Mahsa Clinto
Instructor: Professor Charles Fawlkes

1.Setting up the final project [COMPLETED]:

I downloaded the files required for the project. I went through the pictures and codes to familiar myself with what needs to be done. I made some organizational decisions on how and where I'd want to add my functions. I worked with a few different scans to find what I preferred and finally picked the "Couple" for my final project.

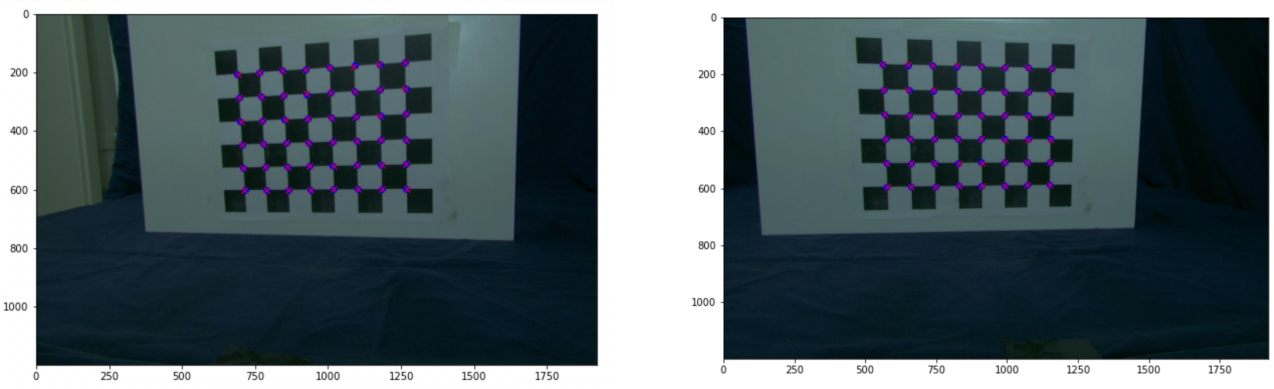
```
# load in the intrinsic camera parameters from 'calibration.pickle'
file = open('calibration.pickle','rb')
object_file = pickle.load(file)
fx,fy,cx,cy,dist  = object_file['fx'],object_file['fy'],object_file['cx'],object_file['cy'],object_file['dist']


```

2. Finding the extrinsic and intrinsic parameters[COMPLETED]:

I changed the directories in the calibrate.py method. Took me some time to realize what pictures to reference for decoding but eventually I was able to calculate my parameters as followed:

```
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]]
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 ]]
```



The checkerboard calibration pose pictures results

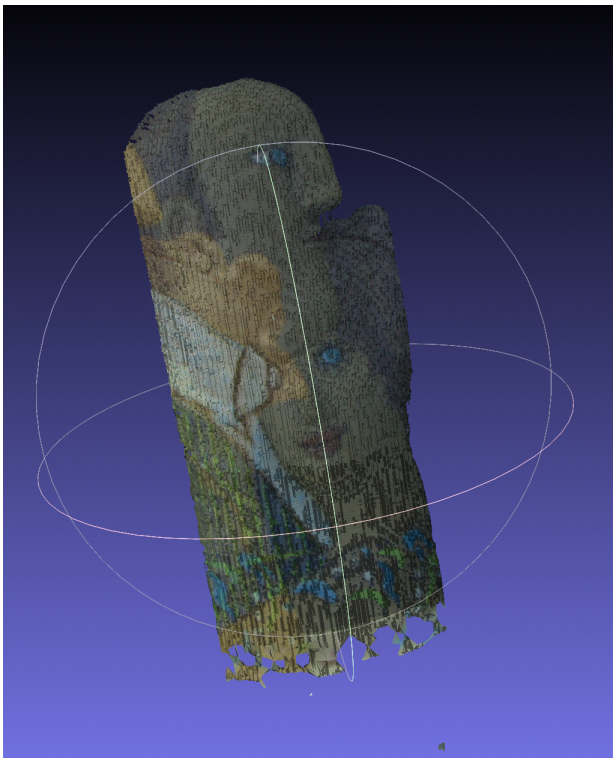
Mahsa Clinto
Instructor: Professor Charles Fawlkes

3. Modifying the reconstruct function[COMPLETED]:

I decided to use the camutile.py that was given for the project. I set out to modify the reconstruct function there to:

a) compute a mask that excluded the background from the color image and add it to the other masks on the left and right in the function.

b) Saving and returning the RGB values corresponding to the pts points. I did this by averaging the different channels on left and right. I then built up a color pixel numpy array called **cpix** with the appropriate color pixels for every coordinate in pts2. I then made sure this array was synchronized all throughout the pruning process.



