CS 117 PROJECT FINAL PROJECT REPORT

Cem Emir Senyurt

06/10/2023

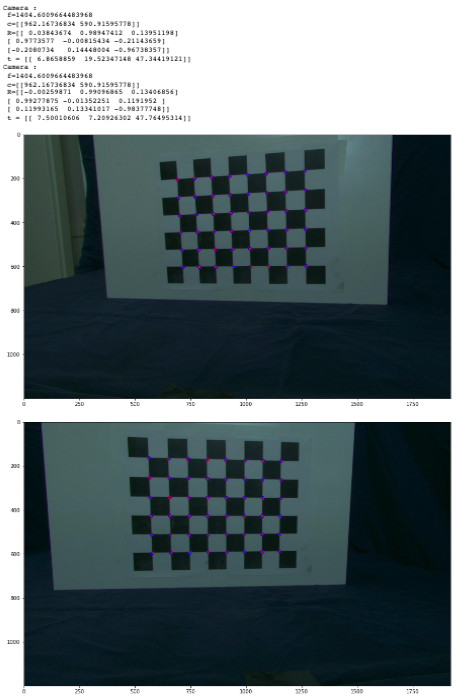
1. Project Overview:

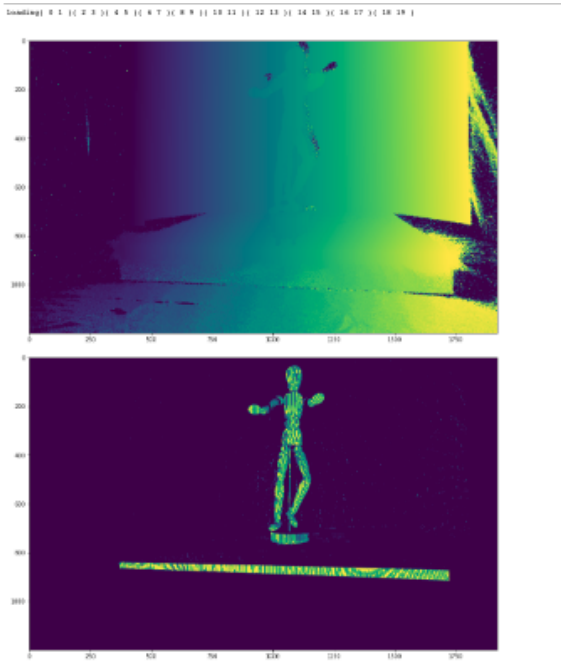
This project's topic was working on teapot scans to produce 3D objects. That was a fun and very exciting project in the beginning. However, alignment, and Poisson Reconstruction, was tough to deal with. The issues with matching the parts were occurring where the final model was not a whole 3D material. Camera calibration, reconstruction was very straightforward since we already have the previous assignments to look up which is very similar to what we do with this project overall. However, the majority of the time spent for this project was dealing with the meshing part. The meshing that has been used on Assignment #4 was not working properly here with either teapot or manny. So, appropriate changes have been made where it will be explained below. After, the meshes were imported to the Meshlab the rest was just alignment and Poisson Reconstruction. The final model of Manny looks very good and clear.

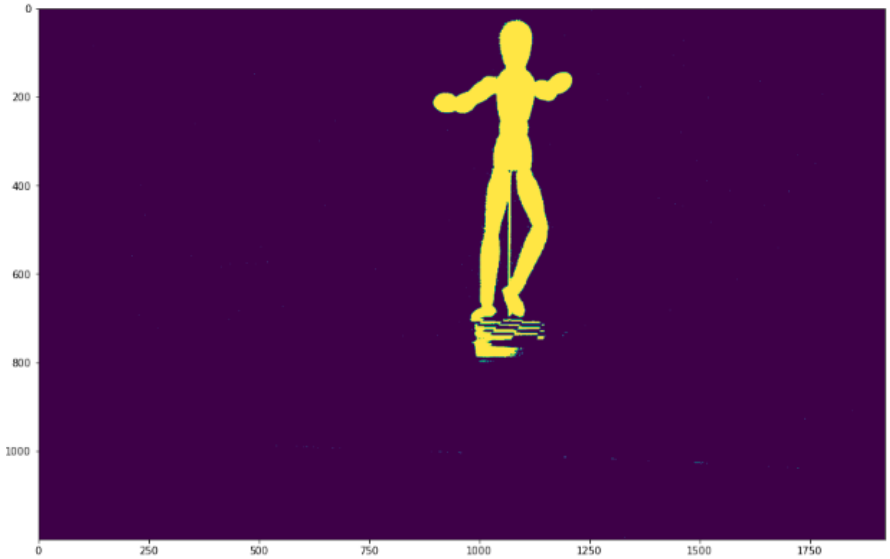
2. Data Model:

There are 4 files where each of those files contain images of Manny from different angles and with different lighting and filter projection. There are four color images, it was like this for each of the provided items and photo files, where two of them were from the left and two of them were from the right. One of the two from both sides are the background only and the other one is the background with the item, so we can use the decode function and distinguish our item, and get the appropriate mask for our item. That was a good way to see that the camera

calibration from the old assignment was working properly. What is happening here is that the calibration pickle is being used along with left and right frames to set the camera calibration to camL, and camR (as we did on Assignments). After I called the calibratePose function and passed the arguments, I got camera properties with chess board visualization.



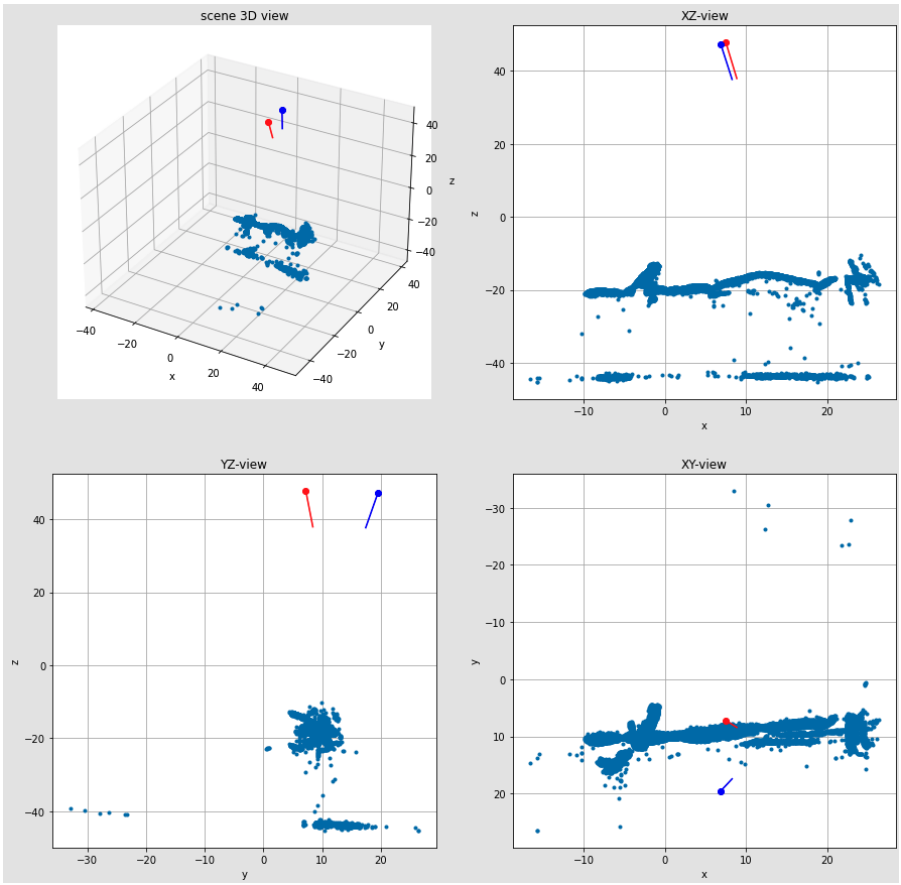




3. Algorithms

- Meshing:

  Meshing was one of the most important things that we have covered in this class. First thing I did was run the vis_scene function over the grab_1_u folder before I even created the mesh function. I did get the following screenshot:



Then, I got my mesh code from Assignment 4, tried to run in but I realized that there was something off. I couldn't inherit the code right away. The reason it wasn't working properly is that this project requires a very clean mesh since the outcome will become 3D ply files, and Poisson Reconstruction will be applied to it but homework 4 was a little bit more tolerant since the only thing the homework

was requiring was the output of the provided mesh view. So some changes have been made. First thing I did was implement my mesh code as a function. With that I could call that function every time when I try to create ply files for each folder (in a for loop) so I could also run vis_scene for them to see what's going on in different angles and views after the meshing since the before view has already been printed above.

