12-7-2021

Surface Reconstruction Library

Jhye Tim Chi
Western Michigan University, jhyetim@gmail.com

Surface Reconstruction

Harsh Sinha, Jhye Tim Chi
Western Michigan University
December 2021

Contents

Abstract

The project aims to convert an arbitrary point cloud into a triangular mesh. Point clouds are a list of 3d points that model the topology of an object. Point clouds can have various issues, such as missing or noisy data. For the scope, we had no control over point cloud generation. We were also unable to deal with underlying registration or alignment problems. Triangular meshes are a list of triangles that have 3d vertices. This aggregate list of triangles defines the reconstructed surface. Our project implementation is based on Alexander Hornung and Leif Kobbelt's method for surface reconstruction using the unsigned distance function [1]. Stryker did not have any preexisting software that we could use to accomplish our project. We had to implement the entire algorithm and sub algorithms from scratch.

Problem Statement

Need

We are unable to discuss the need for this technology because Stryker wishes to protect it as a trade secret.

Objective(s)

The objective of this project is to take an arbitrary point cloud and turn it into a triangular mesh. Moreover, the reconstructed triangular mesh had to be an accurate representation of the original object that was scanned. Point cloud generation was outside the scope of the project, so our group had no control over the quality of the point clouds given to us.

Terms, Acronyms, Glossary

Acronyms/Terms	Definition
Mesh	A collection of vertices and edges that define the shape of a polyhedral.
Triangular Mesh	A mesh is made up of a collection of triangles.
Morphological Dilation	An operation that probes and expands the shapes contained in an input image by using a structuring element.
Point Cloud	A set of 3D points that represent a 3D shape or object in space.
Point Normal	A point where a line is perpendicular to an object.
SRL	Surface Reconstruction Library
Volumetric Grid	A collection of voxels that creates a grid-like structure that represents the point cloud.
Volumetric Resolution	The resolution of the volumetric grid in three dimensions.
Voxel	A point in a 3D grid where each of the coordinates is defined in terms of its position. It is a singular component of the volumetric grid.
6-Neighborhood	The six voxels neighboring at each surface of a single voxel.

Problem Analysis and Research

The first major challenge that we encountered was that no one in our group had any prior knowledge about Surface Reconstruction. As a group, we decided that we would find and read current white papers published to try and gain more background knowledge in the area. Our Stryker sponsors suggested that we read survey papers on Surface Reconstruction rather than trying to go out and find research papers to read ourselves. We found that reading these papers was quite challenging because there were a lot of new concepts and terminology that we were unfamiliar with. We were able to get through this period by rereading the paper multiple times, asking questions, and doing some background research for our background research.

The next major challenge was that we were tasked with defining the methodology for going from a point cloud to a triangle mesh. We had closed-door discussions with Stryker to better understand their use case. We decided that the best route would be to take an existing white paper and implement it. We then proceeded to read the survey papers to try and identify a white paper that was suited for the intended use case. We worked to identify papers by continuously reading white papers and checking in with our Stryker sponsors on whether the said paper covered all the different things they were looking for. It was very challenging because there was a lot of back and forth involved with trying to better understand the requirements and getting feedback from Stryker.

Once we had defined the white paper that we would use for the implementation, the next challenge was understanding all the different steps they took to go from a point cloud to a triangular mesh. We had to reread the paper multiple times to try and get a better understanding of what was going. But we were able to break down the white paper into specific steps, and below we go over the challenges involved with each step.

Volumetric Grid Creation

A volumetric grid is a data structure that is typically implemented as an octree, which is analogous to a binary space partitioning tree (BSP). While a BSP partitions into two different branches, an octree partitions into eight different branches. Our first challenge was learning about the octree data structure and time and memory-efficient representations of the octree. We tackled this issue by watching multiple videos and reading other white papers that used octrees for their data structure. We also did libraries that had efficient implementations of octrees. But we were advised against using them because Stryker did not wish to deal with open-source libraries. Their reasoning for not doing so is because of the challenges they have faced in the past.

Volumetric Grid Dilation

The next challenge was dilating the volumetric grid. Conceptually, this is an easy concept because dilations simply use a morphological operator. However, this was quite challenging for us because when we create new nodes or voxels in the volumetric grid or octree, we are inserting a new point in the octree. The challenge was using surrounding voxels to try and estimate the center of the new voxel/node that is generated. Our group was able to overcome this challenge by working with Stryker sponsors to try and derive a mathematical formula based on the current voxel/node index and voxel side length to determine voxel centers.

Unsigned Distance Function

The next step was understanding how the researchers intended to use the unsigned distance function. Computationally, this was simple, because all we had to do was loop through every voxel and average the six neighborhood unsigned distance functions to estimate the new unsigned distance function. What was confusing to us was why these researchers deviated from using the traditional signed distance functions to try and reconstruct a surface. What we were able to find through research was that signed distance functions have trouble reconstructing surfaces if the input point clouds have no point normal information. Through continuously rereading the white paper, we were able to understand that the researchers used unsigned distance functions to estimate the confidence value of a voxel being a part of the optimal surface.

Graph and S-T Min Cut

The next challenge was converting the volumetric grid into the described graph structure in the white paper. This was more of an algorithmic challenge because each voxel face had a node placed on it, and voxels that shared faces had shared nodes. We had to ensure that our graph did not contain duplicate vertices in places where it should not.

