Mesh Addition Based on the Depth Image (MABDI)

Lucas Chavez

Dr. Lumia[1] Dr. Fierro[1]
Dr. Anderson[2]

[1]University of New Mexico

[2]Sandia National Laboratories

November 2016

This work was supported in part by:
Sandia National Laboratories under Purchase Order: 1179196 and NSF grant OISE #1131305.

2016-11-09

Mesh Addition Based on the Depth Image (MABDI)

Mesh Addition Based on the Depth Image (MABDI)

Lucas Chavez

Dr. Lumia[1] Dr. Fierro[1]
Dr. Anderson[2]

[1]University of New Mexico

[2]Sandia National Laboratories

November 2016

Outline

1. Introduction
2. Approach
   - Implementation
   - Software Design
3. Experimental Setup
   - Simulation Overview
   - Simulating a RGB-D Sensor
   - Sensor Path
   - Simulation Parameters
4. Results

Outline

1. Introduction
2. Approach
   - Implementation
   - Software Design
3. Experimental Setup
   - Simulation Overview
   - Simulating a RGB-D Sensor
   - Sensor Path
   - Simulation Parameters
4. Results

Outline

1. Introduction
2. Approach
   - Implementation
   - Software Design
3. Experimental Setup
   - Simulation Overview
   - Simulating a RGB-D Sensor
   - Sensor Path
   - Simulation Parameters
4. Results

Outline

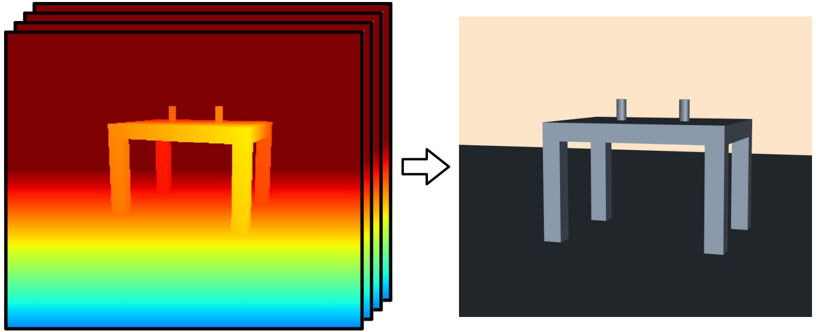
Introduction
Approach
Implementation
Software Design
Experimental Setup
Simulation Overview
Simulating a RGB-D Sensor
Sensor Path
Simulation Parameters
Results

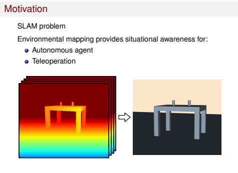
Motivation

SLAM problem

Environmental mapping provides situational awareness for:

- Autonomous agent
- Teleoperation





- This work is in the field of Environmental Planning
- In general Environmental Planning provides situational awareness
- Examples Autonomous - Path planning, obstacle avoidance, object manipulation
- Examples Teleoperation - Search and Rescue, Hazardous Environments

Outline

1 Introduction

2 Approach
- Implementation
- Software Design

3 Experimental Setup
- Simulation Overview
- Simulating a RGB-D Sensor
- Sensor Path
- Simulation Parameters

4 Results

Outline

Introduction
Approach
Implementation
Software Design
Experimental Setup
Simulation Overview
Simulating a RGB-D Sensor
Sensor Path
Simulation Parameters
Results

Variables

Description of the main variables

Variable Name	Description
D	Depth image from RGB-D sensor
P	Pose of the sensor
D_n	Parts of D that are *novel*
S	Novel surface generated from D_n
M	Global mesh

