University of New Mexico

Adaptive Mesh Based Surface Reconstruction For Noisy and Incremental Point Cloud Data Sets

A thesis proposal submitted in partial fulfilment for the degree of Masters of Science

Lucas Chavez
lucasc@unm.edu

Department of Mechanical Engineering
University of New Mexico

Advior
Dr. Ron Lumia

December 2012

Contents

1 Goal

The goal of this work is to design a method which can create a reliable representation of the environment from sequential registered noisy point cloud data sets. The method must be computationally feasible for online applications and have a low memory requirement. In addition, the method must adapt the representation to new measurements of revisited parts of the environment.

2 Problem

A rich representation of an environment is essential for most autonomous systems because it allows the agent or operator to have an increased situational awareness of the world. The methodology to build this representation is a continuously evolving subject in the field of robotics. The origins of the research into this problem date back roughly 25 years. Since then the methods and the representations themselves have continued to evolve at an impressive rate. The main catalyst behind this growth is the advancement of sensing technologies over the same time period. In general, sensors have continued to generate more measurements at higher rates over the years. This has provided an amazing opportunity to build richer and more useful representations of the environment.

In robotics the map building in an unknown environment is referred to as the Simultaneous Localization and Mapping (SLAM) problem. This label describes the fact that a methodology which solves the SLAM problem must simultaneously locate the robot in the environment as well as mapping the environment. The methodology by which the representation is built is called mapping and is the focus of this work. Early mapping methods represented the environment as a set of landmark locations. The result was a sparse set of points usually on a 2D plane. This allowed research to show that their SLAM solutions worked but it soon became clear that a richer representation of the world was needed. In response several methods were developed using various other representations. The resulting representations can be seen on Table 1.

	Adaptability	Computationally Inexpensive	Low Memory Requirement	SA: Robot	SA: Human
Landmark Locations	x	x	x	-	-
Point Clouds	-	x	-	-	-
Surfels	x	x	x	-	x
Implicit Functions	x	-	-	x	x
Static Mesh	-	x	x	x	x
Adaptive Mesh	x	o	o	x	x

Table 1: Characteristics of current forms of representation

Table 1 compares the characteristics of the representations which are currently used by mapping methods. Adaptability describes the ability of the representation to correct itself given new information. Computational expense describes how difficult it is to create and maintain a representation. Memory requirement describes how large much memory a method must use to run. Situation Awareness (SA) describes how well suited a representation is for both robot and human decision making. Robot decision making requires a representation that can be used for such problems as obstacle avoidance. Human decision making requires a method that can be allow an operator to intuitively understand the state of the robot given the map. The Table is supposed to reflect what a representation is capable of and not necessarily where the state-of-the-art is.

A mesh based representation is arguably an extremely good choice in comparison to the other representations. It has been used as sole representation by the gaming community because it is the best for representing large environments with the minimum needed memory. Also, this sort of representations works well to increase the SA of a robot because methods for performing physical simulations such as obstacle collision already exist. In addition, a mesh based environment is a very natural and comfortable method to display information to a human operator.

Currently, the problem with mesh-based environmental mapping techniques is that they are greedy. Once the mesh is in place there is no mechanism to adapt to newer measurements. The problem of adapting a mesh to new information is a very well studied problem in computer graphics, but these methods were not designed with large scale environmental mapping in mind. The biggest questions are: How can we quickly decide which measurements should be used to adapt what part of the mesh? How can we quickly detect new and removed objects? How can we robustly deal with noise and obtain a methodology that makes use of the new measurements of an already existing part of the representation? So the real question is can we develop a methodology that can solve all of the above questions and still have a manageable memory requirement and be computationally feasible?

3 State of the Art

A major problem in robotics has been and continues to be: How can we create the best representation of an unknown environment? This question has been termed by the robotics community as Simultaneous Localization and Mapping (SLAM). This body of research has continued to grow over the years and there have been many good solutions. However, it is difficult to say that the problem is solved. In reality, the quality of the solution depends on the application. For example, it is perfectly acceptable for the environment to be represented by a sparse set of points for a satellite in space but a surgical robot needs a much denser representation of the world in order to plan movements. In the following sections we will outline the two core technologies which are now being used in the current state of the art of the SLAM problem. The first being the early

work of SLAM which, in general, aimed to create sparse representations of the environment. The second being a field named Surface Reconstruction from Computer Graphics. The goal of Surface Reconstruction is to create a very accurate representation usually at a very high computational cost and not in real-time. These core technologies are now at use by the state of the art in dense SLAM methodologies. Therefore, we will end by discussing current dense SLAM technologies and explore what is next.