After converting the volumetric grid to the graph structure described in the white paper, our next challenge was implementing the Boykov Kolmogorov min-cut algorithm. It is a commonly used min-cut algorithm used throughout various computer vision applications. The major challenge with this was understanding the implementation of the algorithm. The white paper on this algorithm simply supplied the user with pseudocode and explanations for each of the different algorithms. We had spent countless hours reading the paper repeatedly to try and implement it properly.

Triangulation

Triangulation was not as difficult as the other steps because the white paper we were using gave us a pseudocode for the triangulation. It still took some algorithmic work to get the pseudocode to work with all the different data structures and formats we used. It was the only pseudocode given by the white paper we used.

Smoothing

The challenge with implementing Laplacian smoothing was trying to understand conceptually how this algorithm worked. We were able to figure it out through Wikipedia and by watching a couple of videos on it. The most challenging aspect was trying to understand the mathematics that was involved. Once we understood the math, the implementation was straightforward.

Requirements

We worked with Stryker's Quality Management team to ensure that our requirements covered the breadth of what was needed for a Stryker software product. The following sub-sections highlight all of our categorized requirements.

General Requirements

Programming Environment

SwReqID1. *SRL* shall be implemented in MATLAB.

SwReqID2. *SRL* shall be implemented for usage in a Windows environment.

Security

SwReqID3. *SRL* exposed functions shall be reentrant.

SwReqID4. *SRL* shall handle dynamic memory allocation and deallocation internally as needed.

SwReqID5. *SRL* shall handle thread synchronization and clean up internally as needed.

SwReqID6. *SRL* exposed functions shall handle null pointer exceptions.

SwReqID7. *SRL* exposed functions shall handle incorrect data being passed in.

Quality

SwReqID8. This module shall meet or exceed all requirements specified in this SRS.

SwReqID9. This module shall be written using the naming conventions defined in Stryker IMT Coding Standards documented in [3].

Safety

N/A

Regulatory

N/A

Serviceability

Updates to SRL will be done in conjunction with end applications only. It is the responsibility of the user to integrate future versions with existing or new software applications.

Installation and Acceptance

SwReqID10. *SRL* shall be delivered as code files that can be compiled with end application.

Functional Requirements

Versioning

SwReqID11. *SRL* shall rely on release strategy for versioning

Application Interface

SwReqID12. *SRL* shall provide an interface to set desired volumetric resolution.

SwReqID13. *SRL* shall provide an interface to set desired number of morphological dilations.

SwReqID14. *SRL* shall provide an interface to set desired point cloud for reconstruction.

SwReqID15. *SRL* shall provide an interface to set desired location to save output mesh.

SwReqID16. *SRL* shall provide an interface to set desired number of smoothing iterations.

SwReqID17. *SRL* shall provide an interface to select usage of multiple threads.

SwReqID18. *SRL* shall provide an interface to set various tuning parameters for reconstruction attempts.

SwReqID19. *SRL* should provide an interface to indicate a success/failure for an attempted reconstruction.

Usage

SwReqID20. *SRL* should have allow for timeout for attempted reconstructions in MATALB.

SwReqID21. *SRL* shall return a mesh as a collection of vertices and edges.

Constraints

1) SRL itself will not provide any critical section protection or multithreaded support. The application must ensure protection for multithreaded environments.
2) SRL expects that point clouds at a minimum represent the top and all sides of the object the user wishes to reconstruct.
3) SRL is unable to reconstruct surfaces with extremely high volumetric resolutions because of memory constraints.
4) *SRL* shall not use any inter-process communication to share data with the application.
5) *SRL* should use data in-place rather than copying data from buffers to buffers.

Other

One of the requirements not mentioned in the requirements documents is that our sponsor wanted to achieve a reconstruction mean error of less than 2mm, and a max error of less than 5mm with the point clouds that they supplied to us. This was not included in the requirements documents because these errors are highly dependent on the point cloud quality that SRL receives, and SRL serves as a generic component.

Standards and Constraints

Applicable Standards

We had to implement the surface reconstruction algorithm in MATLAB for use in a Windows environment.

Constraints

In the following sub-sections, we give an ethics analysis for our project.

Introduction

The question that needs to be answered is whether hospitals should use computer vision-based medical technologies. We are unable to conduct an ethics analysis on our surface reconstruction project directly because we are unable to talk about the need or usage of the project. Instead, we offer an ethics analysis for generic computer vision-based medical technologies used in hospitals. We will not specifically refer to a unique existing product currently in the market but talk about the product as a concept.

Recommendation

After an analysis of this question using Hosmer's 6-step process for ethical decision-making, our recommendation is for hospitals to use computer vision-based medical technologies. The benefits provided by using computer vision-based medical technologies, such as recognizing hidden patterns and making diagnoses with fewer errors than human professionals, far outweigh the drawbacks.

Moral Issue

In computer vision-based medical technologies, cameras have to be able to take pictures or videos of patient-specific anatomies. For example, a computer vision-aided tumor identification application would have to be supplied with patient-specific MRI scans. This information is considered sensitive and is protected under HIPAA. This exposes this type of technology to ethical concerns because if this data were to get leaked, it would be a huge violation of the patient's sensitive data. Moreover, as computer vision-based technologies start gaining more traction, we will see them used in various operations on various body parts. One can imagine that some areas are more private than others.

