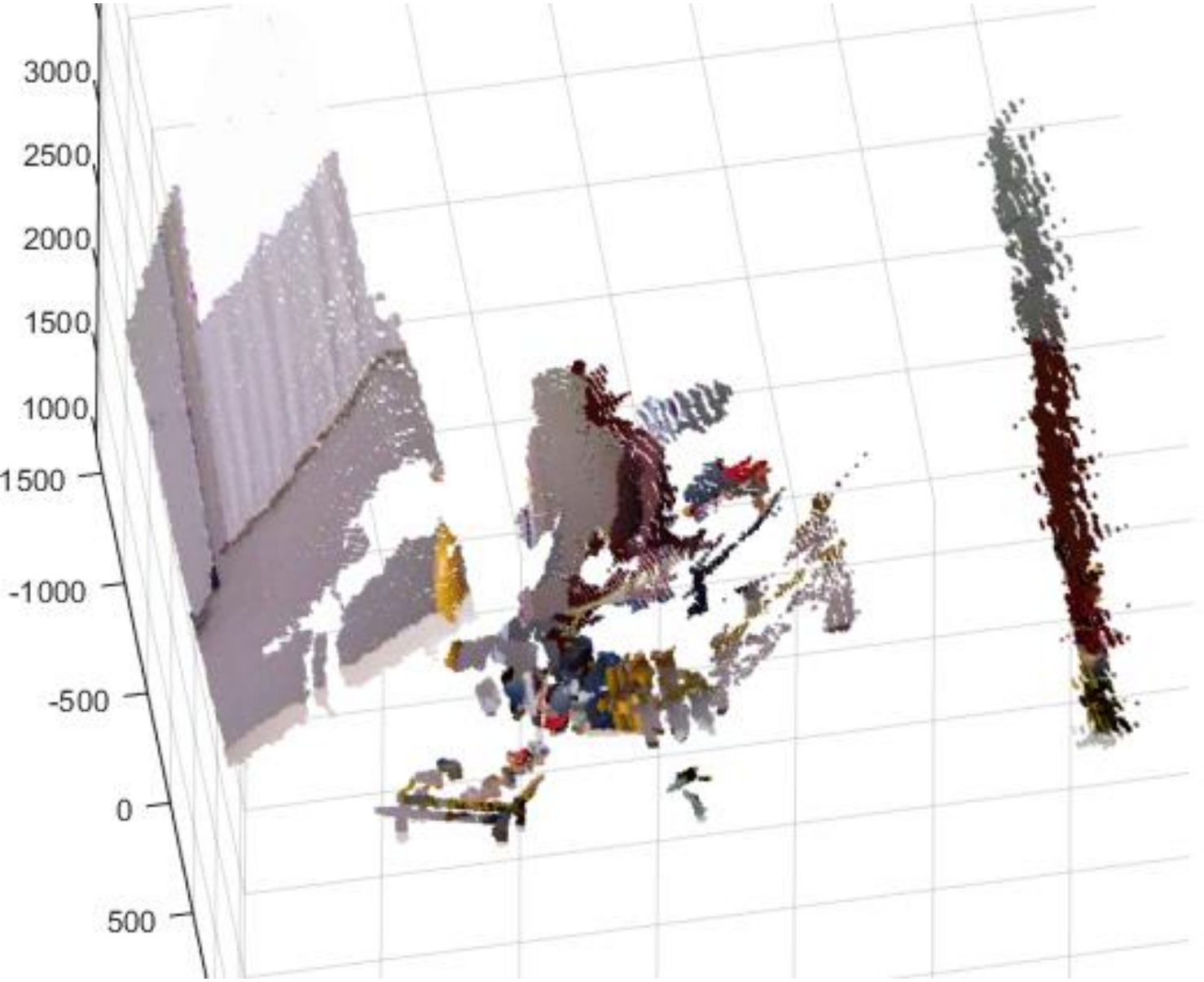


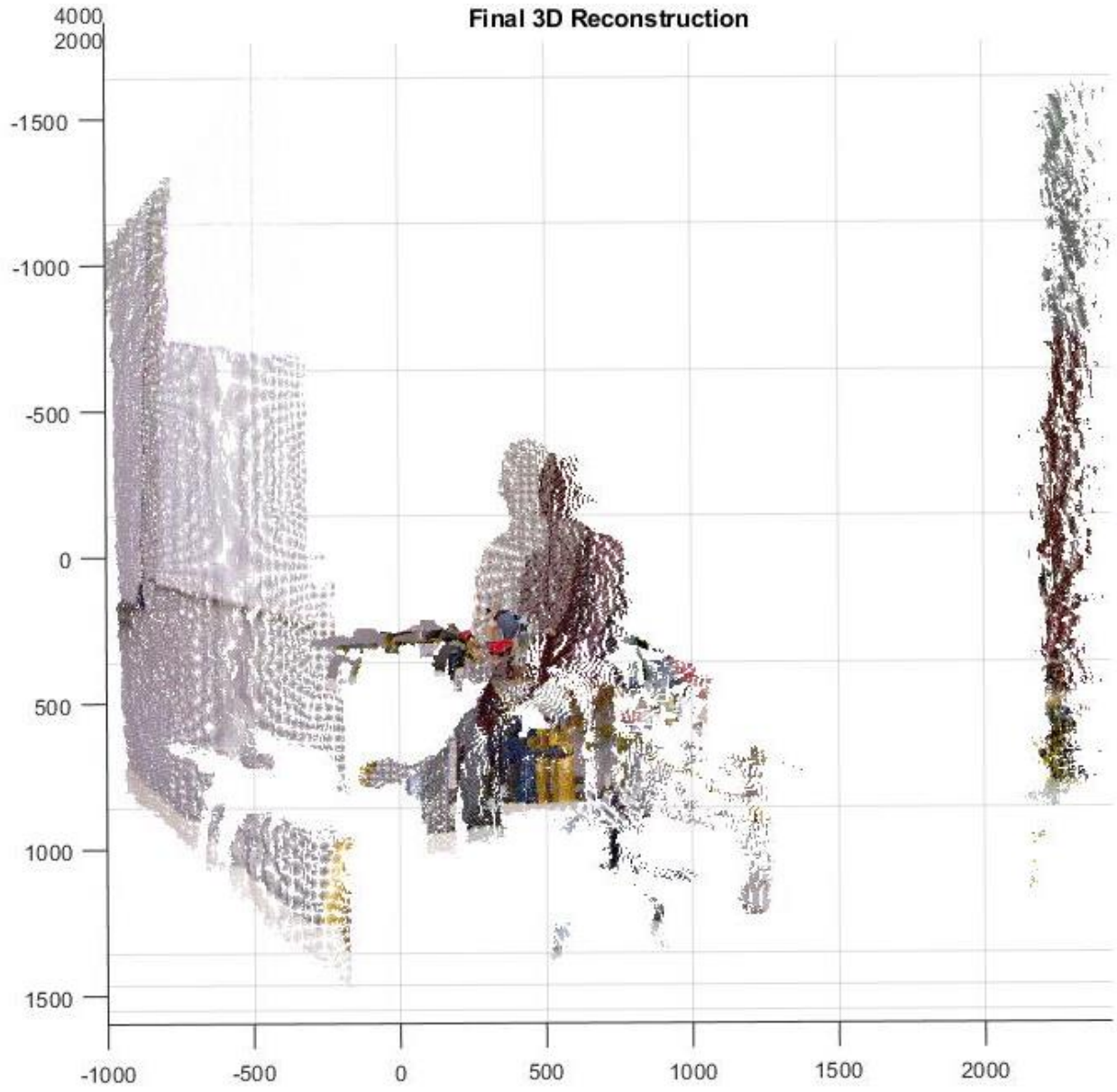


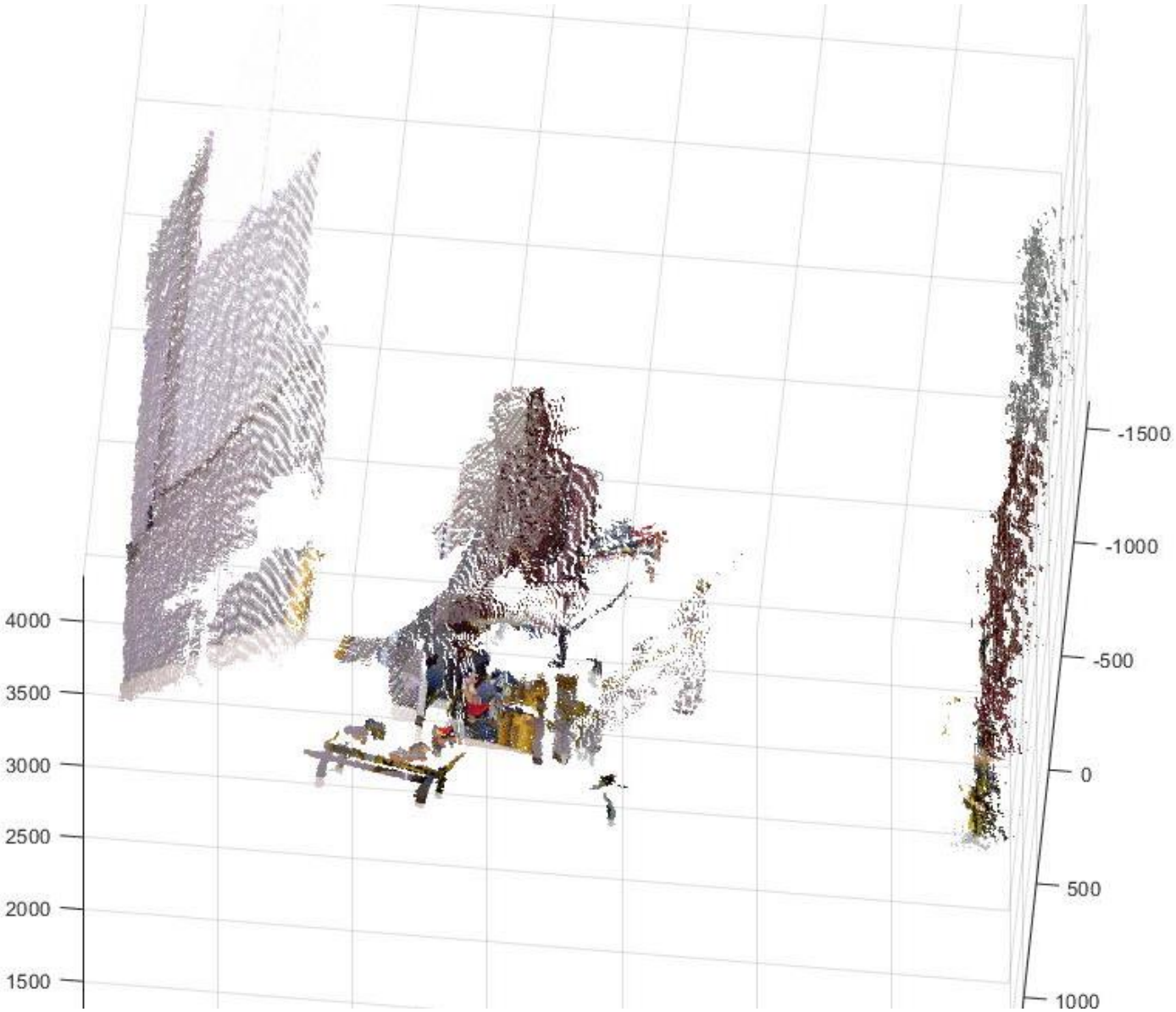

3D Reconstruction of Right Image
3500
3000
2500
2000
1500
1000
-1000
-500
0
500
1000
-1500
-1000
-500
0
500
1000
1500
3500
3000
2500
2000
-1500
-1000
-500
0

Final 3D reconstruction :




3000
2500
2000
1500
1000
1500
-1000
-500
0
500


Final 3D Reconstruction
4000
2000
-1500
-1000
-500
0
500
1000
1500
-1000
-500
0
500
1000
1500
2000