3.1 Early SLAM

Early solutions to the SLAM problem began with the work of Smith et al. [5] in 1990. In this work Smith laid out the mathematical framework of which most early SLAM works would follow.

3.2 Surface Reconstruction

3.3 Current SLAM

4 Contribution

This work will add to the state of the art by creating a new methodology to leverage the depth image of the RGBD sensor to quickly identify the measurements of the depth image which are new and which are supporting a preexisting surface. The method to perform this categorization has never been performed on the depth image. The advantage of performing it on the depth image is that computationally efficient machine imaging techniques can be used. This method also quickly assigns the new measurements with the corresponding part of the mesh that needs to be adapted. In addition the adaptation procedure leverages QEM methods to perform the adaptation. Previously, this method had not been applied to adapting a preexisting mesh to new data. This methodology will avoid the trouble of over smoothing the mesh which are inherent in other methods such as virtual springs. The QEM adaptation is a computationally efficient method to create and adapt a piecewise smooth representation while preserving sharp details.

5 Impact

There are two main autonomous applications which will benefit from a richer representation of the environment.

A methodology for generating a rich representation mesh based representation of the environment would be useful for all autonomous systems which require situational awareness. There are many classical robotic applications which seek to reason about the environment in which the

robot is working in. One major example would be object manipulation with a robotic arm. There are many already developed methods which use physical simulation methods to perform grasp selection and trajectory planning. In addition, mobile robotics can use a mesh map to perform path planning and obstacle avoidance. In general, a mesh a useful representation to plan actions and simulate those actions because such simulation engines already exist due to the gaming community. Planning those actions correctly requires a correct mesh. The proposed adaptive mesh procedure can give the best representation using the prior measurements. It can also work in a dynamic environment which is needed for real world applications.

In addition, an adaptive mesh based mapping procedure is needed for Urban Search and Rescue (USAR). The work of Bruemmer et al. [1] and other similar works [2, 4] has shown that the most effective user interface is one which display information in a similar way to many popular First-Person Shooter (FPS) video games such as Quake and Half-Life. The most natural way to build these types of interfaces is by rendering a mesh. Mesh rendering techniques, hardware, and software are already optimized for using meshes. Other types of representations can be displayed as a mesh, such as implicit functions, but they require a post processing step. The proposed methodology will build and maintain the map directly. The main impact of this work on USAR applications is that a much better mesh will be shown to the user than a mesh created through greedy methods, which is the current state of the art. The mesh presented will adapt to true environment over time and with enough measurements. In addition, it will be able to operate in dynamic environments which have new or removed objects. In this way, the operator will always have a better awareness of the real environment.

6 Approach

A clear idea of the algorithmic structure of the proposed system is given by the System Flow Diagram in Figure 1. A basic description of the main variables can be found in Table 2. The inputs to the system are RGB-D data from a Kinect-style sensor D and the pose of the sensor P. The end goal of the system is to update the current mesh representation in each iteration. This update is represented by the last step in the System Flow Diagram. We can see that the update step is primed by two distinct processes. In the diagram, each processes is signified by a blue background. On the left hand side we have a processes which is named the Triangulation Process. This process defines a triangulation T based on the current depth image. On the right hand side we have a processes which is named the Categorization Process. This process categorizes the measurements based on the effect they will have on the model. This triangulation and categorization will be used by the update procedure to efficiently evolve the map based on the current sensor measurements.

In the remaining sections of the Approach we will discuss the two processes which prime the update step in detail. Finally, we will discuss how the outputs are used to update the existing mesh

Variable Name	Description
D	Depth image from RGB-D sensor
F	Frequency response image
$K_S, K_L,$ and K_G	Image convolution operators
V and C	Mesh vertices and connectivity of the current triangulation
T	Current triangulation. Contains both V and C
P	The known pose of the sensor
E	Expected depth image
$D_u, D_s, D_n,$ and D_r	Regions of the D classified by the effect on the model

Table 2: Basic description of the main variables

map.

6.1 Triangulation Process

The goal of this process is to use the current depth image D to estimate a mesh of the current view of the environment. We can see a simplified system flow chart of this process in Figure 2. Also, we can see an example of the outputs of the process using an example input in Figure 3. It is important to remember that not all of these new elements will be used in the update. The decision of which elements from T to use will be based on the output of the Classification Process.