Additional Facts

1. Who can access this information?
2. Are the patients aware of what data is collected and how the data can or cannot be used?
3. Is the data being logged to a local hard drive?
4. Is the computer vision application connected to the internet?

Available Alternatives

Computer vision-based medical technology is a new field that brings new capabilities to the medical field. There are no other technologies that bring the same capabilities as what computer vision can bring to the table. To ensure the absolute privacy of the patient, this technology could have restrictions placed on it that it should never connect to the internet or log any patient-specific data. This would mitigate some of the concerns of possible data breaches and malicious agents gaining access to

this patient-specific data. Moreover, this type of medical imagery is similar to a hospital storing images of x-rays. While x-rays do not have the potential to have as much sensitive information as computer vision technologies, they can still be thought of as very similar.

Personal Impacts to Decision Maker

We viewed the hospital as the decision-maker in this case. There are several ways in which either choice, using computer vision-based medical technologies or not, could affect them. If the diagnoses done using computer vision-based medical technologies were inaccurate, it could lead to the hospital losing trust among the community and future patients. Patients could also threaten to use legal action due to misdiagnosis.

Shareholder Theory

We viewed the shareholders of the computer vision-based medical technology as the corporation selling the technology to the hospitals. The goal of corporations is to maximize profits, and being able to sell computer vision products that help doctors do their jobs better would allow them to place higher price tags on the products. These computer vision products would also help build the company's brand and image if they could produce better results or ease the doctor's job. On the other hand, if these products were liable to data breaches, it would severely harm the company brand and image, and make hospitals less inclined to buy products from them in the future.

Stakeholder Theory

We viewed the stakeholders of computer vision-based medical technologies extending past the corporation to the patients, doctors, and nurses that would interact with or use this technology.

For the doctors and nurses in the hospitals, this technology could give them access to advanced diagnostic tools that can recognize hidden patterns and make diagnoses with fewer errors. With better diagnosis technology, patients would also benefit by receiving the most accurate diagnosis they require. The integration of computer vision-based medical technologies could also lower the cost of medical diagnosis for patients. However, an issue would be how the data collected could be used without the patient's consent or be stolen in a data breach.

Virtue Theory

The Virtue Theory states that every individual has the right to be free from any unwarranted publicity. Since we cannot predict if a patient's data can be breached, we look at other considerations. If the public can access advanced technologies to help diagnose them more efficiently with fewer errors, then using computer vision-based medical technologies in hospitals is recommended.

Conclusion

After applying Shareholder Theory, Stakeholder Theory, and Virtue Theory to the problem, a conclusion is clear. Hospitals should use computer vision-based medical technologies. Shareholder theory, Stakeholder Theory, and Virtue Theory lean towards hospitals using computer vision-based medical technologies. Personally, we also agree and lean towards the same conclusion.

System Design

Since we are unable to go into the system design as it is considered Stryker IP. So we will be going into more detail about the overall steps that we take to get from a point cloud to a triangular mesh.

Volumetric Grid Creation

Volumetric grids are analogous to pictures in 2d. Each picture contains a collection of pixels referred to by their x and y index in the grid. In a volumetric grid, there is a collection of voxels referred to by their x, y, and z index. Each volumetric grid defines a bounding box around the point cloud. This bounding box is thought of as the min and maxes on each axis such that the entire point cloud is contained within it. Each voxel is a smaller bounding box within the overall volumetric grid. This concept allows us to insert points from our point clouds into our volumetric grid.

We start by defining a voxel length desired which will partition the volumetric grid into a collection of voxels. Next, we insert points from our point cloud into the volumetric grid, and for voxels, with multiple points in them, we average the points (each voxel only contains a single point). As described, we have chosen to represent our volumetric grid as an octree. An octree is synonymous with a binary space partitioning tree, but instead of only partitioning into two sections, the octree partitions into eight sections. Notice that there are eight octants in a 3d graph.

Using the volumetric grids allows us to better organize the data for manipulation and retrieval. This data structure is an efficient way to store 3D data and allows for faster manipulation in comparison to using the raw point cloud.

Volumetric Grid Dilation

Once we have created a volumetric grid, we address the first challenge of missing point cloud data by dilating the volumetric grid. This is extremely similar in concept to the dilation of a 2d image. We use a 6-neighborhood dilation operator described in the paper to dilate our volumetric grid. Using a series of dilations, we can close holes that previously existed. Each dilated voxel gets filled with a point that exists at the center of the voxel bounding box. We must be careful about how many dilations we use as this increases input size and processing times in the subsequent steps described below.

One important thing is that if a whole is too big to fill in with a reasonable number of dilations, we report back to the user that this is a failed reconstruction attempt. The reason for this is through dilations, we are trying to estimate regions with holes in them, and we do not want to provide a reconstruction that has an inaccurate guess at what the surface looks like at the hole. This is influenced by our requirement for the accuracy of reconstructed surfaces.

Unsigned Distance Function

As we dilate voxels, we close holes but we also introduce uncertainty into the problem. The dilated voxels were not part of the input point cloud, and we are unsure if they are a part of the optimal surface or not. The paper uses the concept of an unsigned distance function to deal with this uncertainty. We start by assigning a value of 0.0001 to all voxels that were a part of the input point cloud and assigning a value of 1 to all voxels that were generated through dilation. Next, we start looking at 6-neighborhoods of all dilated voxels and average the value of the unsigned distance

functions. We allow the user to specify how many iterations of this averaging they would like to take place. This makes it so that dilated voxels closer to original voxels have a lower unsigned distance function value, and voxels further away from original voxels have higher unsigned distance function values. Conceptually, a closer value to zero states that the algorithm is more confident that the voxel is a part of the optimal surface.