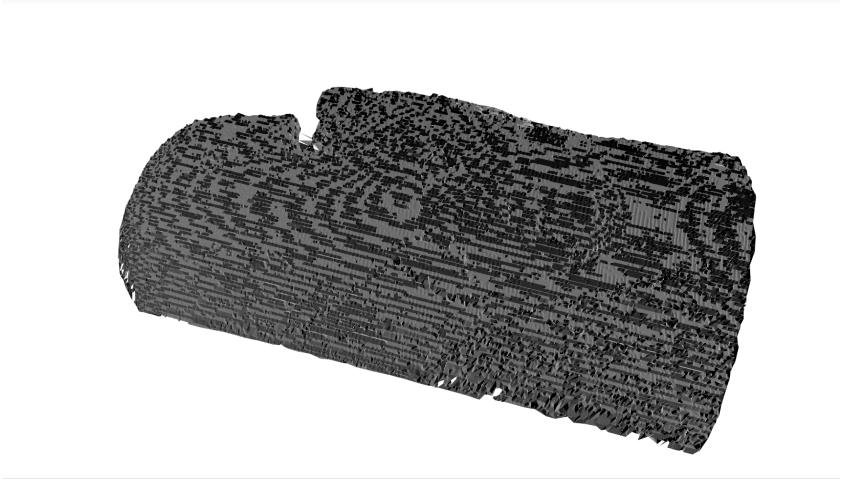
an image of a single view on Meshlab, colors nicely matching

4. Writing a mesh function and testing it[FINISHED]:

I wrote a function called pruning and added it to the meshutil.py. This function takes care of triangle pruning and removes the points outside the limit box.

```
In [5]:
boxlimits = np.array([-12,18,2,17,-30,-3])
trithresh = 0.8
tri, tris, pts3, cpix = meshutils.pruning(boxlimits, trithresh, pts2L, pts2R,pts3, cpix)
```

To begin with I realized I had a bug in my original pruning algorithm that took me several days to fix. I changed my algorithm many times and finally was able to cut the long triangles out successfully. I experimented with different shapes and decided on the Couple scans.



My current mesh of the “Couple”: the surface seems very edgy but hoping that smoothing will take care of it.

“Teapot” looks smooth but it wouldn’t triangulate correctly

5. Writing a smoothing function and testing it[IN PROGRESS]: Currently I’m working on my smoothing function. I’ve added it to my meshutil.py file and called it smoothing (picture shown below). My plan is to either use the tri.neighbors of the Delaunay triangulation or go through the list of tris once and save the neighbors and then access that list of neighbors to calculate averages that way. My issue is that I have already changed “tri” by deleting vertices in my triangle pruning step. If I can’t successfully integrate the correct neighbors from tri.neighbors I will be using the other method. I’m still trying to work

things out on paper before I can finish my final function. I'm planning on smoothing multiple times for every view.

```
def smoothing(tri, pts3):
    """
    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)

    Return
    ----------
    tri
    pts3

    """
```

6. Finishing up meshing[IN PROGRESS]:

After finishing my functions I will work with every single view separately to find the best trimesh and box limit values. I am already working on three of the views and have one ready. Picture of the one view I have completed is already posted above in the meshing section. This will be a time consuming process.

7. Alignment[NOT STARTED]:

I have already downloaded meshlab and worked with it a little bit. I found some online tutorials that I will be using before I start the process.

8. Poisson surface reconstruction[NOT STARTED]:

I have looked over the function I need to use and it seems to be straightforward at this point.

8. Renderings for your final writeup[NOT STARTED]:

At this point I'm hoping to visualize my rendering in Meshlab. If I have enough time in the end I will try out Maya or Blender.