6.1.1 Frequency Response Image

The objective of this step is to use image processing techniques to quickly give an estimate of the frequency content in the depth image D. The end product will be an image F which is the same size as D and will have high values in regions with high change. In order to accomplish this we will use a sequence of image convolutions. Equation 1 gives the mechanics of the calculation. The Sobel operator K_S is used to give a response at corners since it is a first order differential operator. The Laplace operator K_L is used to generate a response in areas of curvature since it is a second order differential operator. The Gaussian operator K_G is used to spread the response to the neighboring areas.

$$F = (\mid D * K_S \mid + \mid D * K_L \mid) * K_G \quad (1)$$

K_S – small Sobel operator

K_L – small Laplace operator

K_G – large Gaussian operator

6.1.2 Histogram and Sample for Vertices

The objective of this section is to use the frequency response image F to create a set of vertices V. These vertices will be defined as locations in D. The idea is to have a denser number of vertices in regions of D which have a high frequency content. In order to accomplish this, we first define regions of similar frequency content by histogramming F. An example of this histogramming can be seen in Figure 3d. Next we will probabilistically sample F to define a set of vertices V. The probability of each pixel being sampled is given by Equation 2. The probability is calculated by the product of two different weights: W_F is based on the region of F where the measurement comes from and W_A is the proportional area of that region. The result of sampling for vertices can be seen in Figure 3e. In addition, because the probability of picking each pixel as a vertex can be calculated independently the process is parallelizable.

$$p(u, v) = W_F(u, v) * W_A(u, v) \quad (2)$$

6.1.3 Determine Connectivity

Here we will use 2D Delaunay triangulation to define a connectivity between the set of vertices V found in the previous step. We are able to define the connectivity in $\mathbb{R}^2$ space because the topology is conserved as we project the elements into $\mathbb{R}^3$.

6.2 Classification Process

The goal of the classification process is to use the difference between the actual depth image and the expected to classify regions of the depth image by the effect they will have on the model. In order to generate an expected depth image E we will use the existing mesh map M and the known pose of the sensor P to create an artificial depth map of the of what we expect the sensor to see E. We can then use image differencing and binary blob detection to segment regions of the depth map D.

I don't think there is a Ref to Fig 4 in your text.

6.2.1 Generate Expected Depth Image E

We will use an existing graphics code library named OpenGL to generate an expected depth image E. A similar approach was used by Fallon et al. in [3]. Figure 5 is a figure from Fallon's paper which gives a clear idea of the proposed method. The procedure will be to give the existing mesh model M and the current pose P to OpenGL and render a depth buffer. It is possible to define the intrinsic parameters of the sensor to match the actual sensor. Figure 5a and 5b give an example of this process. Figure 5b represents the output of this step being the expected depth image E.

what about c and d?

6.2.2 Find Unknown Parts of the Scene

The objective of this step is to determine regions of the depth image D which correspond to areas of the environment which have never been seen before. This will be accomplished by finding regions of E ~~which~~ that have no measurement. This occurs when the ray traced through the pixel never hits a mesh element. Regions which are unseen will be designated as D_U and the rest will be designated as D_S.

6.2.3 Find New and Removed Objects

The objective of this step is determine if measurements from the known region of the environment D_S correspond to new D_N or removed D_R objects in the scene. If they do not belong to D_N or D_R then they support an existing surface in the mesh map. Essentially they belong to a supporting region D_S until proven otherwise. In order to prove otherwise we will use image differencing between the expected E and the seen parts of the environment D_S. We will threshold the differenced image with $-\epsilon$ and $+\epsilon$ to make two distinct binary images. Blob analysis will then be run on this image to determine D_N and D_R.

6.3 Update Mesh

??

This is the last and most important step of the proposed method. Here we will combine the triangulation T found in the Triangulation Process and the regions D_N, D_S, D_R, and D_N found from the Classification Process to efficiently update the existing mesh map M.

6.3.1 Add or Remove Mesh Elements

If the measurements in D correspond to new objects D_S or to unseen parts of the environment D_N, new mesh elements will need to be added. The new elements will come from the triangulation defined in T. In order to successfully merge the new elements with the existing mesh M the

bordering vertices of the region will need to be found and stitched. It is proposed that this merging operation can be done in D.

If the measurements in D correspond to removed objects D_R, then the elements are removed from M. This is done by marking the vertices within the D_R region and deleting them and their connections.