The motivation for this step is that it allows the algorithm to assign confidences to different voxels of how confident it is that the voxel is a part of the optimal surface. The next step is to compute the subset of voxels that minimize this unsigned distance function and still form a manifold surface (no holes). Researchers have found that the fastest way possible to do this type of computation is through the S-T min-cut algorithm. Thus, it requires that we convert our volumetric grid into a graph structure.

Graph Creation

The paper defines how to convert a single voxel into the graph structure required for the S-T min-cut algorithm. It states that for every voxel that has a point in it, we place a graph node on each face of the voxel (a voxel can be thought of as a cube here). Looping through all active voxels, we had to take care of repeated graph nodes because many voxels share faces. Once we have these nodes, we must connect every node in a voxel to all other nodes in the voxel. Each edge weight in a voxel is determined by the unsigned distance function value of the voxel.

S-T Min-cut

For simplicities sake, I will not define a graph cut, min-cut, or s-t min-cut as it is assumed that the reader has sufficient knowledge of the concepts. We use the famous Boykov-Kolmogorov S-T min-cut algorithm famously used in many computer vision algorithms. The BK S-t Min cut works by using a push relabel strategy but improves the concept by using search trees such that the breadth-first search is not repeatedly recomputed. More information can be found on this through the original paper published [2]. The output of this algorithm is cut edges that represent the min-cut of the graph structure.

Triangulation

Once we have computed the min-cut edges, we can then start to triangulate the surface. Conceptually combining the min-cut edges and edge structure in voxels, we see that there are multiple 3d polygons defined. These polygons define the optimal surface. Our requirement of outputting triangular meshes requires us to loop through these polygons to transform them into a collection of triangles. Our algorithm works by looping through each polygon, converting them to a collection of triangles, appending these triangles to the overall list of triangles. After this loop, we have a collection of triangles that represent our reconstructed surface.

Smoothing

After the initial crude mesh has been generated, we allow the user to specify if they wish to apply Laplacian smoothing to the mesh as well as the number of iterations. Laplacian smoothing works by generating a linear approximation of the Laplacian operator for each vertex. It then tries to drive this approximation down by shifting the vertex in space to get rid of spikes or extremities in the mesh. By doing this process on all the vertices in the mesh, we get a smoother mesh rather than a blockish estimation generated because of the grid-like structure of the volumetric grid.

Testing

In the following sub-sections, we go over our test strategies for the different requirements found in the SRS, unit tests for implementation, and accuracy testing for reconstructed meshes.

Software Requirements Specification

Please reference software requirements IDs provided in the requirements section.

SwReqID1

After discussion with our sponsor, we decided that the best way to validate this requirement was to do a static code analysis with Stryker software engineers. We also discussed that if we wrote all of our test cases in MATLAB, it would also validate this requirement.

SwReqID2

After discussion with our sponsor, we decided that the best way to validate this requirement was to carry out the development of the SRL on a Windows machine that uses MATLAB. We also discussed that if we wrote all of our test cases in MATLAB, it would also validate this requirement.

SwReqID3

At the end of development, SRL only had a single exposed function. We decided that the best way to validate this requirement was through static code analysis showing that SRL does not hold any global data that can be accessed by other scripts or functions running in MATLAB.

SwReqID4

After discussion with our sponsor, we concluded that the best way to test this requirement was to write a unit test that did an arbitrary reconstruction and use MATLAB's in-built profiler. The profiler allowed us to view the memory utilization of the system before and after the function call. Our unit test took in a point cloud, attempted a reconstruction, and remembered memory usage before and after the reconstruction attempt. The test would then compare the before and after memory usage and pass only if they were the same. We also conducted static code analysis with Stryker Software Engineers to verify that there were no glaring memory leaks identifiable.

SwReqID5

We concluded the best way to validate this requirement was to write a unit test that attempted a reconstruction that both failed and succeeded because this would allow us to view memory and CPU usage on both attempts. We validated that the output of both attempts was as expected. In the failed attempt, we verified that a failed reconstruction error was generated. In the successful attempt, we validated that a valid mesh was generated and was the expected mesh based on the input. We were unable to do further testing for thread synchronization because MATLAB does not have any in-built thread profiling that we could use. To further ensure satisfiability of this requirement we conducted a static code analysis with Stryker Software Engineers.

SwReqID6

MATLAB does not have the concept of NULL pointers but rather has empty objects. We interpreted this requirement as if we were passing an empty object. Then we would not crash but be able to report this as an error. SRL at the end of the project only had an exposed function, so we wrote

multiple test cases where we passed in every combination of empty objects that could be passed in. The test cases verified that the reconstruction attempt failed with a NULL pointer error.

SwReqID7

SRL at the end of the project only had an exposed function, so we wrote multiple test cases where we passed in every combination of incorrect objects that could be passed in. The test cases verified that the reconstruction attempt failed with an incorrect parameter passed in error.

SwReqID8

This requirement refers to the quality of SRL and is automatically satisfied if all other requirements are satisfied. So, no unit tests or static code analyses are required.

SwReqID9

We decided that the best way to validate that SRL was written to the supplied coding standard was through a static code analysis conducted with Stryker Software Engineers.

