Group Name: Amaranth Moose
Group Members: Kent Sommer , Joshua Stegman, Jordan Hayes
Project: 3D Object Reconstruction

Intro: (overview of project)

The focus of our project was to take a 3D object and reconstruct it using multiple images from views all around the object. This was broken up into 3 parts. The first part was to calibrate the camera which we would be using for taking the pictures of the object. The second part was to perform space carving using shape from silhouettes to get a point cloud for the object. The third part was to add texture and color to the model provided by step two. Once all the pieces were put together, the output was a dense reconstruction of the original object.

Calibration:

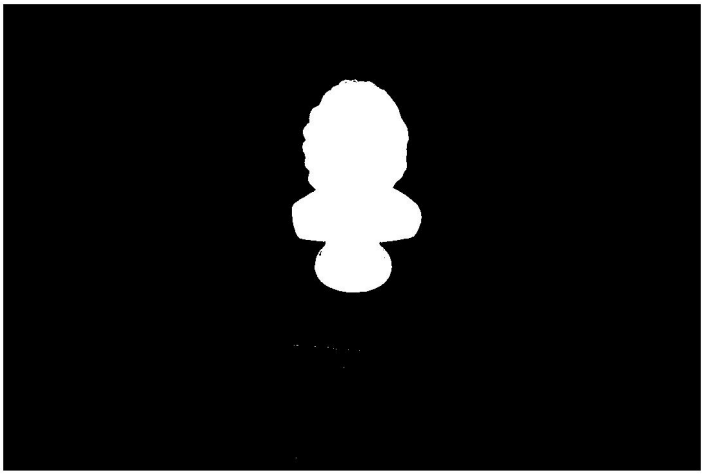
The first approach we used for camera calibration was using the Camera Calibration Toolbox that MATLAB provides. Our calibration was different than what most people try to obtain when calibrating a camera. We needed the projections matrix for each view that would be used for the object we were attempting to reconstruct. Since we needed images of that object from every angle to get as much detail as possible, we needed to calibrate the camera from all of these angles as well. Using the checkerboard pattern provided by MATLAB, we attempted taking 36 images of the pattern at equal intervals of rotation using a Logitech Pro Webcam C920. When we imputed these images into the calibration toolbox provided by MATLAB, it denied over half of our images. Even with the images it accepted, it was returning relatively high mean reprojection errors of around 1.25. After a few more tries at using this, we decided that MATLAB's toolbox might not be the best calibration tool for us. We found another toolbox provided by George Vogiatzis and Carlos Hernández for download online that did calibration in

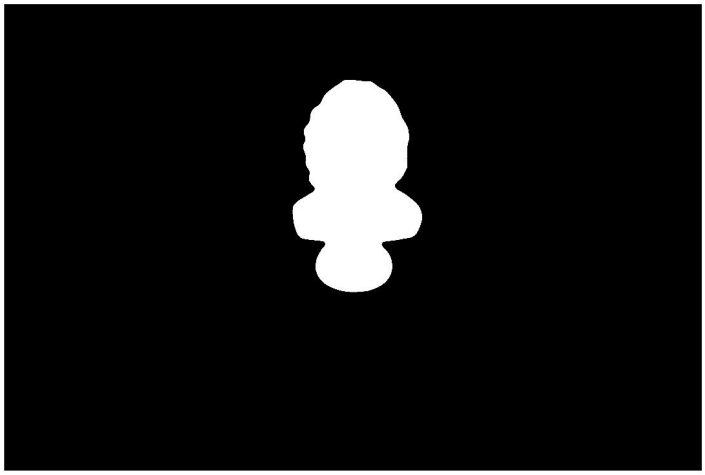
a slightly different way [1]. This approach used a dot pattern instead of checkerboard. Following a similar procedure used for the checkerboard pattern, we took 28 images of this dot pattern and ran it through this new calibration technique. This method allowed all of our images to be processed successfully; however, it was still returning relatively high reprojection errors which prevented us from being able to use the projection matrix for each image in any of our further steps in the pipeline. After seeing this similar reprojection error to what MATLAB was returning to us, we ended up trying a new camera, the MacBook air Webcam. We took the same 28 images of the dotted calibration pattern and ran those through the toolbox. This ended up returning a mean reprojection error for each image of around 0.12. This was a small enough reprojection error to allow the next steps of our program to utilize the projection matrices for each of the 28 images.

Silhouette extraction and Space Carving (shape from silhouettes):

My approach to this task ended up shifting a few times over the course of the semester as other parts of the project started to come together and necessitated small changes to the overall flow and structure of my algorithm. Silhouette extraction was the most finicky part of this portion of the project, and it is still very sensitive to noise and required multiple photo acquisition attempts before high quality silhouettes could be extracted for each view. Originally I had wanted to use chroma keying to get the object silhouette, however, once we got the camera calibration portion running it became obvious that going that route was not going to be feasible. Background subtraction was then the only other method I knew I could figure out in time for the deadline of the project. As you can see in the two pictures, it took a few attempts to make the background subtraction algorithm work properly due to either improper threshold selection or

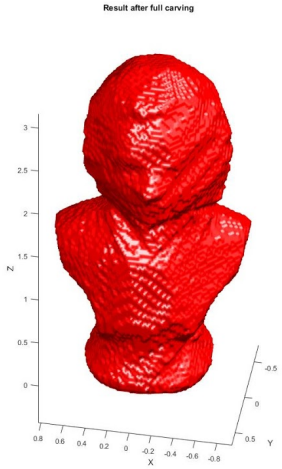
simply because the images contained noise in the form of shadows (Left is without extra processing, right is with the extra processing. It is a little hard to see but there is salt and pepper noise underneath the silhouette and a hole on the left side).





