Meshlab: Open Source or Licensing?

Casey Boomer, Brandon Withington, Seika Mahmud

Intro

One of the main goals of this project has been to produce a software solution that is both affordable and accessible to the general public. To that end, we have focused on utilizing open source meshing software in order to minimize costs. However, it has become increasingly apparent that the open source options available are insufficient for us to develop a high-quality product within the remaining time frame. After testing various free meshing programs, including Meshroom, Meshlab, and CloudCompare, it was Meshlab that stood out the most in terms of ease of use and functionality. Even so, after weeks of testing and researching different meshing and texturing methods, we have been unable to produce a mesh of high enough quality to suit our purposes, especially when compared to other license-required software.

In this paper, we will discuss why we believe that we should look away from MeshLab and focus on licensed software, particularly Agisoft Metashape. We will do so by explaining the work we have done with Meshlab, the difference in quality and cost, the target users of this software and the current timeline for our project.

Work with Meshlab so far

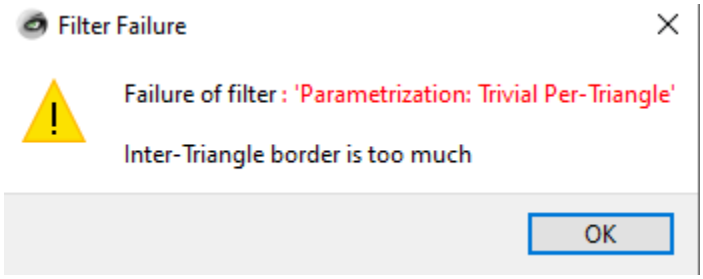
Over the course of several weeks, we have used Meshlab to produce numerous meshes with varying levels of success. Although many of them appear to be of decent quality in Meshlab's viewport, the extent to which their textures have degraded becomes apparent when imported into Blender or Unreal Engine. Ironically, increasing the resolution of the mesh results in more artifacts obscuring the texture.

We have conducted various tests on generating meshes with Meshlab testing two different types of surface reconstruction. Ball pivot and poisson reconstruction.

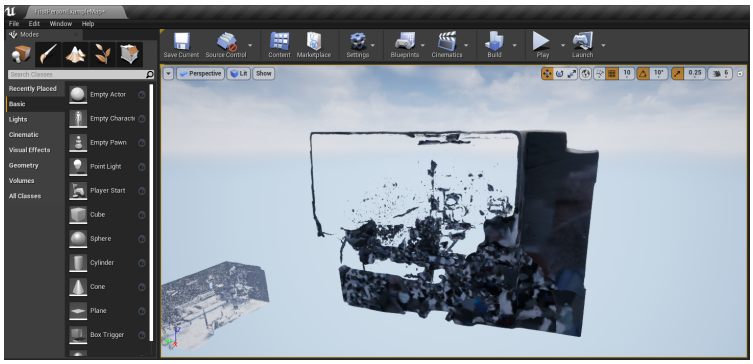
Ball pivot surface reconstruction starts with the normals of the point cloud and uses a triangle based algorithm to reconstruct the desecrated scan. This utilizes three points to form a triangle around an edge within a point cloud. It revolves around edges until all edges have been contacted. When it touches another point, it will then form another triangle and begin that process of edge detection over again. This process of generating triangles with edge detection occurs within a ball of increasing radius from the beginning point.

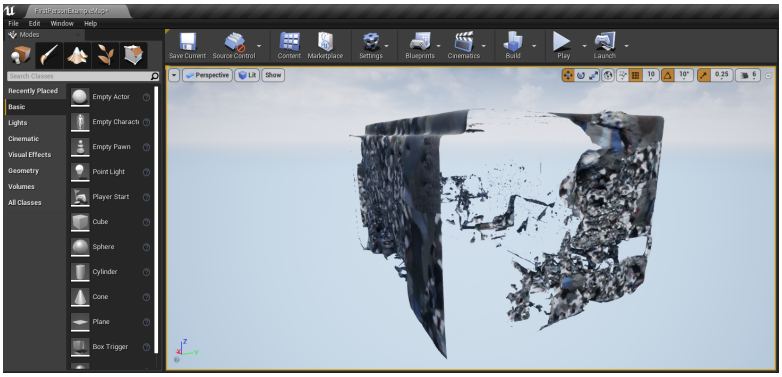
Poisson surface reconstruction is used to reconstruct 3d surfaces from point samples given to the program. It allows for the fitting of scanned data, the filling of surface holes within that scanned data and meshing of existing 3d models. Generally speaking, for both ball pivot and poisson surface reconstruction, as the number of vertices gets larger so does the number of