SwReqID10

We tested this requirement by handing SRL implementation and test code files to the project head and verified that all the test cases that were written passed on his machine. Moreover, we ensured that it could reconstruct meshes with point clouds that he had generated or had access to.

SwReqID11

This requirement was more of a project management requirement, so we were not responsible for any test coverage for this one.

SwReqID12

We decided that the best way to test this requirement was to write multiple test cases that set different volumetric resolutions (voxel side length). The generated mesh in each test case would be decomposed into a list of vertices. We would loop through each vertex and find the 6 closest vertices and average the distance to each of these 6 vertices from the original vertex. Once we complete this for all the vertices in the mesh, we would average this and would end up with an average distance that any vertex is from other vertices. This average distance should be within 10% of the volumetric resolution because of how the algorithms work. This strategy ensured that the set volumetric resolution was being used.

SwReqID13

We decided that the best way to test this requirement was to write multiple test cases with the same input point cloud and a different number of dilations. The input point cloud had holes in it that were filled with any number of dilations greater than 3. Our test cases tried all dilations from 0-4 and observed the output. In the test cases with 0-2 dilations, we checked the output of the reconstruction for a failed attempt because there was a hole that we were unable to fill. In the test cases with 3+ dilations, we checked for a successful reconstruction. All test cases were done using the same volumetric resolution.

SwReqID14

We decided that the best way to test this requirement was to write a single test case where we reconstructed a mesh on a unique point cloud A and then reconstruct another mesh on a unique point cloud B. We then compared the results of the reconstructions to ensure that they were not the same mesh.

SwReqID15

We decided that the best way to test this requirement was to write a single test case where the requested mesh location was a subdirectory called ‘test_dir’. After the reconstruction, we would check if a file was created in the subdirectory with the expected name and was the expected mesh for the test case.

SwReqID16

We decided the best way to test this requirement was to write single test cases. We started by reconstructing a point cloud into a mesh with zero iterations of smoothing. We then took this mesh and applied 25 iterations of Laplacian smoothing using open-source software MeshLab. We then wrote our test case to take the original point cloud and apply 25 iterations of smoothing. Then we compare the newly generated mesh to our manually generated mesh and ensure that they are the same meshes.

SwReqID17

We decided the best way to test this requirement was to write a single test case where we attempted two reconstructions. The first reconstruction would use the single-threaded implementation, and the second reconstruction would use the multi-threaded implementation. We knew that the multithreaded implementation would be able to reconstruct a mesh faster than the single-threaded implementation. We used the same input data for both reconstruction attempts and timed the amount of time each reconstruction took. We passed the test case if the multi-threaded implementation was able to reconstruct a mesh faster than the single-threaded implementation. We had to resort to a timing-based case because of how MATLAB handles its threading and limited threading debug tools.

SwReqID18

We decided the best way to test this requirement was to write multiple test cases where we changed various tuning parameters for reconstructions. Each test case was encapsulated under a master test case that then compared the output of each of the sub-test cases and ensured that all the outputs were different because of different tuning parameters. We used the same input across all the test cases other than the tuning parameters.

SwReqID19

Many of our other test cases have been described to use the success or failure output feature of SRL. We decided that we have shown that we meet this requirement through other test cases for other requirements.

SwReqID20

We decided that the best way to test this requirement was to write a single test case. And to attempt a reconstruction with a timeout of one second and ensure that SRL adhered to the timeout.

SwReqID21

Many of our other test cases have been described using the output of SRL. So, we decided that we can satisfy this requirement through other test cases described for other requirements.

Implementation Testing

Since we are unable to provide system design, we are also unable to disclose the classes implemented and the relationship between all of them. For all the classes implemented, we wrote unit tests for all exposed functions by mocking input data and manually computing what the output data should be. We will give an overview of the general unit tests for the steps we took for going from a point cloud to a triangular mesh.

Volumetric Grid Creation

We wrote test cases where we manually created point clouds and ran them through the software to create a volumetric grid. We then checked the entire volumetric grid to ensure that only voxels that should be populated are populated. Moreover, we ensured that populated voxels started with the correct initialization data.

Volumetric Grid Dilation

We wrote test cases where we manually created point clouds and ran them through the software to create a volumetric grid. We then used the generated volumetric grid for a multiple number of dilations and compared them to the manually created expected volumetric grid after the dilation.

Unsigned Distance Function

We wrote test cases that took volumetric grids that have already been dilated and ran them through the software to compute the unsigned distance function for each voxel. We then looped through every voxel in the volumetric grid to ensure that no new voxels were populated. Moreover, we compared the unsigned distance function to manually generated expected output.

Graph Creation

We wrote test cases that took volumetric grids that have already have the unsigned distance function computed and ran them through the software to create the graph structure. We then compared the generated graph to the expected graph. This was quite a tedious task to create expected output for, so our expected output and generated inputs are quite small in comparison to what is generally encountered during reconstructions.

S-T Min-cut

Since we were tasked with implementing the Boykov-Kolmogorov min-cut algorithm we decided the best way to test this was to implement a simpler min-cut algorithm and compare the outputs. We used graphs generated through the graph creation stage and ran them through both the BK-Min-cut and Ford Fulkerson-Min-cut algorithms and compared the output of the two. We passed the test case if both outputs the same cut edges as the min-cut.

Triangulation

We used generated data from all previous steps of the algorithm and manually created the expected output. We wrote multiple test cases with different expected outputs and compared the generated outputs to what was expected.