6.3.2 Adapt Existing Mesh

Regions of the depth map which support an existing surface of the mesh model will be used to adapt the mesh in order to better approximate the real world. The idea of this process can be seen in Figure 6. In this figure the real world is represented by the blue surface in Figure 6a. The red dots represent the measurements from D. The mesh M is represented by the green surface. In the figure we see a single vertex being adjusted. In Figure 6b a set of planes are defined at the surrounding vertices and through the measurements which correspond to this particular vertex. This neighborhood of measurements will be found by defining a neighborhood in D. In Figure 6c the vertex is adjusted. This adjustment will be done by applying the Quadratic Error Measurement (QEM). The QEM will adjust the vertex to minimize the distance between all planes which were found in Figure 6b.

7 Validation

In order to evaluate and quantify the effectiveness of the proposed mapping methodology, a series of experiments will be run in simulation. The reason for validating through simulation is that we will have control of all aspects of the experiment. In addition, we will have a known position of the sensor in a known environment. This will allow us to quantitatively measure our error. Figure 7 gives an idea of how the simulation will be accomplished. We will use a 3D modeling software named blender to create the environment and save it to a .ply file. Then, we will use OpenGl to open the .ply file and simulate readings from an RGB-D sensor. Finally, we will port these measurements to Matlab where the proposed mapping algorithm will be developed and tested.

With this simulation pipeline we can design a set of experiments which will sequentially test the abilities of the proposed mapping system. The idea will be to start with the easiest tests first in order make sure the system can map the environment under the most ideal conditions. Then, each subsequent experiment will aim to test a particular part of the system. By testing in this sequential manner we will be able to isolate and troubleshoot problematic parts of the system. Consequently, we will gain greater insight on the behavior of the system as a whole and the final system will be robust. The last experiment will test the robustness of the system with real world data. The following lists the proposed experiments and briefly describes the intention behind each

experiment. For discussion purposes we can envision an environment which is made of a table and a can on the table.

1. Static scene; static object; static sensor
   Here we will test the system under the most ideal conditions. We want to see that the system will categorize all measurements as D_S after the initial mesh is created. We will also test the ability of the system to adapt the current mesh over time.

2. Static scene; static object; dynamic sensor
   Here the sensor will move in the environment. For example, we can have it pan across the table by 1 meter. We want the system to recognize measurements which are from unseen parts of the environment and categorize them as D_U. Also, we want new elements to be added in unknown regions using the triangulation T from the Triangulation Process.

3. Dynamic scene; static object; static sensor
   This experiment will test the ability of the system to detect new and removed objects. We will do this by spontaneously adding and removing a second object in the scene such as another cup on top of the table. We will be looking for the system to successfully categorize the measurements as either D_N or D_R. We will then test the ability of the system to quickly remove or add the corresponding mesh elements from the current mesh.

4. Dynamic scene; dynamic object; static sensor
   This experiment will also test the ability of the system to react to new or removed object, however the object will be moved into place over time. This will be a much more thorough testing of the Categorization Process and the ability to quickly add and remove elements.

5. Dynamic scene; dynamic object; dynamic sensor
   This experiment will test the entirety of the system in simulation. We can have the sensor circle the table while new elements are being added and removed.

6. Real world data
   This test will show the ability of the system to work with real world data from a RGB-D sensor. We will make use of an open source data set which is complete with pose information.

8 Tasks

There are six major phases which will need to be completed for this research. The following sections will list the steps which must be completed for each phase.

8.1 Simulation Pipeline

- Model all needed environments and object in Blender.
- Simulate a RGB-D sensor viewing the environments using OpenGL.
- Read in the simulated sensor output to Matlab.

8.2 Triangulation Process

- Calculate frequency response image F
- Sample F to obtain vertices V
- Triangulate vertices to obtain T