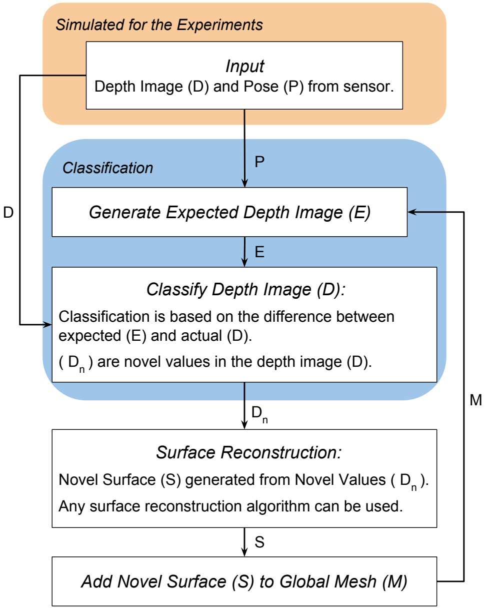
Variables

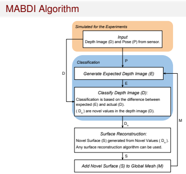
Description of the main variables

Variable Name	Description
D	Depth image from RGB-D sensor
P	Pose of the sensor
D_n	Parts of D that are *novel*
S	Novel surface generated from D_n
M	Global mesh

- Before we begin, let us get familiar with common variable names.

MABDI Algorithm





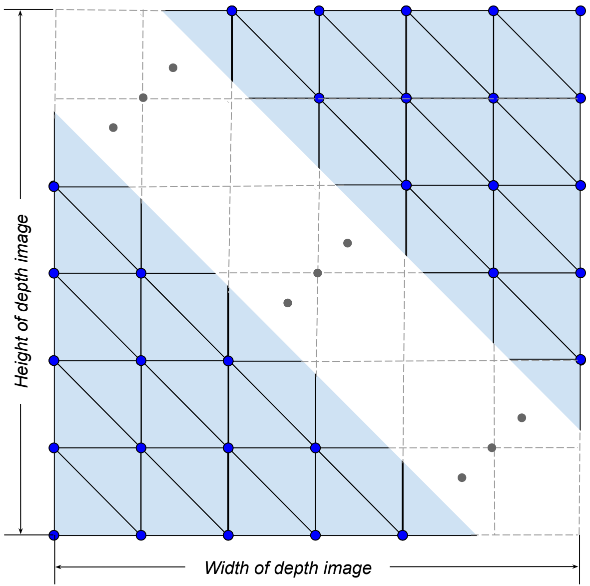
Input
- Has been simulated for this work
- Cover in detail in next section

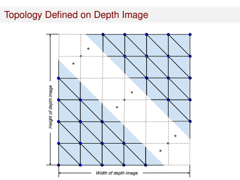
Generate Expected Depth Image
- What we expect to see from our sensor

Classify Depth Image
- Determine which points from D are novel

Surface Reconstruction
- Create a mesh structure from the novel points
- Cover in detail next

Topology Defined on Depth Image





- Mesh consists of vertices and elements
- - Vertices are points
- - Elements define connections between vertices
- Depth image
- - Not a set of unorganized points
- - Has structural information
- - This allows us to define a topology in 2D that is preserved when projected to 3D

Removing elements

Elements are removed from the S if they touch pixels from the sets:

- D_{known}
- $D_{boundary}$
- $D_{invalid}$

Removing elements

Elements are removed from the S if they touch pixels from the sets:

- D_{known}
- $D_{boundary}$
- $D_{invalid}$

Removing elements - D_{known}

$$D_n = |D - E| > threshold$$
$$D_{known} = D \setminus D_n$$

Removing elements - D_{known}

$D_n = |D - E| > threshold$
$D_{known} = D \setminus D_n$

Removing elements - $D_{boundary}$

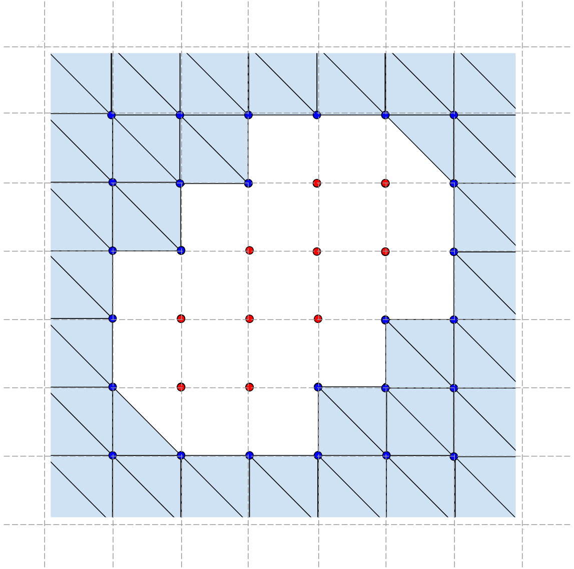
$$K = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$$
$$D_{boundary} = (D * K) > threshold$$

