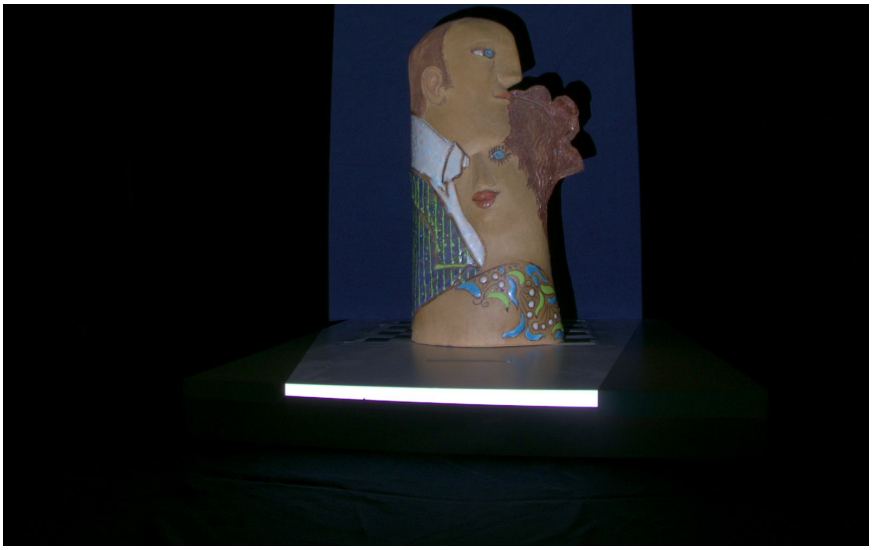
Final Report
CS 117

Jeremy Parnell / 27005248
Spring 2018
Default Project (Couple)