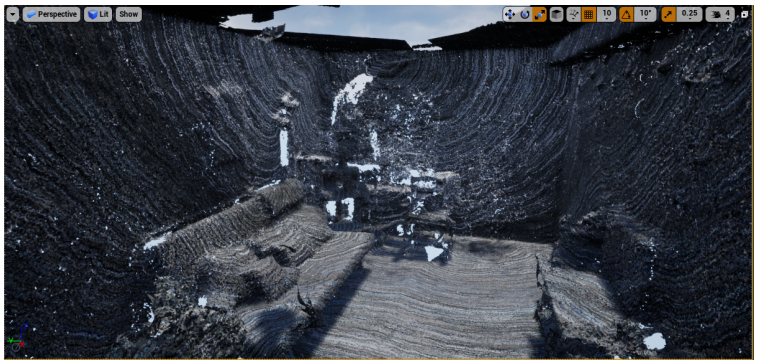
faces. This in turn increases the amount of triangles required to generate an accurate mesh for the scan. There comes a point where, in Meshlab, there are too many faces and too many triangles for its parameterization trivial per triangle texture generation method.



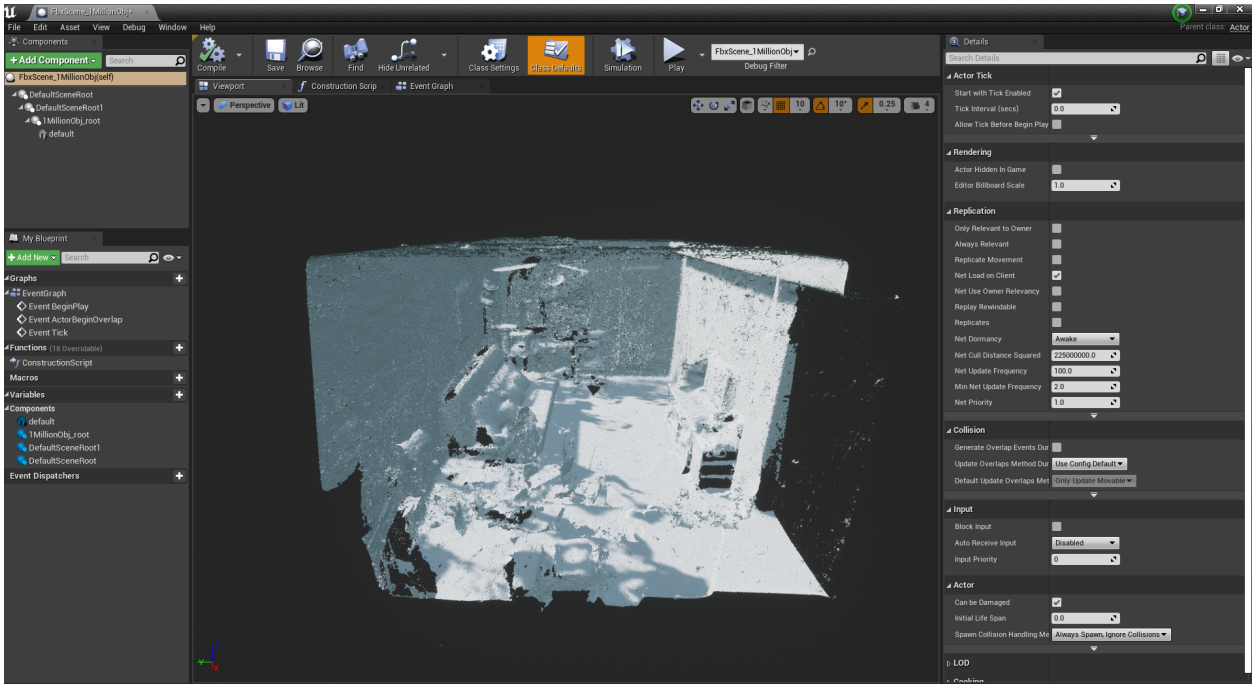
This means that the texture and mesh we are attempting to generate from Meshlab is too large for the program (Meshlab) to handle. One quick mitigation that we can do to generate some kind of texture for our OBJ is to resize the resolution of our texture. However it is impossible to know what resolution our scans are going to need let alone user scans as well. Even if we found out what resolution our scans need, it would be impossible to determine that for each and every new individual scan coming into Meshlab. Here is the result of resizing the resolution texture.