Smoothing

We did not explicitly test for smoothing because we were covered by our test strategy for SwReqID16 described in the requirements section.

Accuracy Testing

We worked with our sponsor to define acceptable mesh accuracy testing standards. We collectively agreed to use open-source software (Cloud Compare) which gave us the ability to do mesh to mesh comparison. Cloud Compare would generate a report that gave us the distribution of errors across the entire mesh. We would use this data to generate statistical data for offline analysis. We did some research and found that cloud compare provides a command-line interface, so we created MATLAB scripts to test reconstructions and plot the mean error of the triangular mesh versus the amount of time the reconstruction took. These plots were then analyzed by Stryker. And we were given feedback on what needed to improve on. And how we could go about doing so.

Results

Realization of Requirements

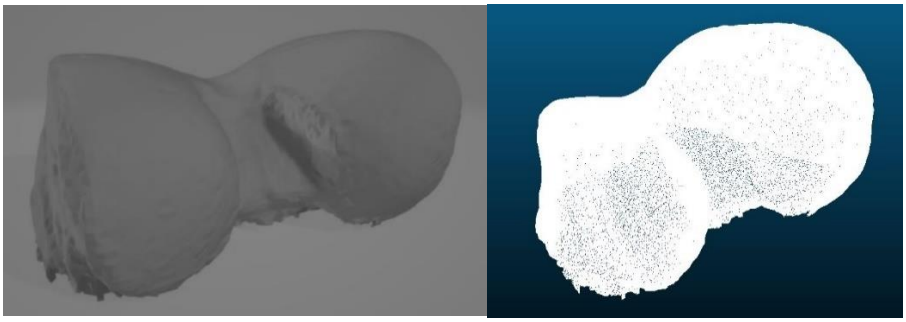
We met all the requirements specified in the Requirements section. This list of requirements was agreed upon between Stryker and our group. We gave them a final demo of the product, handed them the implementation, and all of our test code. They then reviewed the material and verified that the requirements were met, and they signed off on it.

Realization of Standards and Constraints

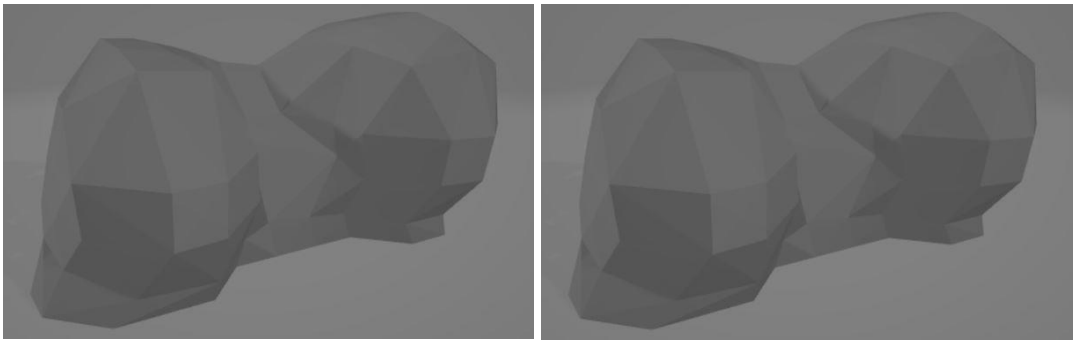
We have successfully implemented the product in MATLAB for use in a windows environment. Therefore, we have realized our standards and constraints. There are other constraints described in the requirements section that our implementation is still constrained by. However, these were constraints that were discussed and agreed upon for SRL.

Testing Results

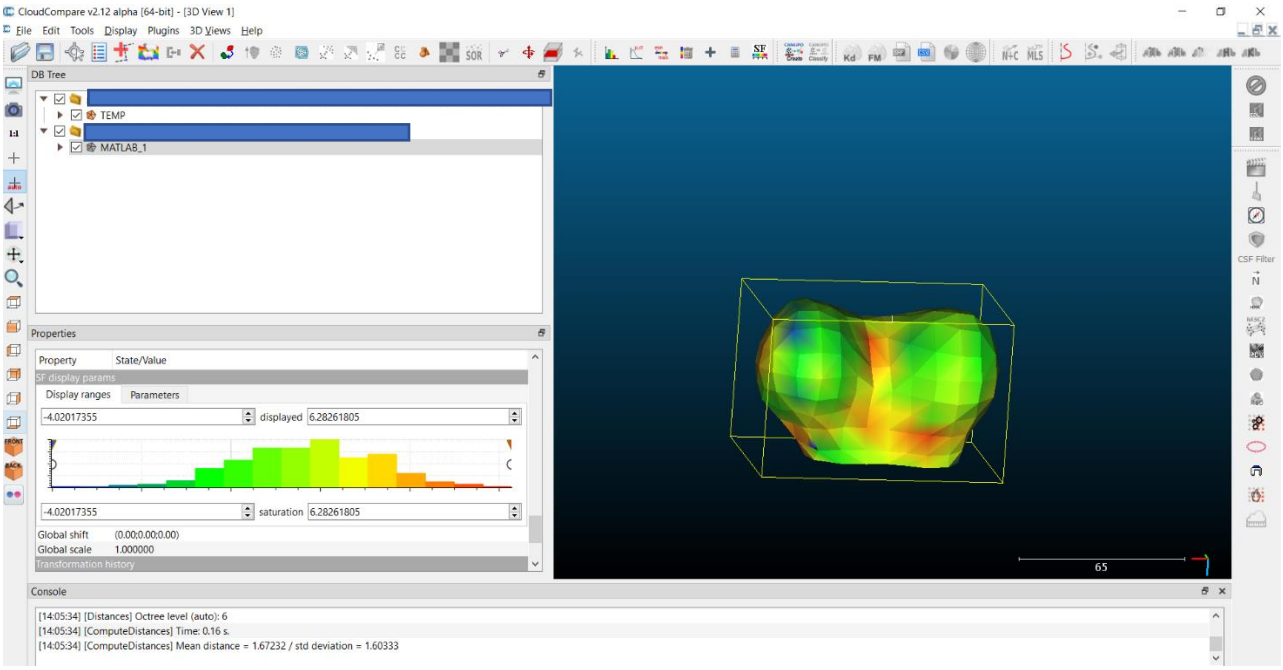
In the following section, we go over how we tested accuracies of reconstructions. We start by importing our ground truth mesh (shown below) into Cloud Compare and using it to generate a point cloud of a million points.