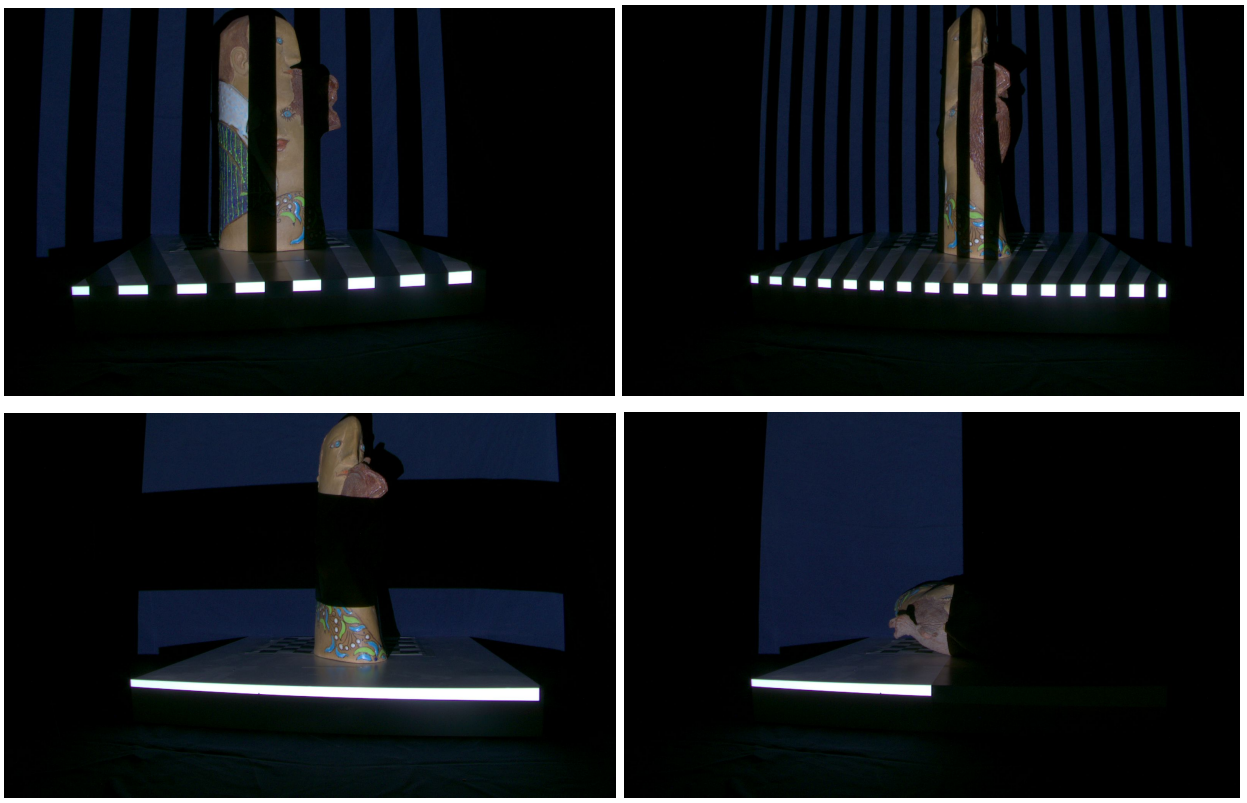
1. Project Overview



This quarter I set out on my final project with the clear goal of being able to create a high-quality 3D model of an object using the tools and lessons I've learned throughout this class as well as following the strict timeline I set out for myself. In order to achieve this goal, there were many problems I had to overcome; problems that included solving ways to successfully implement techniques for mesh cleanup and smoothing, properly allowing for clear and efficient mesh alignment, essentially learning how to combine meshes into a final model, and lastly, being able to render a model in order to allow for the addition of color/texture mapping. After these past three weeks of hard work, I finally believe I've come up with the results I desired and have achieved the goals I set at the beginning of it all. Here are the results:

2. Data Sets

Initially I had set out with the goal of scanning a personal object of mine (a Walle figurine) and had intended for it to be my 3D model. However upon further inspection of the scans that I had I gathered during my time with the TA, I realized that they weren't sufficient enough in order to build the high quality model I wanted, and so I opted to use one of the given set of scans (specifically the "couple" statue). The professor provided about 7 viewpoints (or "grabs") from around the statue, each viewpoint consisting of about 45 pictures/scans - adding up to about 315 photos/scans in total. For the scanning portion of the project, I used the given equipment provided by the teaching staff in the computer vision labs at UCI. All the camera parameters used throughout this project are saved in the camParam Matlab data file within the cameraParams variable.



3. Algorithms

Professor's implementation:

Triangulate (triangulate.m): Given points in a left and right image and the camera parameters of both the left and right camera, Triangulate will return a matrix of 3D points in world space

Decode (decode.m): Given a image path, a starting image, stopping image and a threshold, decode returns an array of decoded faules for 10-bit values and a binary image that indicates which pixels were decodable

Mesh_2_ply (mesh_2_ply.m): Given by a previous year's student, mesh_2_ply simply converts a given mesh to a ".ply" file in order to be used elsewhere. I used it for the purposes of Meshlab in my project

Nbr_smooth (nbr_smooth.m): Given a set of points, their triangulation, and a specified integer number of rounds, nbr_smooth smooths point locations. This was used in order to smooth my meshes in preparation for alignment.

Rigidaling (rigidalignment.m): Takes two sets of 3D coordinates, runs SVD on them, and return X2 with R and t applied

My implementation:

User Clicks Alignment (useralign.m): My function takes two image paths, two sets of 3D points as well as each of their corresponding sets of 2D points from their left cameras, and a max distance threshold integer. In return, useralign.m returns a copy of the second group of 3D points, however the difference is that the points are rotated and translated per the results of SVD. The way it works is like this: the user is prompted to click on at least 4 corresponding points in both images. My algorithm then takes those points and finds the nearest corresponding point in my set of 2D points for each image (xL). Once each point has found a corresponding 2D point in xL, the 2D points are used to index their 3D point counterparts. Once the 3D points have been found for both images, my function calls SVD on those points in order to solve for the best R and t to align the two meshes. Then the R and t are applied to the entirety of the second matrix of points and the result is returned by the function. I created this function in order to get the initial alignment for my meshes before running them through my ICP algorithm for fine tuning.