As you can see, resizing the texture resolution causes the texture for the object to either not exist or completely be distorted in an unfixable matter. This is unavoidable. These are each three different trials of attempting to resize the resolution with a stable model : this is the stable model that the textures are attempting to be applied to :



We as a team are concerned with another aspect of Meshlab: Texture and mesh generation timing. During our tests with Meshlab we continuously found that, if Meshlab did not crash, while it was generating textures and meshes it would take on average at least an hour to two hours (depending on the size of the scan) to complete generating low quality textures and meshes. Metashape takes on average 10 to 30 minutes to complete mesh and texture generation with higher quality textures. The amount of time Meshlab takes to generate textures and meshes is not acceptable.

<u>Tests with self-scanned environment and Meshlab</u> :

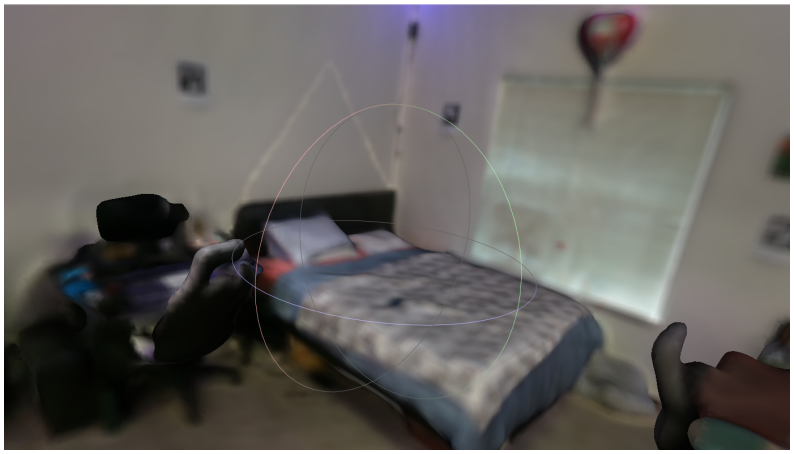
# of Vertices in Scan	# of faces generated	Able to set parameterization per triangle	Can texture colors be mapped	Face creation method used	Texture outcome
54,718	101,729	Yes	Yes	Ball Pivot	Spikey and inaccurate. Texture is not applied to the scan and cannot be viewed in other programs.
54,718	126,602	Yes	Yes	Poisson Reconstruction	Textures are accurately

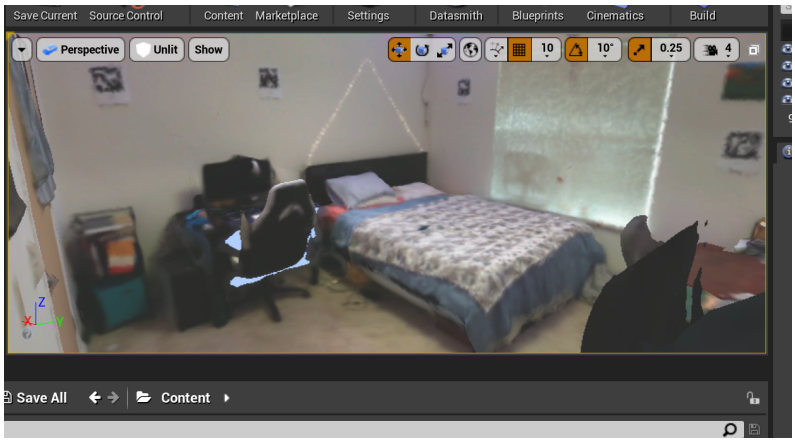
					mapped onto the 3d model however texture is blurry.
389,175	715,393	Yes (Took 40 minutes to process)	Yes	Ball Pivot	The same outcome as the first Ball pivot : spikey and inaccurate texture mapping.
1,279,652	2,186,398	No	No	None	No outcome.
2,215,564	3,885,134	No	No	None	No outcome.
61,688,364	Meshlab Crashes before faces can be made.	No	No	None	No outcome.

The reason why some of these in the table above have “none” listed for their face creation method is because they, 100% of the time, will crash Meshlab. Resulting in no mesh or textures being generated for the object.

Quality Comparisons

The quality of the meshes created with subscription-based meshing software is much better than open source software. Every mesh that has been generated from Meshlab has turned out with oversimplified details and muddy, low quality textures. Meshlab works well with small point clouds and scans with lower levels of detail, but for large-scale scans of detailed rooms, the quality of the scan begins to decimate quickly.