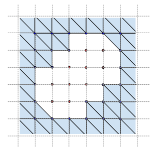
Removing elements - $D_{boundary}$

$K = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$
$D_{boundary} = (D * K) > threshold$

-

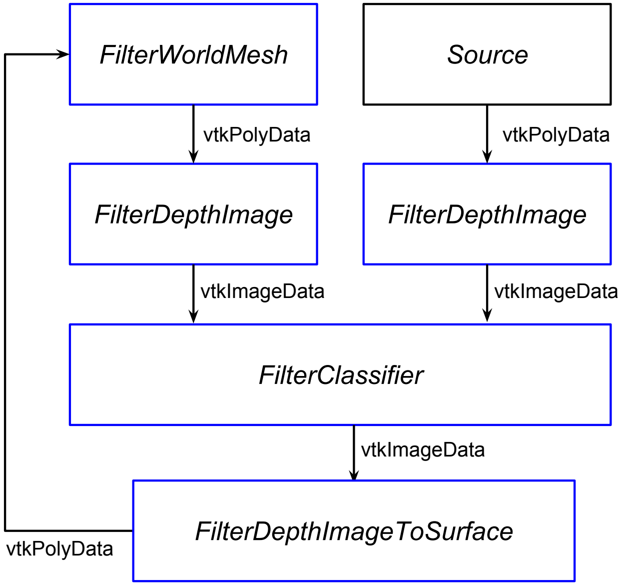
Removing elements

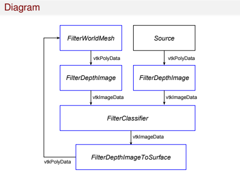




•

Diagram





- *Source* - Classes with the prefix Source define the environment that is used for the simulation and provide a mesh in the form of a vtkPolyData.
- *FilterDepthImage* - Render the incoming vtkPolyData in a window and output the depth buffer from the window as a vtkImageData. The output additionally has pose information of the sensor.
- *FilterClassifier* - Implements the true innovation of MABDI, i.e., takes the difference between the two incoming depth images (vtkImageData) and outputs a new depth image where the data that is not novel is marked to be thrown away.
- *FilterDepthImageToSurface* - Performs surface reconstruction on the novel points. The surface is output as a vtkPolyData.
- *FilterWorldMesh* - Here we simply append the incoming novel surface to a growing global mesh that is also output as a vtkPolyData.

Outline

1 Introduction

2 Approach
- Implementation
- Software Design

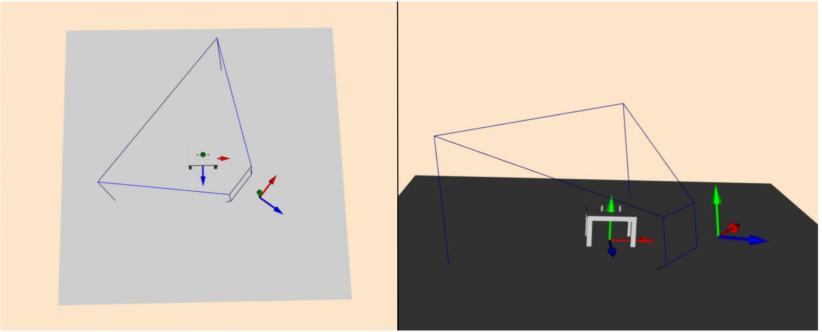
3 Experimental Setup
- Simulation Overview
- Simulating a RGB-D Sensor
- Sensor Path
- Simulation Parameters