The meshing function executes a better structured image of passed scans depending on the supplied 2D and 3D points, color data, imprefixL, and imprefixR. A bounding box established by boxlimits is initially used to confine the area of interest. By iterating over its columns and choosing only the points that are included within the given box limits, the pts3 array is trimmed. To match the pruned pts3 array, the corresponding items in pts2L, pts2R, and color are also filtered.

The points in pts2L are then subjected to a Delaunay triangulation using scipy.spatial to create a triangle mesh (tri = scipy.spatial.Delaunay(pts2L.T)). To make the triangulation better, a nested function named prune_triangles is defined inside of the meshing function. It examines each triangle in the mesh to see if its edge lengths are correct.

What prune_triangles, function that is defined inside of the meshing function, does is that each triangle in the mesh is examined to see if its edge lengths fall below a certain limit which is called trithresh. The triangle is kept if all three edges meet the threshold criterion; otherwise, it is thrown away.

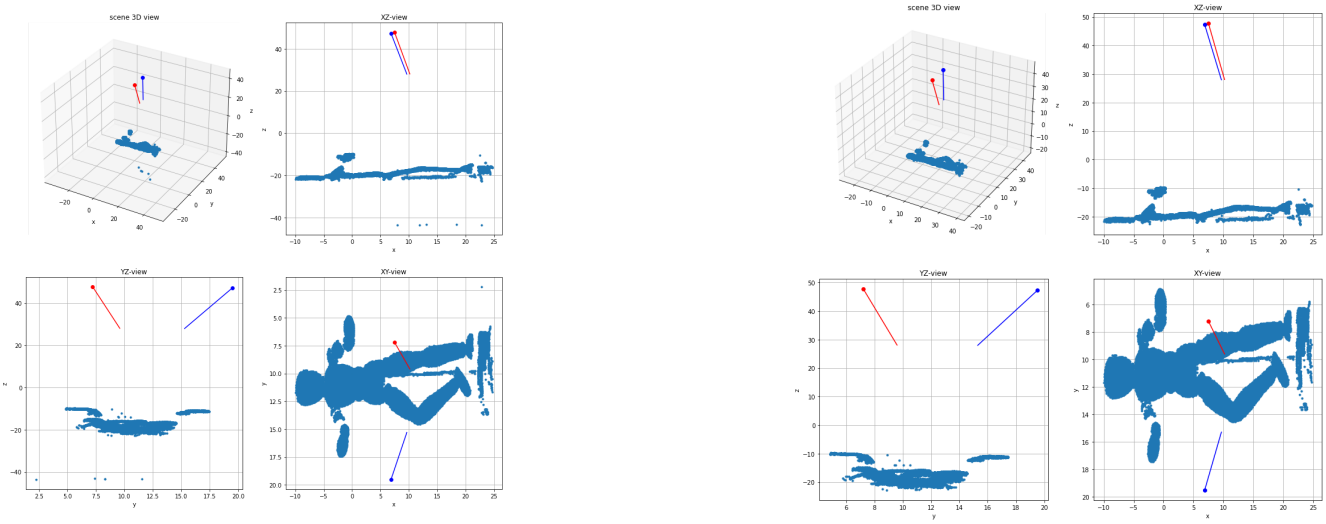
The tri object is updated with the set of triangles that have been pruned by using the prune_triangles function at the end. The function outputs the 3D points that have been pruned (pts3), the 2D points that correspond in the left and right views (pts2L and pts2R), the modified triangulation (tri), and the color information (color).

Furthermore, the next step is to run a loop five times (there are 4 scan files for Manny), and with each iteration, appropriate arguments have been passed to reconstruct to get the right pts2L, pts2R, pts3, color, color_mask.

