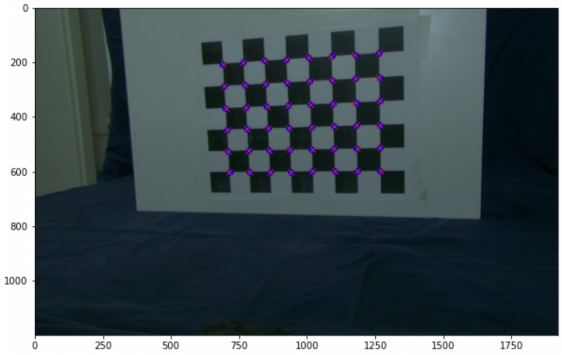
Mahsa Clinton
Final CS117 project report
Spring 2021

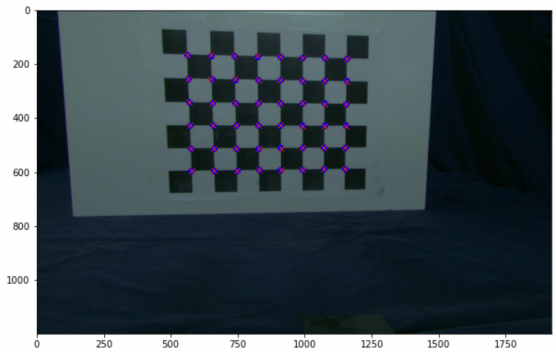
Project Overview:
The goal of my individuale final project was producing a high quality 3D reconstruction of an object from a collection of structured light scans. A set of scans from the provided google drive was utilized for calibration, reconstruction and creation of a set of meshes. Each mesh was smoothed using a simple smoothing algorithm, then was aligned using meshlab. The final model was reconstructed using poisson reconstruction.

Data:
To calculate the extrinsic and intrinsic parameters the calib_jpg_u file in the shared folder was used. The object selected for the 3D model was The Couple. I used all 7 different angles the couple was captured in. Folders grab_0_u to grab_6_u for final alignment. Each grab folder included 40 structured light scans and 4 colored pictures. These 4 pictures were used for removing background points and building the color pixel array.



```
Camera :
 f=1404.6009664593485
 c=[[962.16736834 590.91595778]]
 R=[[ 0.03843674  0.98947411  0.13951198]
 [ 0.9773577  -0.00815434 -0.2114366 ]
 [-0.20807341  0.14448005 -0.96738357]]
 t = [[ 6.86588564 19.5234718  47.34419111]]
```

Image1: Left camera parameters



```
Camera :
 f=1404.6009664593485
 c=[[962.16736834 590.91595778]]
 R=[[-0.00259871  0.99096865  0.13406856]
 [ 0.99277874 -0.01352252  0.11919527]
 [ 0.11993172  0.13341017 -0.98377747]]
 t = [[ 7.50010574  7.20925996 47.7649529 ]]
```

Image2: Right camera parameters

Algorithms:
Functions used for camera calibration, triangulation, decoding and reconstruction of the points are included in camutils.py. Functions used for pruning and smoothing and finally saving the mesh were added to meshutils.py. Both camutils and meshutils were provided by the instructor. The needed functions

were then added to these files respectively and in the order mentioned above. The Algorithms used for the project were completed and implemented in three different sections:

1. **Reconstruction**
   In this section of my code I utilized a reconstruct function that is a simple reconstruction based on triangulating matched pairs of points between two views which have been encoded with a 20bit gray code.

   First: I created a set of left and right masks for each angle that excluded the background from the object.This mask was computed by comparing the difference between object images and their backgrounds with a given threshold.

   Second: I computed a separate set of masks by decoding the gray code pattern with a given threshold value. At each angle the couple was captured with 40 structured light scans, 20 from the left camera view and 20 from the right. Each 20 light scan included the right scan and its inverse. The decoding algorithm took a light scan and it's inverse and compared their difference with a threshold value to make sure 0s and 1s are correctly detected, creating a set of masks for un-decodables bits . After that I converted the 10 bit code from gray code to BCD and then to decimal using this algorithm:

   $$\sum_{n=0}^{9} B[9-n] * 2^n$$

   Finally I constructed the combined 20 bit code using C = H + 1024*V for each view. I then multiplied all of my masks to this combination. After that, I used my combined 20 bit code to generate the pixel coordinates for the matched pixels in both views. After calculating my points on the left and right, I used a triangulate function to triangulate them into 3D points in the real world.

   Third: I Saved and returned the RGB values corresponding to each point. I did this by averaging the different channels on the left and right picture. I then built up a color pixel numpy array called 'cpix' with the appropriate color pixels for every coordinate in pts2. After that, I made sure this array was synchronized all throughout the pruning process.