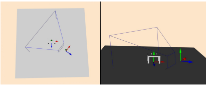
4 Results

Outline

Introduction
Approach
Implementation
Software Design
Experimental Setup
Simulation Overview
Simulating a RGB-D Sensor
Sensor Path
Simulation Parameters
Results

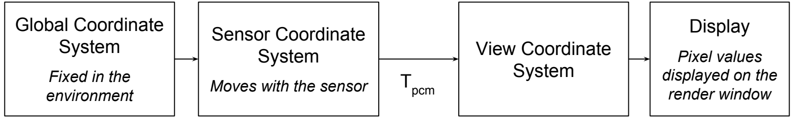
Overview

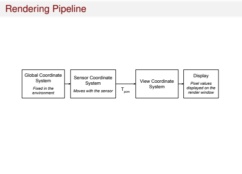




•

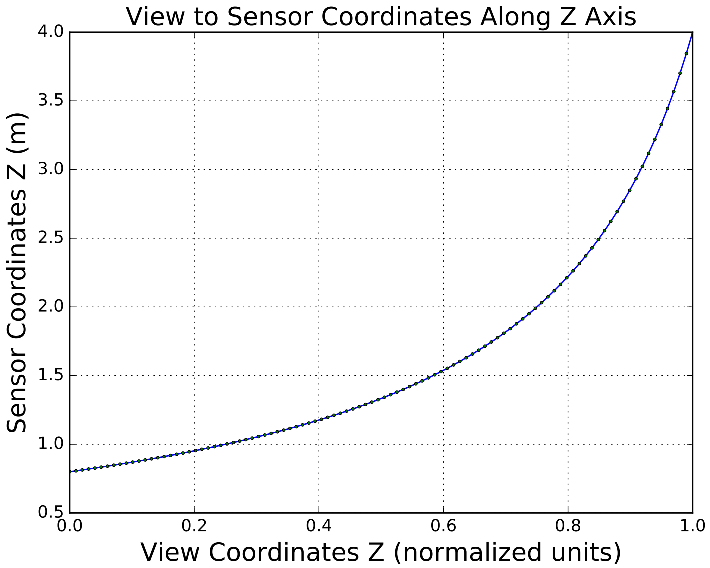
Rendering Pipeline

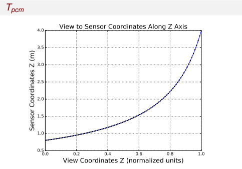




- Render pipeline: projects 3D global coordinates to 2D pixel coordinates

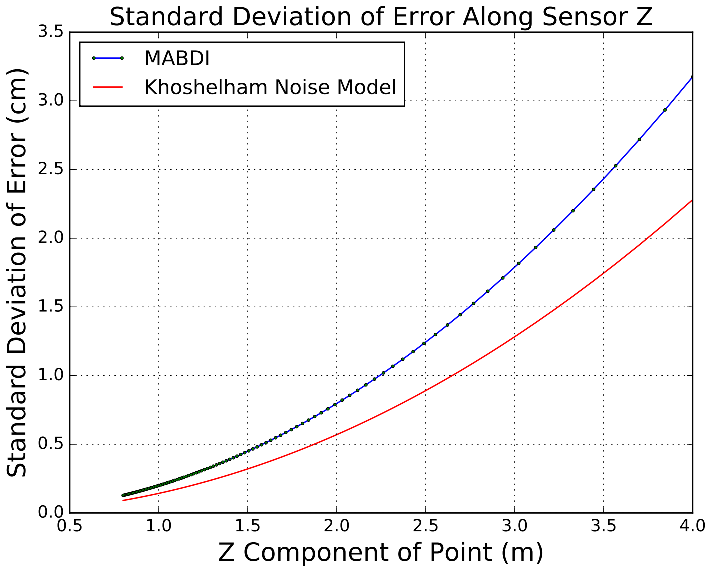
T_{pcm}



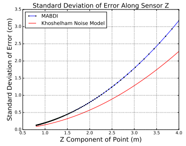


- The pinhole camera transformation, T_{pcm}, creates a non-linear relationship between values in the depth image and their corresponding location in the sensor's coordinate system.