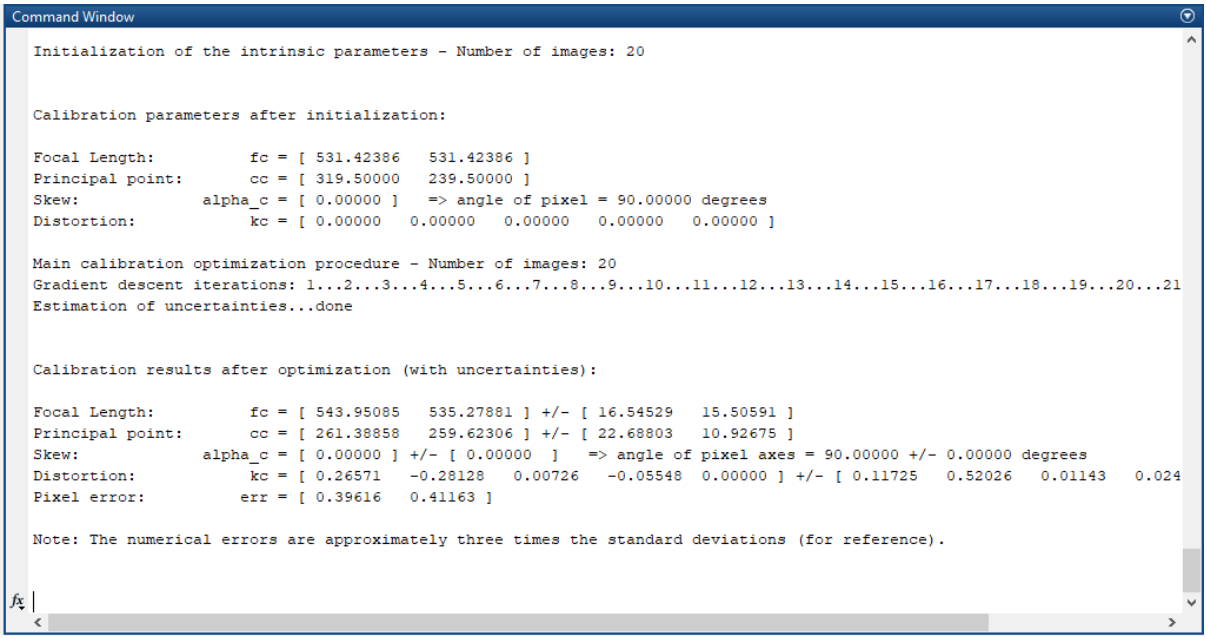


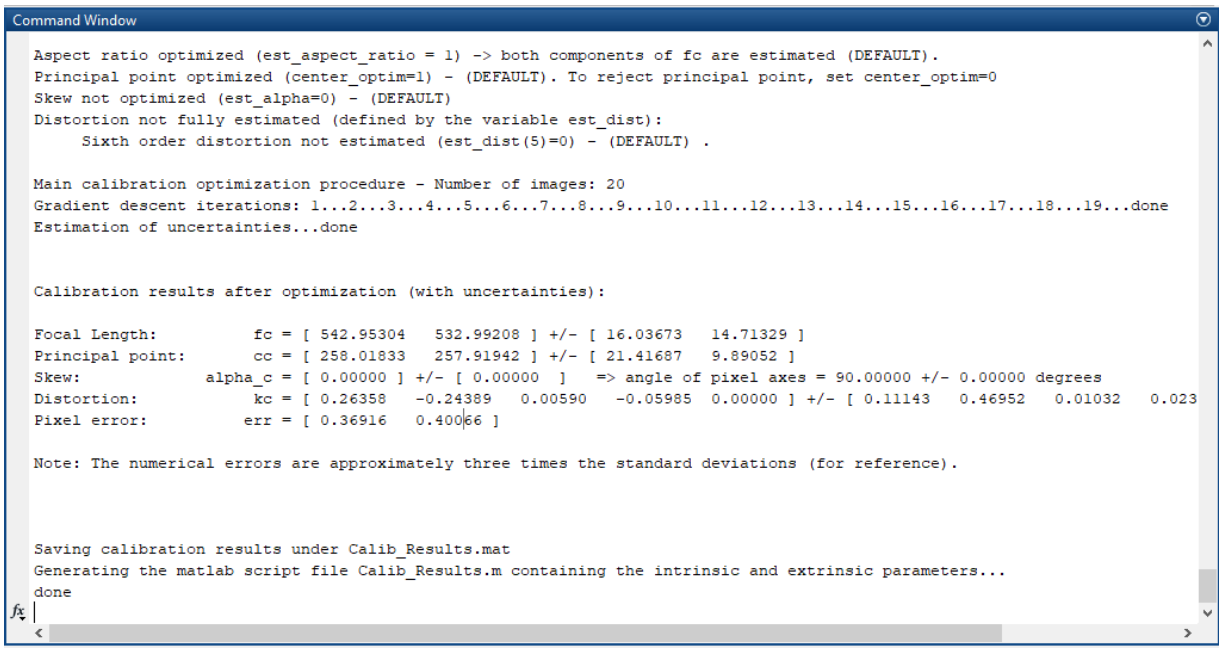
Conclusion

The reason that we could not get good result from calibration could be enivrement condition such that lightness and enviroment is sparse.

Appendix

The pixel error for 20 images before and after optimization (respectively before and after optimization).





References

[1] L. Zheng, G. Li and J. Sha, ""The survey of medical image 3D reconstruction"," 2007.

[2] "Kinect," [Online]. Available: https://en.wikipedia.org/wiki/Kinect#Kinect_for_Windows_v2_(2014).

[3] D. Ravì, "Kinect the next generation of motion control".

[4] "Camera Calibration Toolbox for Matlab," [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/.