8.3 Classification Process

- Generate expected E from current mesh M
- Image differencing and blob analysis

8.4 Map Update

- Adaptation procedure
- Add/remove elements

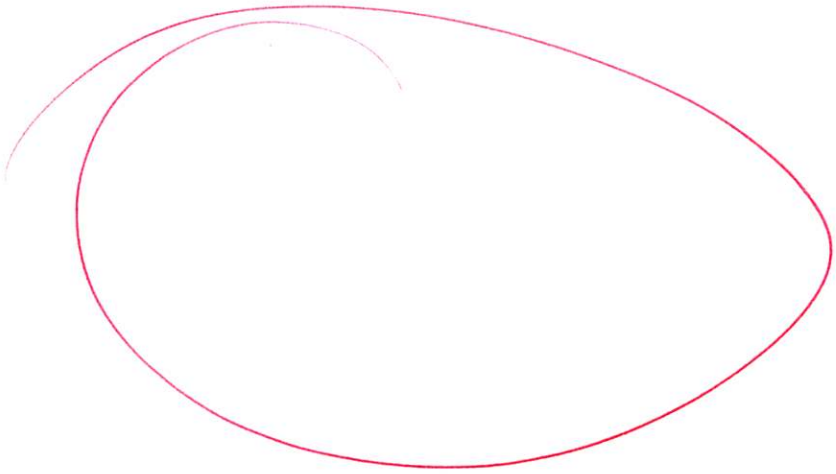
8.5 Experiments

Run validation experiments 1-6 as discussed in the Validation section.

9 Gantt Chart

In order to complete this work it will be necessary plan the completion of the major tasks which were listed in the Tasks section. Figure 8 shows a Gantt Chart with the deadlines for each of the 5 major tasks and also shows the time needed for writing the Thesis. The Simulation Pipeline will be created first in order to have a database which will be used in code development of the other processes. It is important to note that some prior code development has been started and is represented by the gray portion of the task bars in Figure 8. I plan to defend my thesis April 15th.

10 Committee



References

[1] D.J. Bruemmer, D.A. Few, R.L. Boring, J.L. Marble, M.C. Walton, and C.W. Nielsen. Shared Understanding for Collaborative Control. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):494–504, July 2005.

[2] J.L. Drury, J. Scholtz, and H.A. Yanco. Awareness in human-robot interactions. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference*, volume 1, pages 912–918. IEEE, 2003.

[3] Maurice F. Fallon, Hordur Johannsson, and John J. Leonard. Efficient scene simulation for robust monte carlo localization using an RGB-D camera. In *2012 IEEE International Conference on Robotics and Automation*, pages 1663–1670. IEEE, May 2012.

[4] M. Waleed Kadous, Raymond Ka-Man Sheh, and Claude Sammut. Effective user interface design for rescue robotics. In *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction - HRI '06*, page 250, New York, New York, USA, March 2006. ACM Press.

[5] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, 1990.