Adding Noise

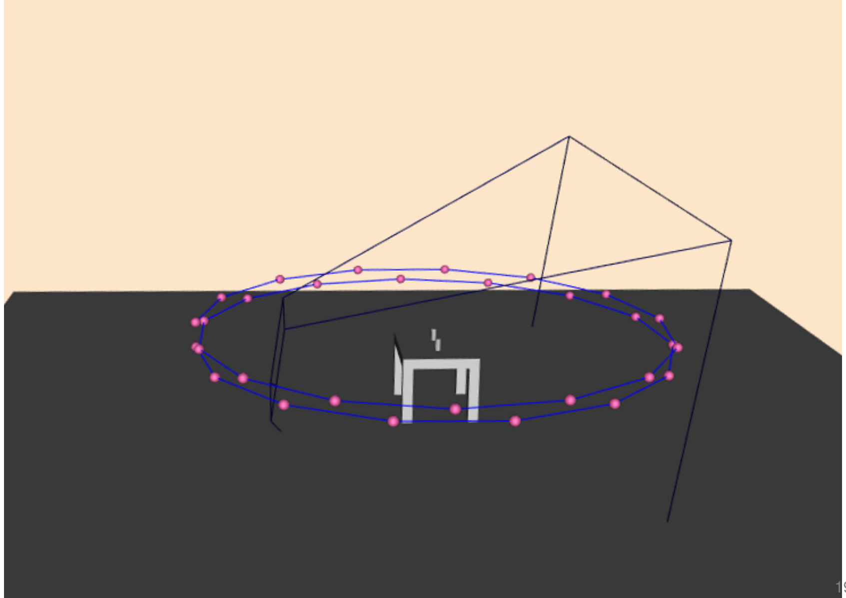


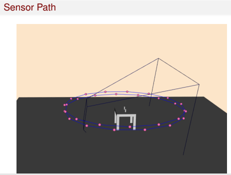
Adding Noise



- Comparison of standard deviation of the error used in the MABDI simulation and the error model from Khoshelham.

Sensor Path





- The blue line indicates the path and the pink points indicate where the sensor stops along the path. The path circles the objects in the environment twice. A helical path was chosen because it returns to a part of the environment that has already been mapped and is thus “known” to the algorithm. Also, because the path is a helix and not just a circle, the sensor views the environment from a slightly different position on each pass.

Simulation Parameters

	Environment	Noise	Dynamic	Iterations
Run 1	Table	False	False	30
Run 2	Bunnies	True	False	50
Run 3	Bunnies	True	True	50

Simulation Parameters

	Environment	Noise	Dynamic	Iterations
Run 1	Table	False	False	30
Run 2	Bunnies	True	False	50
Run 3	Bunnies	True	True	50

- Environment - This parameter specifies the environment used to generate the simulated depth images. *Table* is an environment consisting of a table and two cups placed on the table. The table is 1 meter tall. *Bunnies* is an environment consisting of three bunnies that are around 1.5 meters tall. These bunnies are created using the Stanford Bunny a well known data set in computer graphics.
- Noise - If true, adds noise to the depth image of the simulated sensor.
- Dynamic - If true, adds an object during the simulation. In the case of this analysis, a third bunny is added half-way through the simulation.
- Iterations - The number of times MABDI will run. This number is equal to the number of stops the sensor makes along the path because every time the sensor stops MABDI is run to update the global mesh.