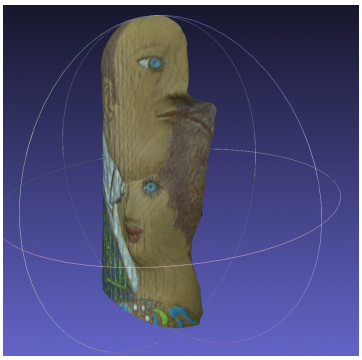


Image3: color pixels currently applied to grab_0 scan of The Couple

2. **Triangle pruning and box limit Pruning Implementation**
   I added a function to *meshutils.py* called *pruning*. This Function took the points from the left and right camera along with the color pixels and points for the object in the real world and pruned them.

```
def pruning(boxlimits, trithresh, pts2L, pts2R, pts3, cpix):
    """
    prunes the pts3 and cpix with the given trithresh and boxlimits.
```

Image3 - the pruning function parameters

The pruning function implemented two sets of pruning. First one used a set of box limits. After removing the points out of the limits provided, the Delancey triangulation function was used to triangulate the points in pts3 with the new set of points.
Given this new set of points and the trihtresh I then implemented triangle pruning for any edge larger than the *trithresh*.

$edges\_1 \ <- \ distance\ of\ the\ first\ two\ triangle\ vertices\ in\ all\ triangles$
$edges\_2 \ <- \ distance\ of\ the\ first\ and\ third\ triangle\ vertices\ in\ all\ triangles$
$edges\_2 \ <- \ distance\ of\ the\ second\ and\ third\ triangle\ vertices\ in\ all\ triangles$
$good\_triangle\_mask \ <- \ (edges\_1 < trithresh) \ * \ (edges\_2 < trithresh) \ * \ (edges\_3 < trit$

$newtri \ <- \ delete\ triangles\ that\ aren't\ included\ in\ the\ good\ triangle\ mask$
$out \ <- \ all\ the\ unique\ indexes\ in\ newtri\ that\ are\ being\ deleted$
$tokeep \ <- \ all\ the\ other\ indexes\ that\ are\ being\ kept$
$update\ pts3\ using\ tokeep$
$update\ cpix\ using\ tokeep$
$update\ tri\ with\ the\ triangles\ with\ new\ indexes\ using\ tokeep\ and\ appropriate\ mapping$

Pseudocode1 - triangle pruning

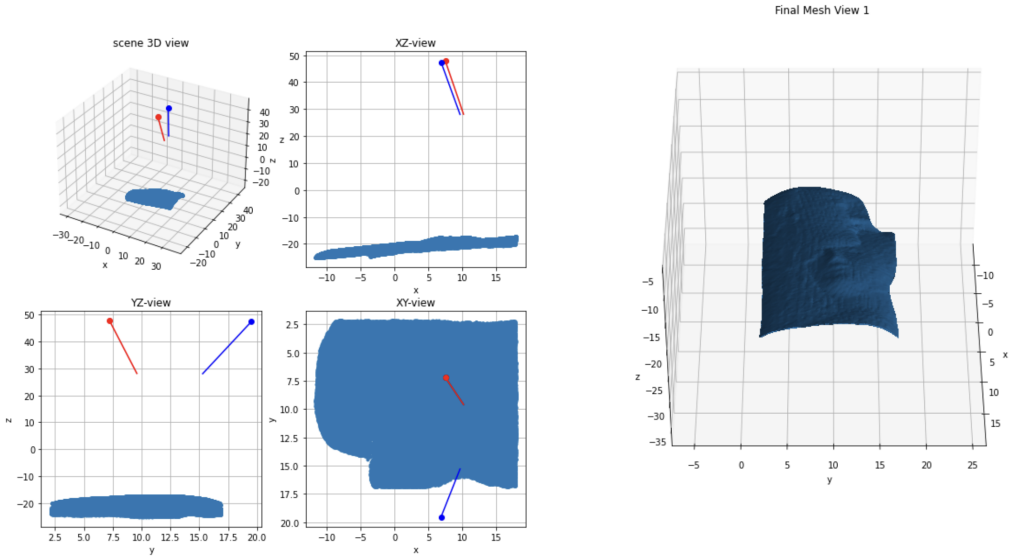


Image4: 3D scan view of grab_0. The visualizations that helped me pick the correct box limits

3. Smoothing function

I added a second function to meshutils.py the parameters and what it returned is shown below. Given *tri*, *pts3* and a *n,* the number of times to smooth, this function smoothes out the mesh by averaging out the points in regard to their triangulated neighbors.

```
def smoothIt(tri, pts3, n):
    """
    smoothes out the mesh

    Parameters
    ----------
    pts3 : 2D numpy.array (dtype=float)
        vertex coordinates shape (3,Nvert)

    tri : 2D numpy.array (dtype=float)
        triangular faces shape (Ntri,3)

    n: number of times we want to smooth

    Return
    ----------
    pts3 : 2D numpy.array (dtype=float)
        vertex coordinates shape (3,Nvert)
        smoothed n times
    """
```