We then ran the generated point cloud through our surface reconstruction algorithm, which generated meshes as such.



We use Cloud Compare to run a mesh to mesh comparison using our generated mesh against the ground truth mesh (example results below).



We were asked to redact some of the names of the meshes and the locations that they were accessed from. In the example above, the mesh titled ‘TEMP’ is our generated mesh, and the mesh titled ‘MATLAB_1’ is the ground truth mesh that we used. In the console output at the bottom, we see that Cloud Compare computed a mean error of 1.67 mm and a standard deviation of 1.60 mm. In the window labeled properties, we can see that cloud compare has drawn a histogram that plots the distribution of the error. Moreover, it is also color-coded the error and uses these colors to animate the mesh shown in the picture to the right. This mesh is the generated mesh with colors that correspond to errors. This was a useful tool because it gave us and our client insight as to where the errors were coming from.

Future Work

Despite meeting all of the requirements that were agreed upon between us and our client, we believe that there exist areas where we can improve the implementation to gain better performance. In the following subsections we highlight areas where we think with more time, we can make huge performance improvements.

S-T Min-cut Algorithm

While we did implement one of the most famous S-T min-cut algorithms for computer vision problems, there has been a lot more research coming out in this field since the time that the original paper has been published. This algorithm accounts for over 75% of reconstruction times, and any work done to reduce this would allow faster and more accurate reconstructions. One idea that researchers have proposed algorithms for is a parallel Boykov-Kolmogorov S-T min-cut algorithm that takes advantage of multi-core processors to efficiently solve the S-T min-cut problem. Another paper introduces the use of GPUs to speed up the computation of S-T min-cut. Overall, this algorithm is part of the NP-Hard algorithm class and does not have a computationally fast implementation. [3] [4]

This was not originally in the scope of this project, so we did not work to implement this feature. We have brought this to the attention of our sponsor, and they have noted this as an area for improvement for future Stryker software engineers to work on.

Graph Creation

The graph structure that we implemented was a simple adjacency matrix. The rationale for this was that MATLAB is optimized for working with matrices and does not handle dynamic memory allocation well. In comparison to C++, MATLAB is often 3x-10x slower when trying to dynamically build an adjacency list. This adjacency matrix uses the most amount of memory in comparison to all other algorithms and data structures. Being able to reduce this memory footprint would allow SRL to perform using hardware with limited memory.

Researchers proposed a compact representation of the adjacency graph that exploits the connectivity generally seen in computer vision problems. For example, our voxels are what we considered to be six connected (each voxel face touches another voxel). This structure shows up in the way the graph is created since many voxels share the same graph nodes as described in the system design section. The researchers stated that this reduced memory footprint of the graph would also decrease the overall computation time for the S-T min-cut algorithm because the new graph structure would showcase better spatial locality. [5]

Smoothing

For the scope of this project, we were tasked with implementing Laplacian smoothing which is one of the simpler smoothing algorithms. However, Laplacian smoothing suffers from mesh shrinkage over a period of multiple smoothing iterations. We could remedy this problem by implementing Taubin smoothing which is a smoothing algorithm that does not cause as much shrinkage in the original mesh. While most smoothing algorithms have some form of shrinkage, the Taubin smoothing algorithm is among those smoothing algorithms that do not cause as much in comparison to others. [5]

Conclusion

At the start of this project, no one in our group had any experience with surface reconstruction algorithms. We started this project by researching current industry standards and surveying the latest white papers published to try and gain more knowledge. Through the continuous presentation of our knowledge to our sponsor and continuous feedback, we were able to start developing an understanding of state-of-the-art surface reconstruction algorithms.

Once we had the technical knowledge to start discussions with Stryker on what they were looking for, we were able to start soliciting requirements. We spent weeks conducting use case analysis on multiple white papers on surface reconstruction. We would generate an overview, pros, and cons of the approach and present them to Stryker. We would also give a rationale as to why a specific approach fit or didn't fit their requirements. Through these discussions, we were able to generate an agreed-upon requirements document for the project as well as an agreed-upon approach for the project.

We started by reading Alexander Hornung and Leif Kobbelt's method of reconstruction using unsigned distance functions and breaking it down into a list of achievable milestones. We were able to break it down into the steps that have been talked about in the previous sections (volumetric grid creation, dilation, etc.). As we read more and more into the specific subsections, we realized that we had to do more and more research to understand what was required for each step of the process.