Pseudocode:

```
[points1,points2] = cpselect to get corresponding points
Find points in xL1 closest to points in points1
Find points in xL2 closest to points in points2
Find 3D counterparts from xL1 to X1
Find 3D counterparts from xL2 to X2
Use SVD on subset of 3D coordinates (rigidalignment.m)
Apply R and t to X2
Return X2
```

Iterative Closest Point (ICP.m): My ICP function takes two sets of 3D points, a max distance threshold, and an integer number of iterations. The way this function works is that for every point in X1 it computes the closest point in X2. Then it finds the subset of points in X1 whose nearest neighbor is closer than the specified max distance parameter. Then I gather those subset of indices as well as their corresponding nearest neighbors in X2, run the SVD method to find the R and t which maps X2 to X1 and apply those transformations to X2. I continue to repeat that process until convergence, or when the number of iterations specified by the parameter “iter” has been reached. This function refines the initial alignments and essentially prepares the meshes for Poisson Reconstruction

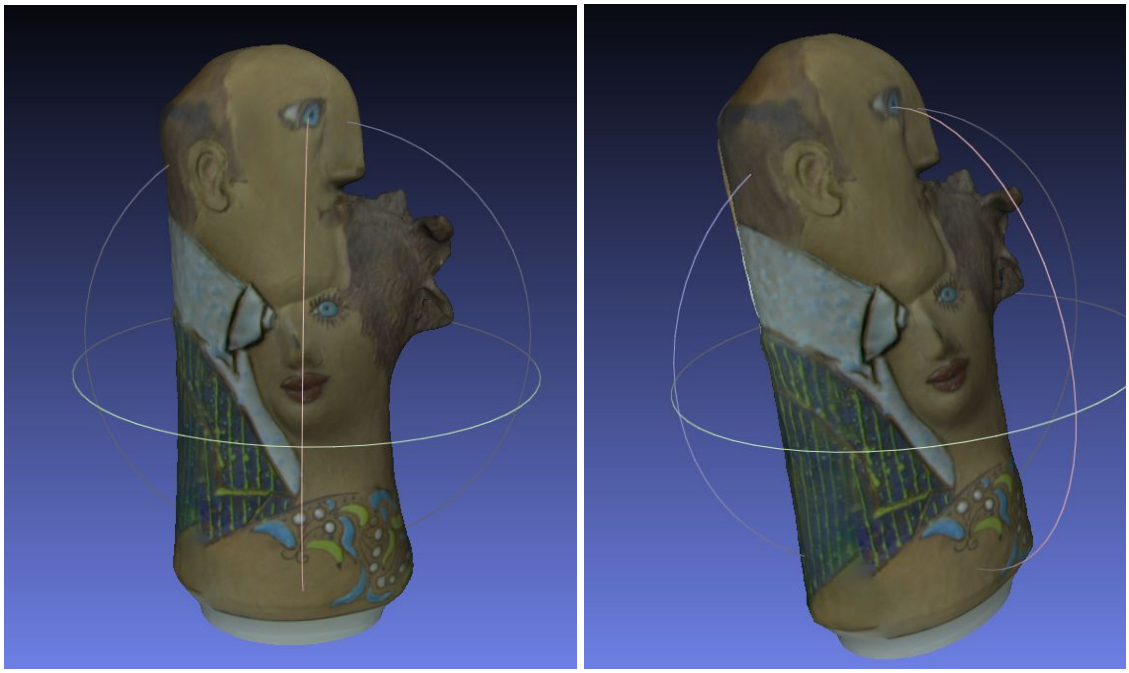
Pseudocode:

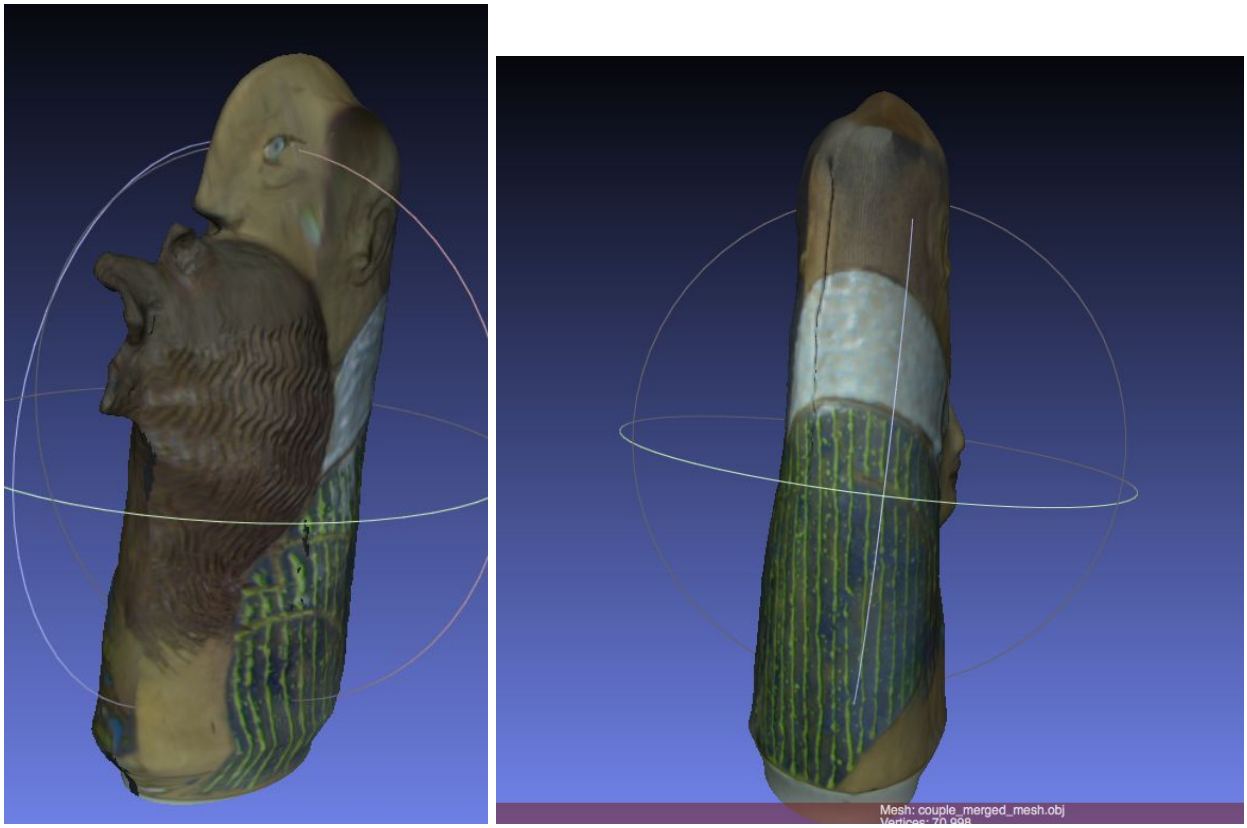
```
For each point in X1
        Compute closest point in X2
        Find subset of points whose dist < max_dist
        Find the nearest neighbors of the subset points in X2
```