Image4 - the smoothIt function definition

The smoothing algorithm implementation is shown in the pseudo code bellow:

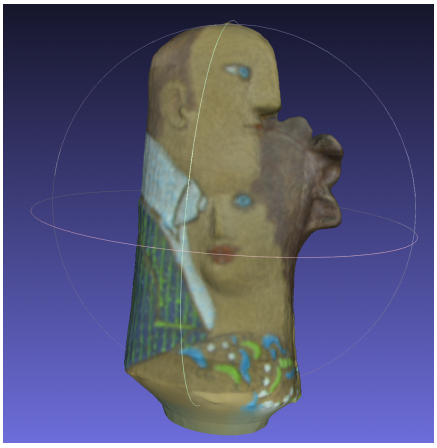
create a neighbor dictionary accessible with O(1) with a key for every pts3 index
    for every triangle in tri:
        for every index of the triangle add the other two
        as it's neighbors in the index key in the neighbors dictionary

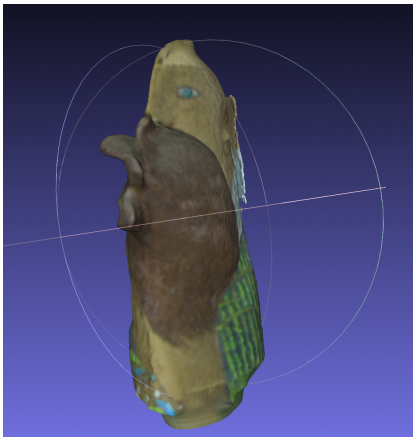
go through pts3, n times and replace each value by the average of its neighbors
    from the neighbors dictionary

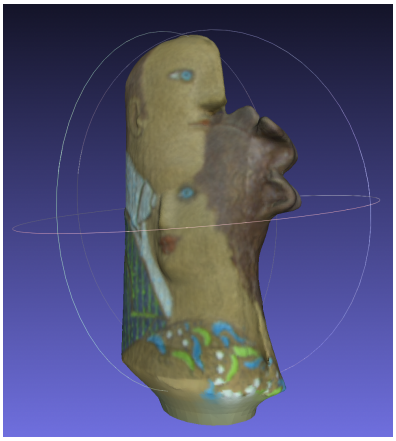
Pseudocode1 - smoothing

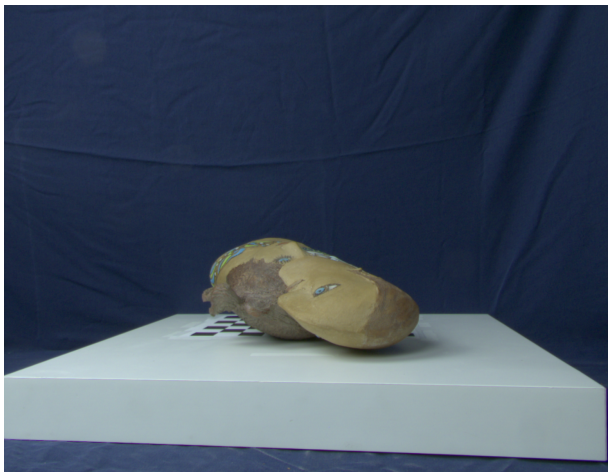
Image5 - mesh of grab_0 before and after smoothing 4 times

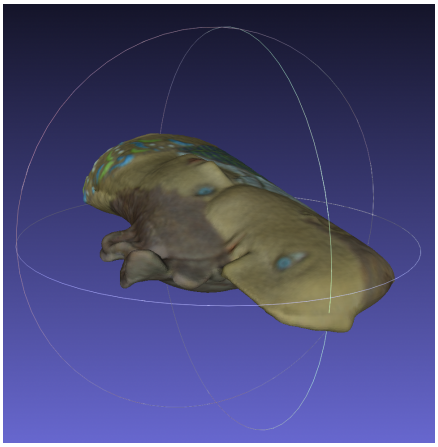
Results:
After the pruning and smoothing operations, I used all 7 sets of pictures provided for "The Couple" and used a function in *meshutils.py* called *writeply* to create ply files for each view. I performed the alignment and poisson reconstruction using meshlab. I aligned all 4 of the views using point based alignment and 2 of them manually. After that I flattened the model and ran poisson reconstruction on it. The result was a smoothed surface.