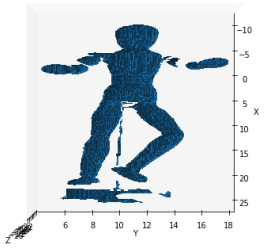
After we get the accurate variables from the reconstruct function, the meshing() function receives the reconstructed pts2L, pts2R, pts3, color, imprefixL, imprefixR. It meshes the manny to make it a better 3D ply.

Lastly, after the meshing part, vis_scene() shows how the meshed manny looks better in different points of view. The mesh is then saved as a .ply file by calling the writeply() function.

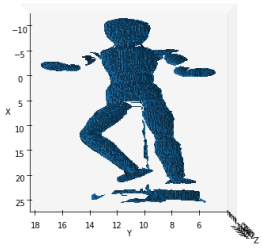
4. Results

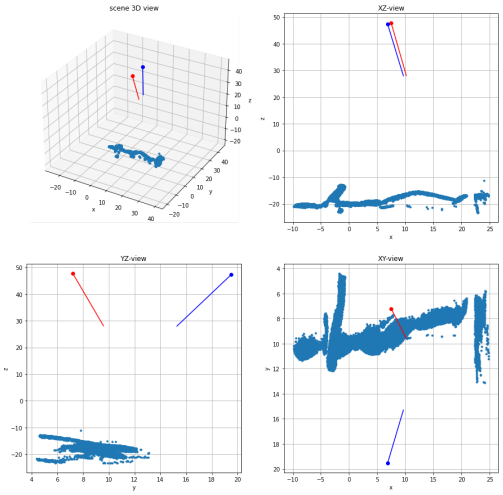


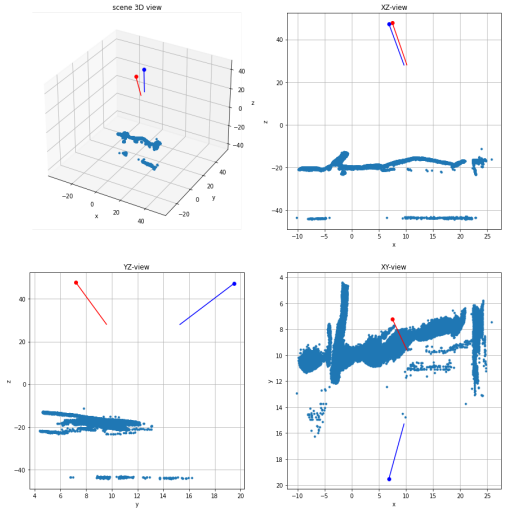
Final Mesh View 1



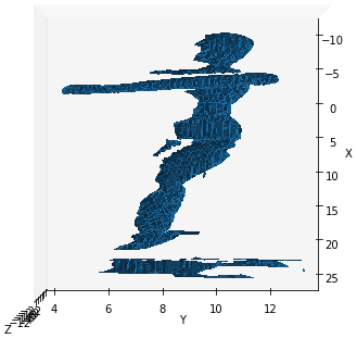
Final Mesh View 2



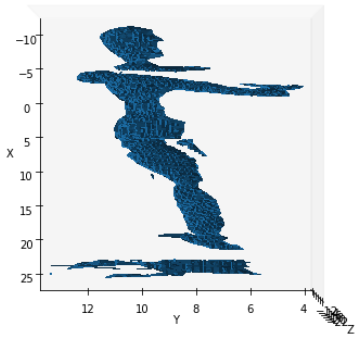


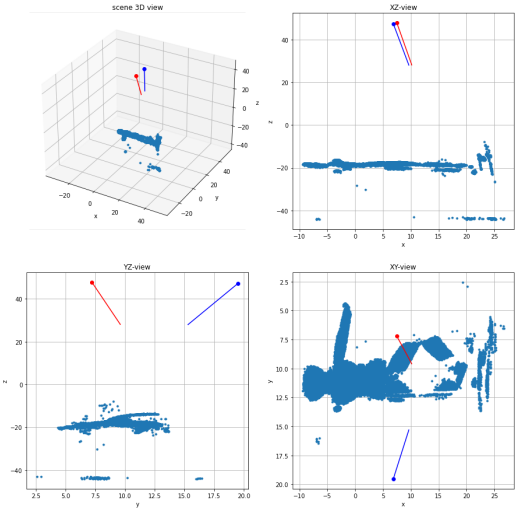


Final Mesh View 1

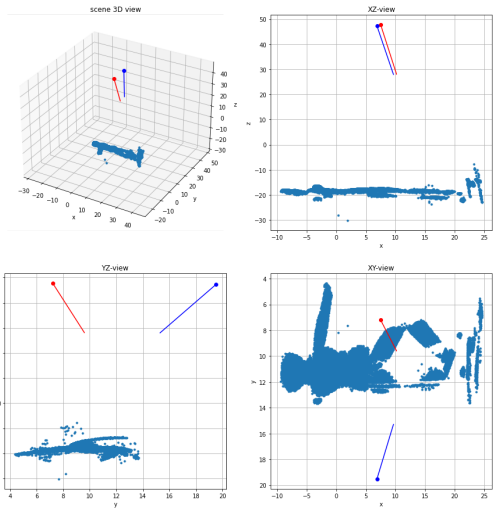


Final Mesh View 2

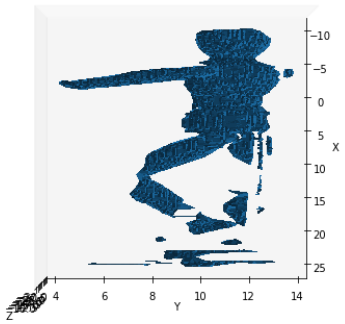


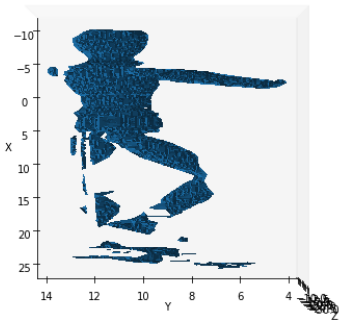


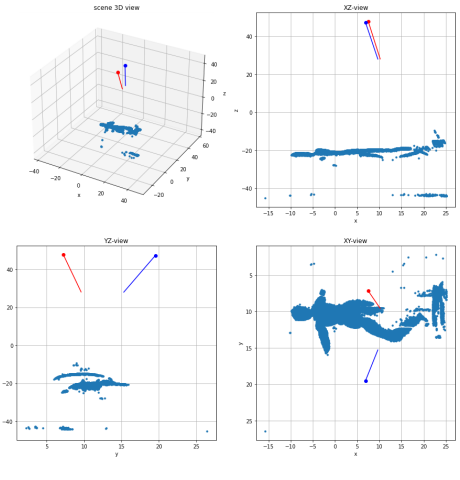
Final Mesh View 1

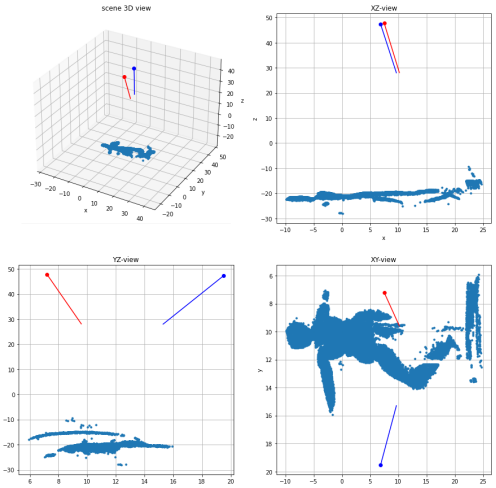


Final Mesh View 2