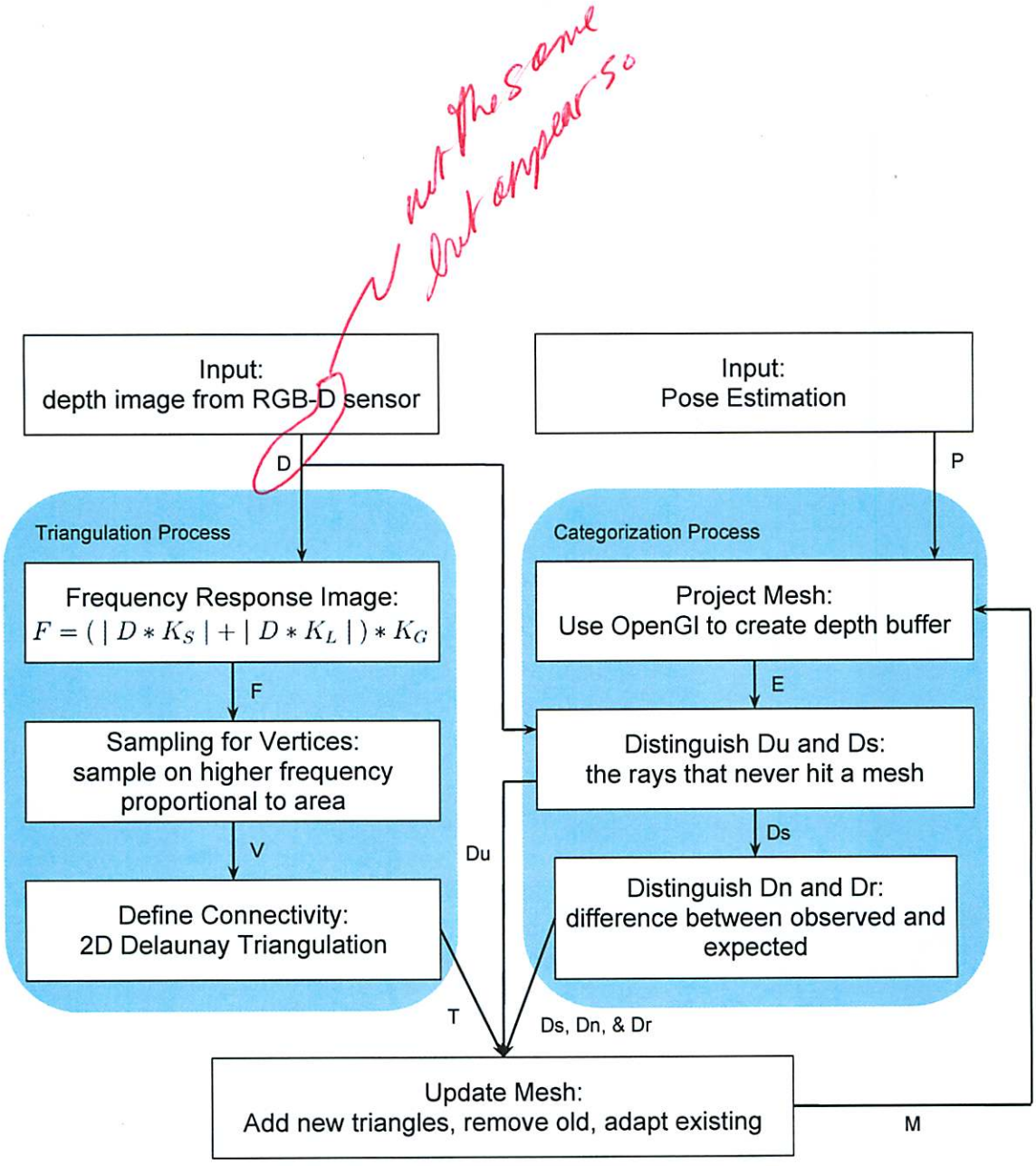


Figure 1: System Flow Diagram

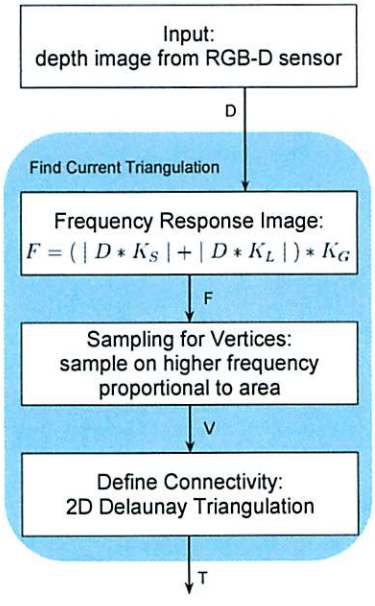
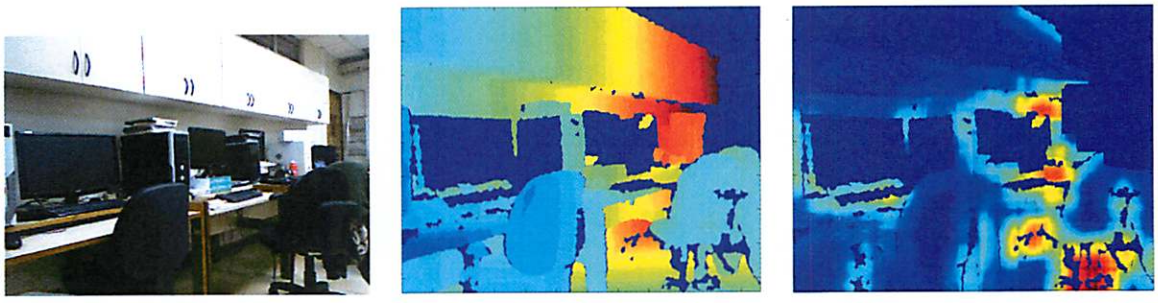
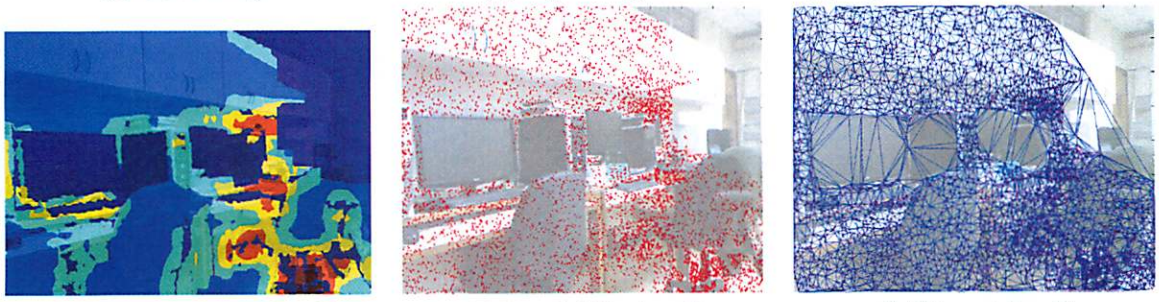