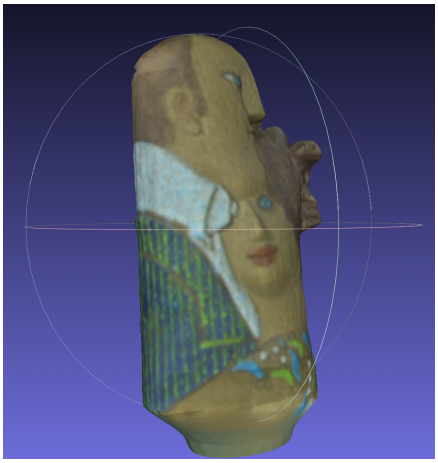


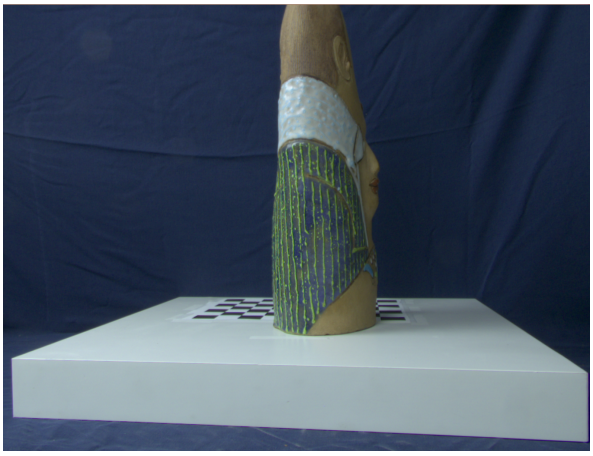


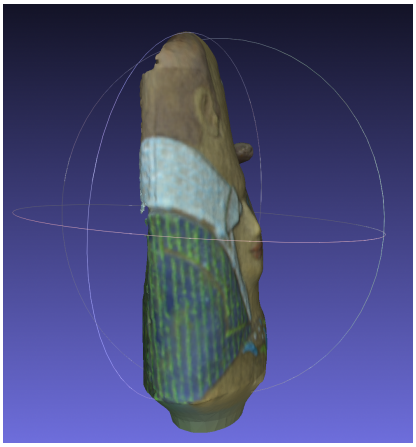












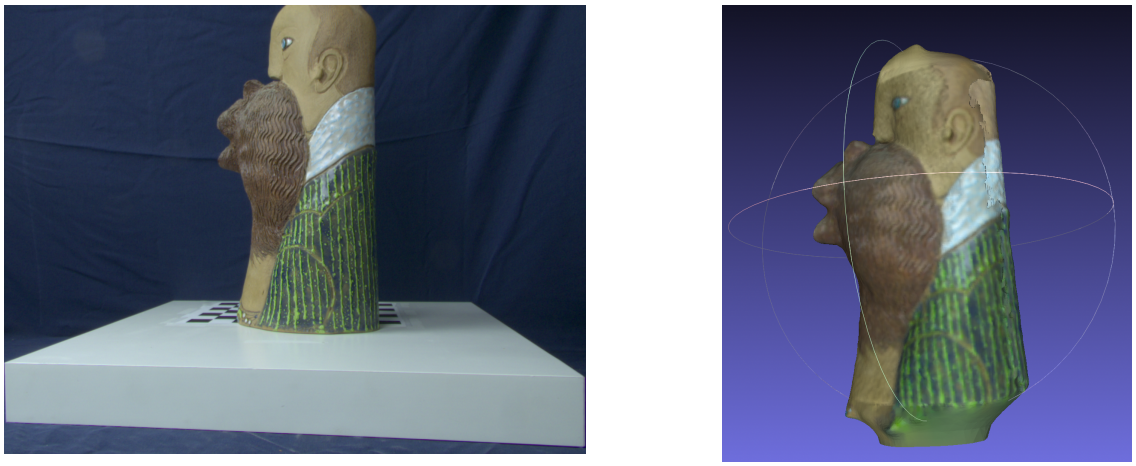


Image6: side by side comparisons image grab vs model

Assessment and Evaluation:
I set out to create a high quality 3D model. I was provided the material necessary to do so. Having completed the task I found the experience challenging but rewarding as well. Although I used a simple approach to smoothing and pruning it worked out well for my specific model and I am happy with the result. The one place I know I could have improved was my alignment. This issue is apparent in the pictures provided above. I was inexperienced with meshlab and the task of putting every view together proved to be time consuming and tedious. I had to start over with alignment multiple times because I could not find a way to undo my mistakes. Another possibility that could have improved my alignment was collecting more picture samples of different angles of The Couple. When performing point based alignment I found it difficult to find points in common. This could have been remedied with more samples. Although the model could have been improved, the process was exciting and being able to visualize the result of my hard work was very satisfying.

Appendix:
I used camutils.py and meshutils.py provided by the instructor and made adjustments to both. In camutils.py, I made adjustments to the reconstruct function in two different ways which were described in the report in detail. I added a couple of functions to meshutils.py. One for pruning and another for smoothing.