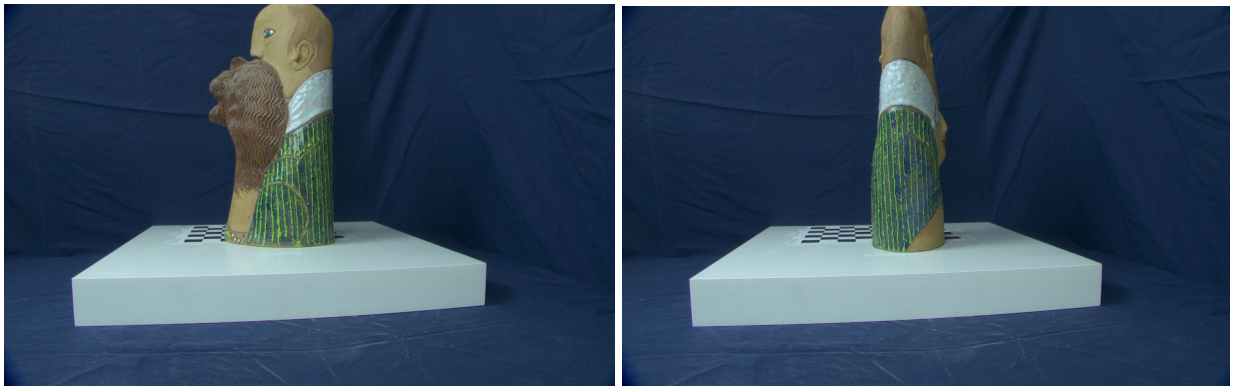
Use SVD to get R, t
X2 ← R * X2 + t
Repeat until convergence or until “iter” amount of loops has been reached

Demo.m (demo.m): Demo.m is the script I wrote that essentially demonstrates everything I’ve mentioned above: Scanning of the object, the mesh cleanup, the mesh smoothing, the mesh alignment, and the exporting of that data for other software use. You run Demo in order to build a high-quality 3D scan of an object.

4. Results







5. Assessment and Evaluation

After having completed this project, if I'm being completely honest, I do believe that this task is easy for a computer algorithm to solve. I feel like the only way to achieve these kinds of results is through computer algorithms. There is highly intensive math occurring all throughout the pipeline of this project. Not only that, but it's occurring over hundreds of thousands of points in matters of minutes or even seconds. The speed and relative ease that this problem can be solved in makes it easy to conclude that this task is made easier by computer algorithms. A task like this would take normal humans ages to solve. The biggest limitation of this project for me was time. A lot of the steps felt rushed for me because I always knew I had more to do and not that much time to do it in, so experimentation was really hard to fit in. But I do consider the time that I DID have to

experiment - a success. I thoroughly enjoyed trying to figure out the mesh alignment portion of this project myself instead of relying on pre-made functions to get the job done. If given more time I would've loved to continue experimenting with things like different smoothing algorithms, hole filling methods, mesh cleaning techniques, better texture mapping, and more. I believe more experimentation would've led to better results overall.

In general, there were a lot of limitations of the data - especially if you got stuck using the default scans. Not having the ability to control the scans or redo them made things a little harder. I definitely felt myself wishing I had been able to go back and redo my scans or redo the scans the professor posted just because some parts didn't scan as well as they needed to be, some pieces felt left out, or I just needed more scans in general (like for more angles). I don't believe, though, that the data limited me as much as the code did. It took a while for some portions of the code to complete (more specifically the ones I coded myself), and with every minute lost, especially in these last few days, it took away from the opportunity to go back and have more chances to refine and adjust. Overall, if given more time and if I coded up some more portions of the project more efficiently, I could've made up for a lot of lost time. As far as the project goes, I do believe that the weakest link for me was the alignment (even though I was proud to say I programmed it myself). My alignments are pretty good, but they're nowhere near pristine or perfect which I believe affects the whole reconstruction aspect and the texture mapping. It's easy to see the breaks in my final mesh, especially if you look at the man's back on the statue. I also wish I had better texture mapping. Some of the textures don't line up perfectly and there are hints of overlapping in the textures themselves. Overall I'm proud of my results and am happy to see what *this* kind of technology can accomplish, however I do agree that there are definite improvements that could be made throughout my project.

6. Appendix

Demo.m
Partly modified/taken from professors code from assignment 4 (scanning, mesh cleaning, smoothing)
Partly written from scratch by myself (alignment portion)

Rigidalignment.m
Borrowed from Slide 11

Useralign.m
Written from scratch but got idea from slides and help from Piazza

ICP.m
Written from scratch but got idea from slides and help from Piazza