Figure 2: Flow Diagram of Current Triangulation Process



(a) Color Image (b) Depth Image D (c) High Frequency Response F



(d) After Histogramming F (e) Sampled Vertices V (f) Triangulation T

Figure 3: An example of the Triangulation Process. The idea is to start with a input depth image D and output a triangulation T which has a topology which is adaptive to the frequency content of the scene.

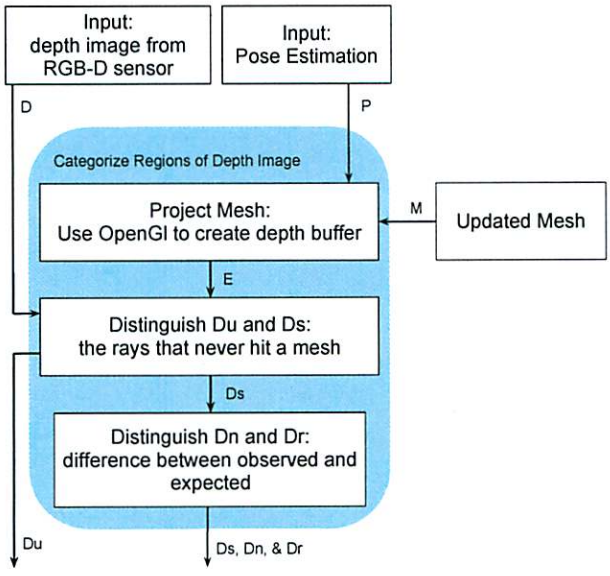
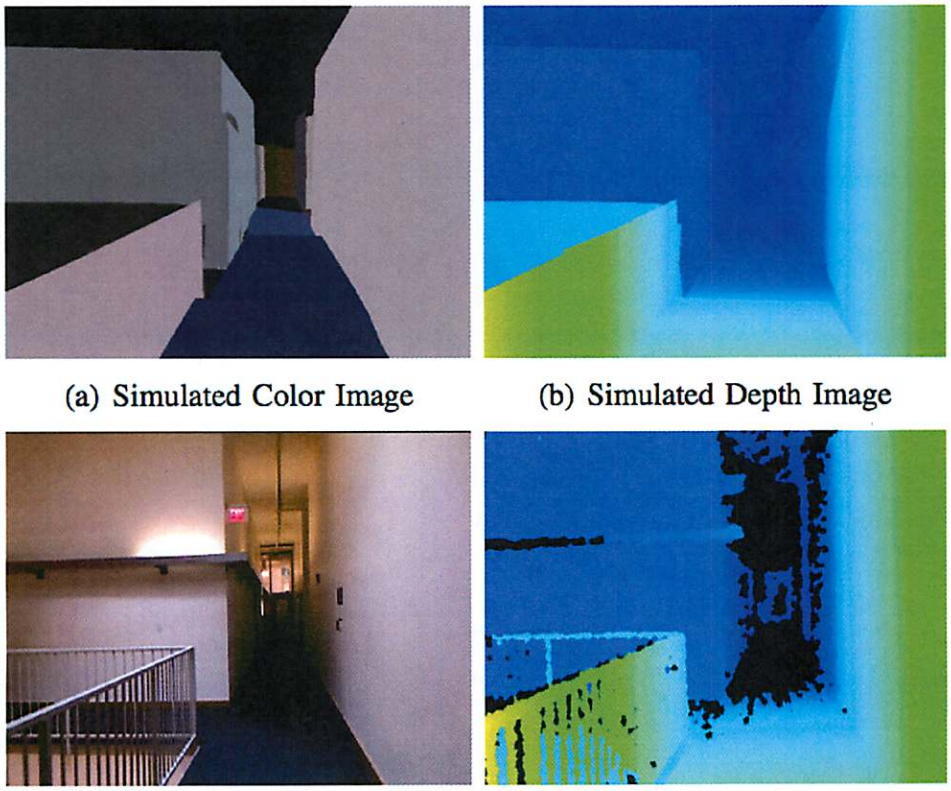