Outline

1 Introduction

2 Approach
- Implementation
- Software Design

3 Experimental Setup
- Simulation Overview
- Simulating a RGB-D Sensor
- Sensor Path
- Simulation Parameters

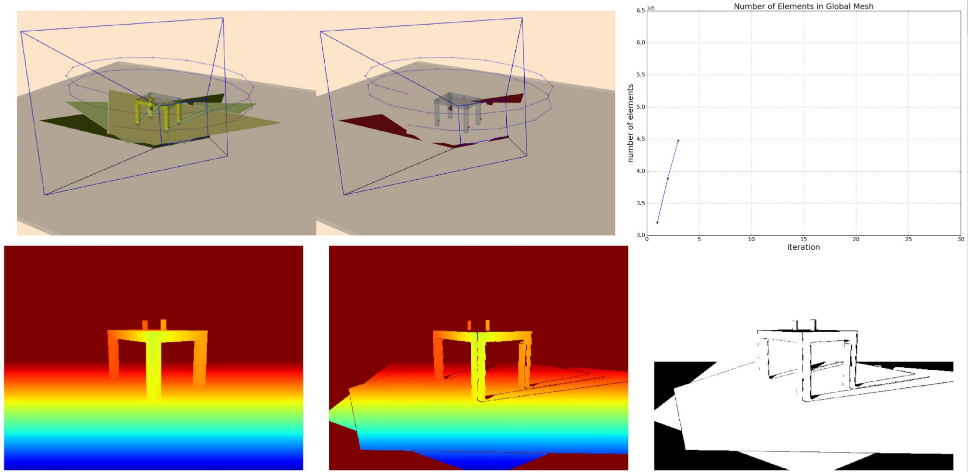
4 Results

Outline

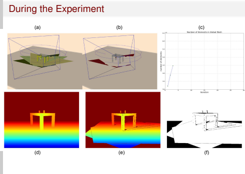
Introduction
Approach
- Implementation
- Software Design
Experimental Setup
- Simulation Overview
- Simulating a RGB-D Sensor
- Sensor Path
- Simulation Parameters
Results

During the Experiment

(a) (b) (c)

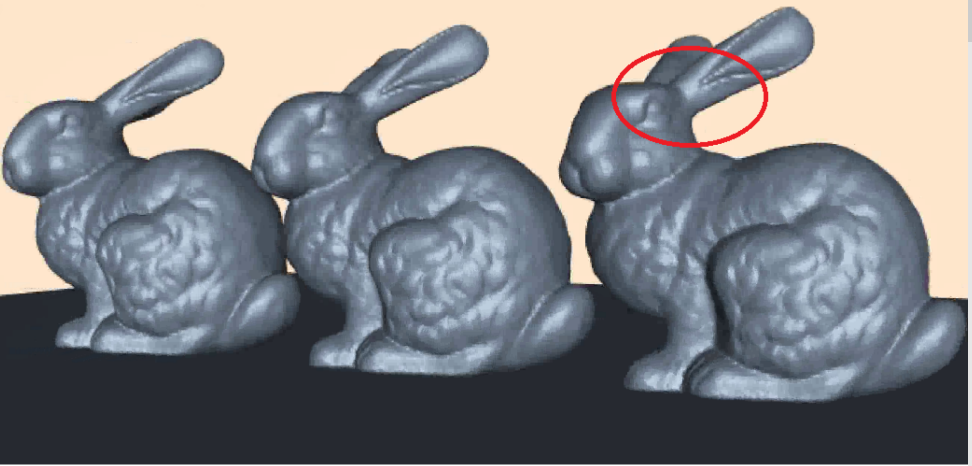


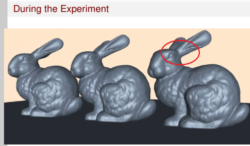
(d) (e) (f)



- Input - **??**(d) shows the depth image *D* generated from the simulated sensor. **??**(a) shows us two important aspects to consider about *D*. First, the pose *P* of the sensor is shown by looking at the sensor's view frustum, indicated by the blue wireframe. Second, the only environmental information used to generate the depth image is shown in light gray.
- Generate Expected Depth Image (*E*) - **??**(e) shows the expected depth image *E*. **??**(a) also shows us two important aspects to consider about *E*. First, the same pose *P* is used to create both *D* and *E* (as indicated by the blue wire frame). Second, the only environmental information used to create *E* is the yellow and light green parts of *M* because that is the only information *M* contains *during* iteration 3.
- Classify Depth Image (*D*) - **??**(f) visualizes the classification process. More specifically, it shows the points as expressed in Equation **??** in white (*D*...). **??**(f) is important for