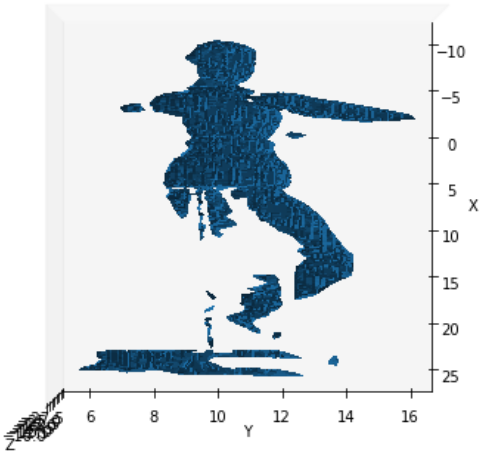




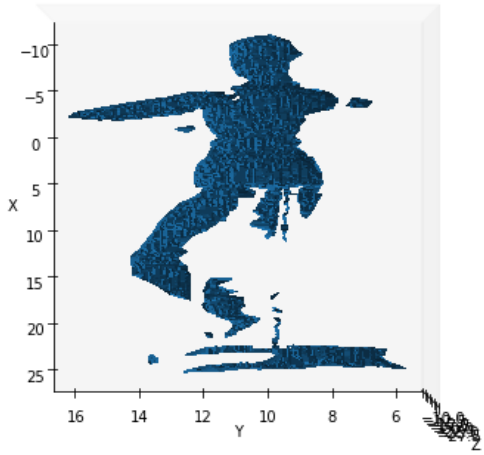


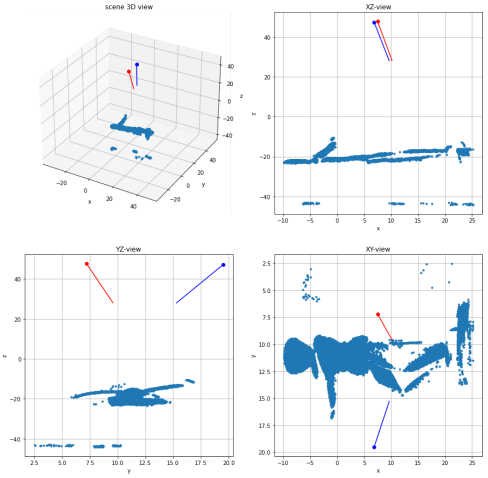


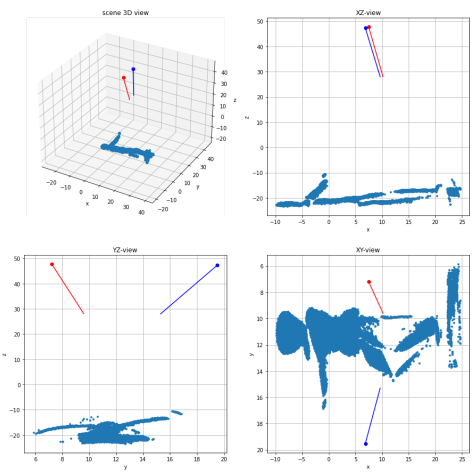
Final Mesh View 1



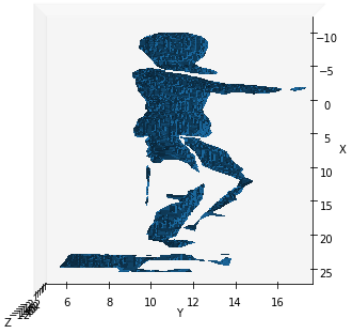
Final Mesh View 2



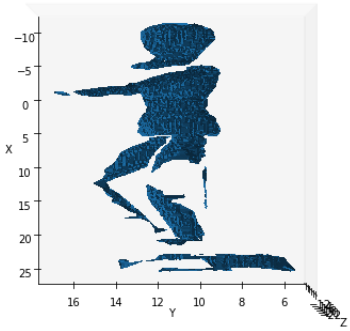


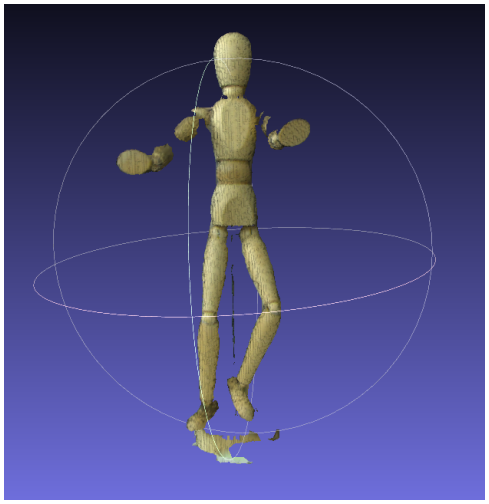


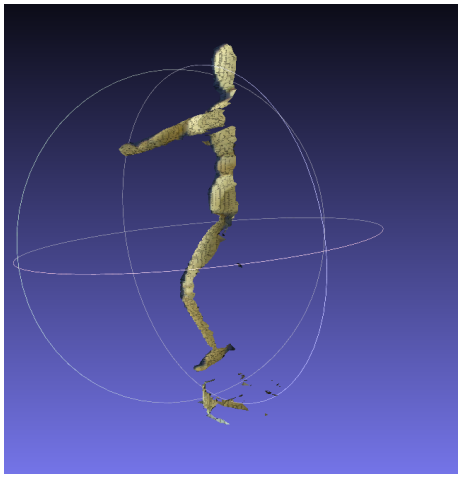
Final Mesh View 1

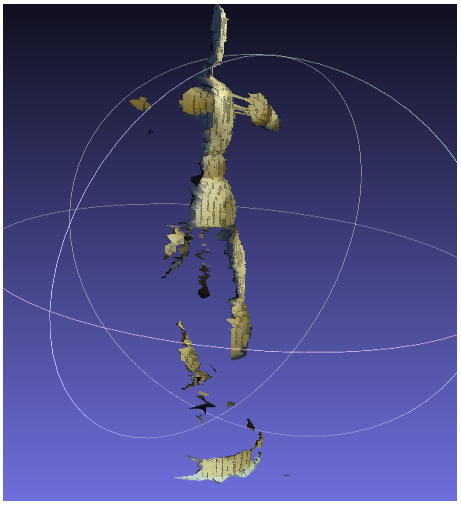


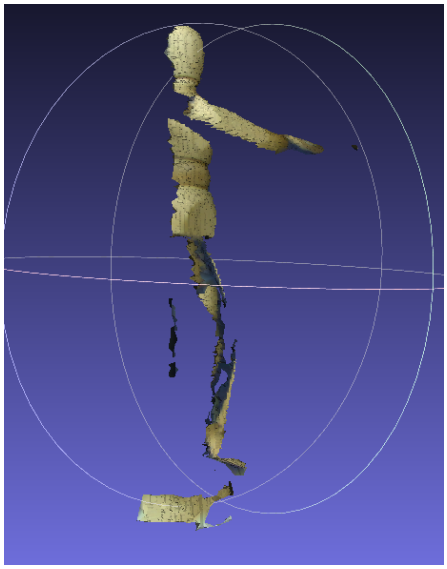
Final Mesh View 2











5. Assessment and Evaluation:

This project was very helpful and crucial to show our learned skills throughout this quarter. The task was straightforward because we are kinda familiar with parts of it as previous assignments, but as a whole it was messier and harder. It wasn't difficult to solve but took some time to play around with which I love so, I had no issues.

Meshlab's alignment and poisson gave me a hard time with the teapot, but trying on manny was a good idea. It seems like the problem was my alignment and matching meshes weren't that good before like it is now.