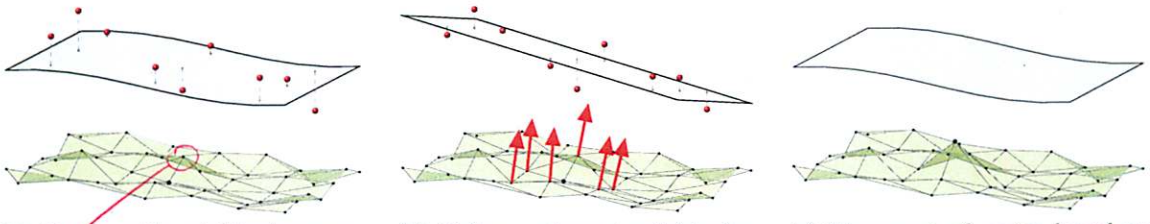
Figure 4: Flow Diagram of Categorize Measurements Process



(a) Simulated Color Image (b) Simulated Depth Image

(c) Measured Color Image (d) Measured Depth Image

Figure 5: Work from [3] which generates an expected depth image E in the same way as the proposed method.



(a) Vertex with neighboring measurements

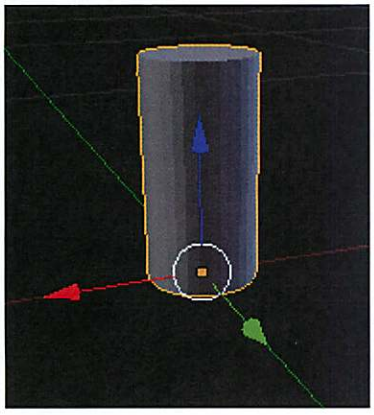
(b) Fitting a plane at neighboring vertices and through neighborhood of measurements

(c) Movement of vertex based on QEM

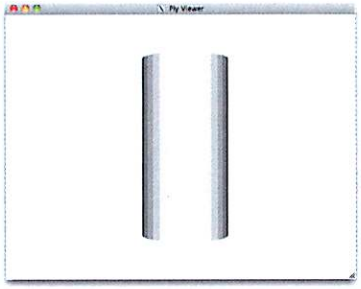
Figure 6: This shows the process of adapting a single vertex with new measurements.

Is That The one changing?

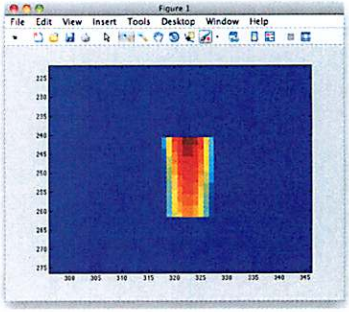
Need to show which location is changing



(a) Define geometry in Blender



(b) Render depth map in OpenGl



(c) Export depth image to Matlab

Figure 7: The steps of the simulation pipeline. The pipeline will allow us to simulate a RGB-D sensor viewing a known environment with a known pose.

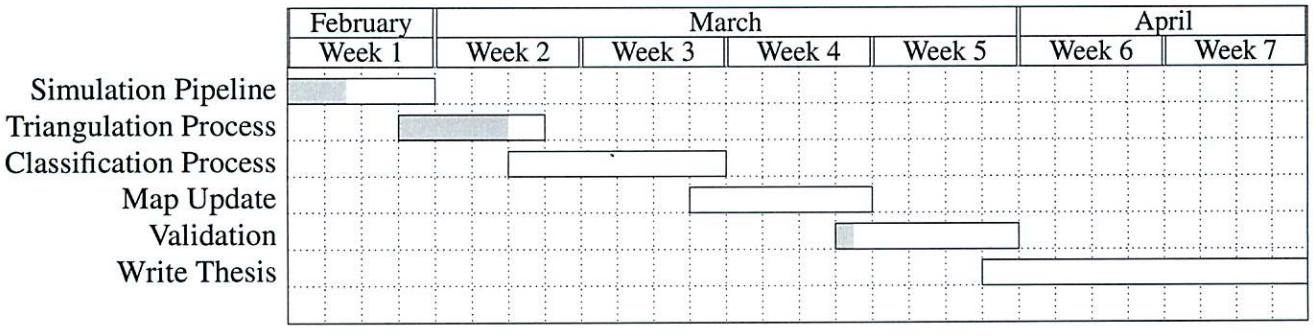


Figure 8: Gantt Chart