During this research phase, we were constantly watching videos and reading white papers on various topics to bring ourselves up to speed on all technical concepts required for the implementation. Moreover, we had weekly meetings with our sponsors who were available to answer technical questions to the best of their abilities or point us to sources that would help our research.

Once we had enough background knowledge, we started implementing the white paper using an extreme programming project management style. During this time, we also held frequent meetings with our sponsor to showcase progress as well as start discussing our testing strategy to validate our requirements. We spent the entire summer implementing the white paper and test cases to ensure that our project was ready to hand off to Stryker.

At the end of the summer, we were able to give Stryker an implementation of the white paper that had passed 100% of the test cases that we had implemented. We also worked on giving demos of usability and reconstruction accuracy to engineers at Stryker before they signed off on the project. Throughout this project, we worked with Stryker and WMU to ensure that no NDAs were being breached and that requirements for the senior design project were being met. There was a small portion of our project time that can be allocated to this synchronization between Stryker and WMU.

Overall, we were able to start with no prior knowledge in the field of surface reconstruction and end with a software solution capable of reconstructing an arbitrary point cloud. Moreover, we worked closely with Stryker to ensure that the end software product was one that they were satisfied with, and could be used for however they intend to use it.

References

[1] Hornung, A., & Kobbelt, L. (2006, June). Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In Symposium on geometry processing (pp. 41-50).

[2] Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE transactions on pattern analysis and machine intelligence, 26(9), 1124-1137.

[3] Bokhari, S. H., Çatalyürek, Ü. V., & Gurcan, M. N. (2014). Massively multithreaded maxflow for image segmentation on the Cray XMT-2. Concurrency and Computation: Practice and Experience, 26(18), 2836-2855.

[4] Jamriška, O., Sýkora, D., & Hornung, A. (2012, June). Cache-efficient graph cuts on structured grids. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 3673-3680). IEEE.

[5] Taubin, G. (1995, June). Curve and surface smoothing without shrinkage. In Proceedings of IEEE international conference on computer vision (pp. 852-857). IEEE.

Appendices

Project Management Plan

Our group decided to use an extreme programming style for our program management. At the start of the project, we were concerned with completing the project within the time frame given because of our lack of knowledge in anything surface-reconstruction-related. We figured that finishing tasks as fast as possible was in our best interest to complete the project on time.

After we had decided on a white paper to base our implementation on, we started brainstorming milestones that we could have for our project. Our paper was laid out in a way that made it simple for us to define milestones. We created a total of seven milestones (volumetric grid creation, volumetric grid dilation, unsigned distance function, etc.). We defined that successful completion of a milestone included a fully working implementation of the feature, fully implemented test cases and that the implementation passed all the associated test cases.

Since there was a lot of uncertainty with what the end product would look like, one challenge was that we were not able to implement a proper design for the software. The way we dealt with this is by working until we felt there was a logical reason to refactor or combine different functions into a class. For example, when we saw that we had any repeated code, we designed a temporary software solution to eliminate duplicate code. Moreover, when there were functions that logically made sense to group together, we would refactor them into a class. For example, it made sense to refactor volumetric grid creation and dilation into a single volumetric grid class rather than having them be separate functions or scripts.

As the project progressed, this refactoring ended up taking a good amount of time while implementing. Because we constantly found ourselves in situations where we could make improvements to the overall design. Where this extreme programming broke down a bit was when we found ourselves refactoring certain functions over and over again. While this was not ideal, this was expected since we did not know the full system design before the start, and this was a hurdle that we just had to get over.

We held weekly meetings to show our progress and show the current state of design of the software. Through this, we were able to get constant feedback on the design and priorities of certain tasks. We would also take this time to review the tasks that we would be working on over the next week. These meetings allowed our sponsor to have full visibility as to what was going on with the project and allowed us to get any feedback or technical guidance from them. Moreover, we also held meetings as needed over Teams for our group to discuss priorities, next steps, and to help with any technical challenges.

As we got further into the project, we started dealing more with trying to gain better performance of our algorithms and better accuracy of our reconstructions. Our group naturally split into two workstreams to make this happen. One person would work to gain better accuracy and performance, while the other would make these changes and refactor them into the overall implementation. This person was also responsible for running the changes against the new features and making sure that they passed.

Overall, we had an extremely positive experience working together and with Stryker on this project. Our group's dedication and motivation allowed us to complete the project.

Progress Reports

We worked on this project over the summer and provided progress reports to our client on a weekly basis. Moreover, they have reviewed our final implementation and are satisfied with the results.

Development Costs

In the following sections, we go over costs for the development of this project incurred by the three different parties interested in this project.

Institution Costs

There was no direct monetary cost to WMU, however, indirectly they had to provide us with access to MATLAB (accessible through computer labs or remote access). This was already provided to students, so we do not consider it a direct cost.

Sponsor Costs

There was no direct monetary cost to our sponsor since we all had access to MATLAB through WMU. Our sponsor cost can be thought of as the time that full-time engineers, quality engineers, and project managers spent helping us with our project. For example, the software engineers that conducted our static code analysis.

Team Costs

We did not have to spend any money to complete this project, as our sponsor was ready to cover any costs that came up. The only cost to our team can be measured in the time we spent working on the project.