The goals and requirements for this project are to replicate the use of the Matterport Camera at a much lower price point. This includes creating scans that are photorealistic, and potentially trick the user into thinking that the scans are just pictures of the room. Working with Meshlab will not provide the photoreal effect that we are looking for in the project.

Quality plays an important role in a user's mindset. To a user that wants to use this as a virtual walkthrough for their real estate website, the quality of the model will give people that are looking to purchase or rent these homes a better understanding of what the building they are purchasing looks like. A prospective customer for these users may refuse to purchase the apartment or home because the quality of the pictures they are looking at are low. This source from Rismedia explains that homes with higher quality pictures increases the chance of selling a home by 32 percent. User with the intention of scanning historical locations or industrial structures would want to have a more detailed image as well, because for the historian the detail needs to be there to preserve the details from the historical structure, and the industrial worker would need to see the scans to make measurements and pinpoint specific details in their industrial structure. Quality of models is extremely important in many of the target groups that we are looking to serve.

Price Comparisons

The biggest concern for not using a licensed meshing software is the price of purchasing a license. The intention of this project is to have a cheaper alternative to the matterport camera while still providing similar services to Matterport.Though for some users, the license to use their software, specifically Metashape, it would still be a cheaper and more reasonable price than working with Matterport, and for many of our target users be a much cheaper option.

The Matterport is a very expensive camera. Currently, there is a sale on their website for the camera at $2,997. However, there is also an additional cost on top of the price of the camera. To be able to access the scanning features of the camera, you have to pay a monthly fee. The cheapest option for the camera is $69 a month, while the other option costs $309 a month. This makes using the matterport very expensive. Using Metashape on the other hand, requires a single payment of $549 for educators and $3499 for non-educators. Though for non educators the initial fee is higher than the initial cost of the matterport, the price of this software weighs itself out over just a few months of use.

This price does not include the license for DotProduct and the cost of the Intel realsense camera. A DotProduct license will cost $249 a month, and the RealSense camera will cost $239. Again, this still makes our option cheaper in the long run, but still appears relatively pricey for the average user. However, because of how the user interface is now set up, there is customization that can be done to make the product much cheaper. Because our user interface begins by taking a point cloud, the beginning part of the pipeline can be done with any software that the user chooses. We can recommend that the user uses Dot3D because of its incredible scanning features, but if the user is on a budget, they could opt to use another software, such as Meshroom to complete the point clouds, and use any camera that they would like. This means that the user could spend zero dollars on this part of the pipeline.

Spending money on the license for Metashape could potentially be expensive upfront, but compared to Matterport, this one-time price point will end up being significantly cheaper over the course of a few months.

Current Timing

As of the date that is writing this paper, there are only a couple of weeks until winter term ends, and we begin spring term. The team will not have the entire term to continue working on this project until we need to begin setting the software up to present at the engineering expo, and the video that we will create with Intel. There is not much time to finish this project. There is a good plan to finish the project with MetaShape by the spring expo through creating effective plugins and a simple user interface. However, if we were to transition back to MeshLab, or even writing our own algorithm, development could take much longer. As of right now we have not been able to put a mesh into Unreal Engine from MeshLab that has looked adequate enough. We would have to continue tweaking the pipeline to be able to start working on plugins that integrate MeshLab with the user interface. Also, MeshLab plugins require a specific compiling

process, which will lead users of our project astray when trying to set up the software. As for creating our own meshing algorithm, The development time would increase substantially due to the time researching, implementing, debugging and optimising the algorithm to work at decent processing times.

Switching back to MeshLab would set back progress on our project by a few weeks at least. This amount of time is crucial, as we are approaching the deadline for our project, and are currently recording a beta demo for the project. If we had more allotted time to develop, it would make plenty of sense creating our own meshing algorithm, as we could optimize it for our interface and make the quality of generated meshes much better. However, with the time allotted, it would be a massive crunch to complete this project on time.

Conclusion

Although we originally envisioned a product that utilizes primarily open source software in order to cut costs and increase accessibility, it has become clear over the course of several weeks that licensed software will be necessary to produce meshes of sufficient quality to serve our purposes. Through our work with MeshLab, comparing costs and quality of meshlab compared to licensed software, and looking at the path ahead to the Spring Engineering Expo, we believe that it is time to stop considering MeshLab as our meshing software of choice.