My weakness about this project was meshing. I think I spent 30 percent of the time on just that meshing function. I tried many thresholds and boxlimits, and came up with the best I could find.

Also, although I struggled a bit on MeshLab, working on real life stuff and playing around with 3D mesh was so much fun. Honestly, I wasn't expecting that. I might even change my career path towards computer vision.

6. Appendix:

calibrate.py was provided and no changes have been made. camutils.py was also a provided function, but there have been a few changes on it. First of all, "path" has been added to the function argument. Since scan's paths were used all around the function, and majority portions of those paths are the same, I passed a path name through the reconstruction function. Also, a calculation part was added. The change has been made to get color_mask, and adding the left and right mask to maskL, and maskR calculation. I just tried and saw that the mesh results were better. Also,the following explanations apply to both of the left and right side of the scans.

First, the foreground image is obtained by subtracting the background image from it, followed by replacing any negative pixel values with zeros. Then, a binary mask is created by setting pixels with values greater than 0.05 to True and pixels with values less than or equal to 0.05 to False. This thresholding step separates the object of interest from the background. So, better mask and reconstruct results have been obtained. Also, color_mask test is shown above with the photo. Furthermore, meshutils.py function has been provided to us. However, I changed the line: f.write('element face %d\n' % tri.shape[0]); Since this line was giving an error, so I realized that the way writeply function is used later on in code, it requires me to use that line like this, f.write('element face %d\n' % tri.simplices.shape[0]). With that, no error has been encountered. Lastly, selectpoints.py, and visutils.py functions have been provided to us, but no change has been made. Those are used as is.