Granted, the noise isn't very significant, but holes in some of the views can cause large issues during carving.

I initially had an extremely hard time getting any results back after doing even one carve (meaning removing any points that are not consistent with one view) using our real world data instead of the dinosaur test data. The issue stumped me for a while, however, I finally realized it was an issue with my initial bounding box limit check. Restraining this limits a bit allowed the algorithm to run and finally return the expected result.



The silhouette based approach that I decided to stick with ended up giving pretty great results actually. Had I had more time I obviously would have tried to implement a way to recover the concave details of whatever object was being modeled, however, my implementation provided what I believe to be very decent results given my lack of previous knowledge in the field and relatively short delivery time frame. If I had it to do over again, I only really wish I would have started a bit sooner and been able to get to some of the more interesting issues like photo-consistency for refining the models shape. All in all, it went very well and I'm excited to continue working on it outside of this class.

Texturing & Coloring:

To texture and color the model each point in the model needs to be projected back to an image that includes that point. In most cases the point will be in multiple images, these images can see the point or something maybe in front of it. This occurs when the point is on the front of the object and is projected to an image of the back of the object. This will be a problem if the object sides are of different colors and my approach to solve the problem was to match key points.

I used vl_sift to find and match key points of image pairs because how stable SIFT is. The implementation of vl_sift was optimized in C and fitted to work in Matlab which made testing my approach fast and accurate. Once I had the matching points I removed outliers by applying a RANSAC filter. I triangulated the inlier matching points into world coordinates to bound the area of the model that was visible in the two images. Iterating over the points in the

area projecting them back to one image, checking to see if it fits within the silhouette, otherwise projecting it back to the other image to retrieve the color.

Figure 1. Patchy Dinosaur

The matching points method did not work very well, and produced results with splotches of colors, as shown in Figure 1. There is also the possibility in this approach that part of the bounding area is on the backside and gets colored in even though it is not visible in either of the images. Since this method was unable to give me good results, I was forced to try a different approach.

The next approach was to use the surface normals, provided by the matlab patch function, and the direction the camera is facing to calculate the angle between the two vectors. An angle of 0° would mean that the point is directly facing the image and an angle of 360° means that the point is facing directly away from the image. The images that matches this point is the image, whose direction dotted with the surface normal is the closest to 0°. Therefore minimizing the dot product of the two unit vectors results in the best image to project back to. The results were much better as can be seen in this image:

How we put it all together:

Putting all the pieces together presented a few challenges. The first and most obvious was that all the pieces built upon each other, so each portion could not be started (or at least tested) until the previous portion was finished. Using a sample data set from the University of Oxford allowed for both the model recoloring and the space carving portions to be started without camera calibration fully working yet. Once camera calibration was ready to go, we all got together and started putting the pieces together. This was easily the most frustrating part of the project since we did not expect nearly as many issues as we ended up with when we tried tying the portions together. With the setup we had, camera calibration wouldn't run. After fixing the setup and swapping cameras, calibration ran but the silhouette extraction method needed to be changed. Once both of those were working, adding in the model recoloring actually proved to be the smoothest integration (apart from needing to rework the algorithm since the results were not as good as required using the first method that was implemented). It took roughly ten hours in total spread out over a few days to go from separately developed segments to fully running and

integrated. Overall, integration could have gone more smoothly, however, everything did end up working correctly when finally finished.

Lessons Learned:

There are a few lessons learned from the project. The first simply being that it is entirely possible that the Matlab camera calibration toolbox would have worked fine if given images taken with the camera we ended up using. Due to our lack of knowledge in computer vision, we assumed it was an issue with how we were trying to use the toolbox and not an issue with our hardware. Had we taken the time to really investigate all of the data provided from the error results that Matlab gave (showed reprojection errors among other things), we could have avoided doing the extra work of finding an alternate calibration toolbox. The second takeaway is that it would have been much smarter to plan out the inputs and outputs of each person's algorithms and portions of the project so that the merging of code could have been done more smoothly. This is just generally a smart idea for any software project and there is no reason why we couldn't have made use of it for this project. The last major takeaway is that in some ways we bit off more than we could chew at first without really actively thinking about how difficult what we were proposing really was. We had initially thought it would be possible to build a 3D reconstruction of someones face in real time on a mobile device (this was driven by a video we had all seen showing this exact thing being done). Had we been a little more realistic in the posing of our initial problem, we could have saved a large amount of time that was used to restructure our problem as we realized how difficult our original proposition actually was.

References

[1] http://george-vogiatzis.org/calib/