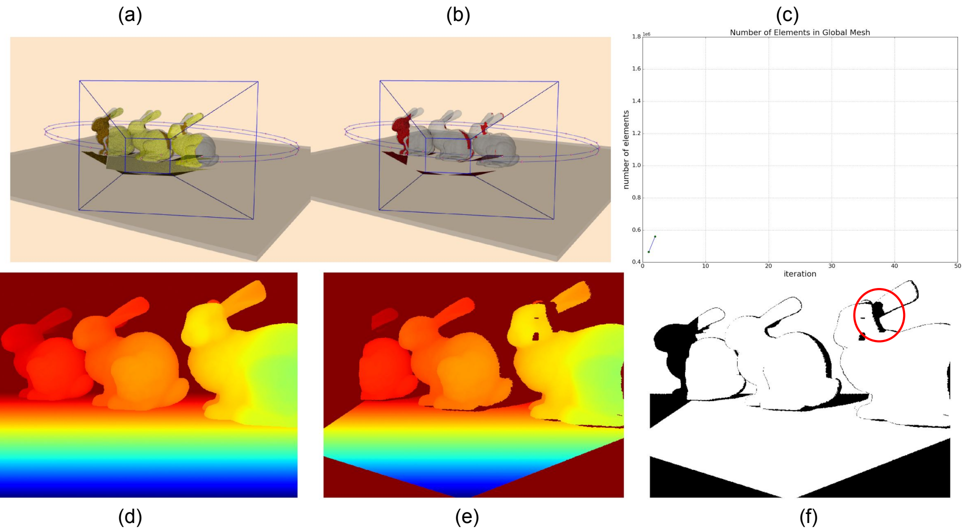
During the Experiment

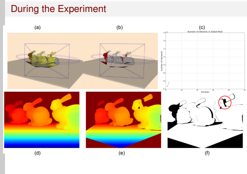




- Novel portion of the environment.
- To be discussed during Experiment 2.

During the Experiment

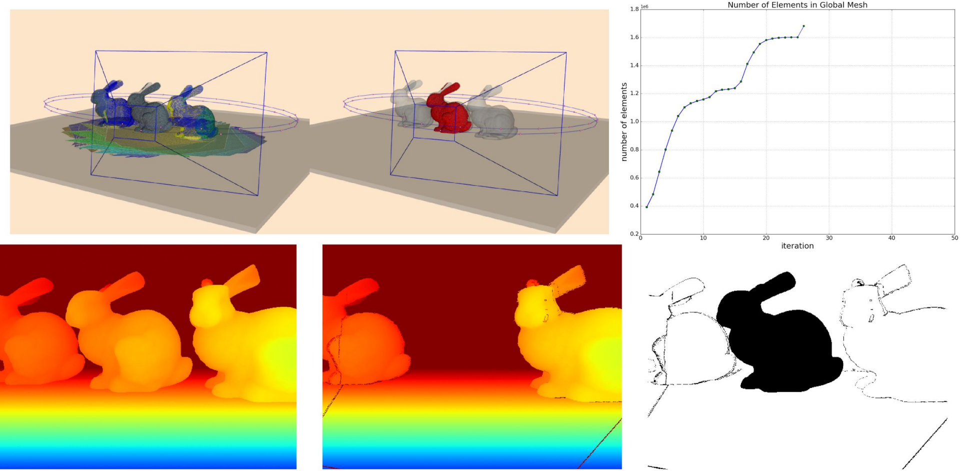




- **??**(a) shows the global mesh M. The yellow portion of the mesh constitutes the entirety of M after the first iteration. We can see the novel portion of the environment was not represented in M after the first iteration due to occlusion.
- **??**(d) shows the depth image D from the new sensor pose P. We can see the novel portion can be seen by the sensor on this iteration.
- **??**(e) shows the expected depth image E. During the second iteration M consists of only the yellow portion shown in **??**(a) consequently, E does not show any points in the area corresponding to the novel portion of the environment.
- **??**(f) shows the classification process successfully identifying points in D that correspond to the novel portion as indeed novel. In the figure the points are highlighted by a red circle.
- **??**(b) shows the novel surface S now represents the novel

During the Experiment

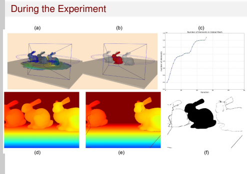
(a) (b) (c)



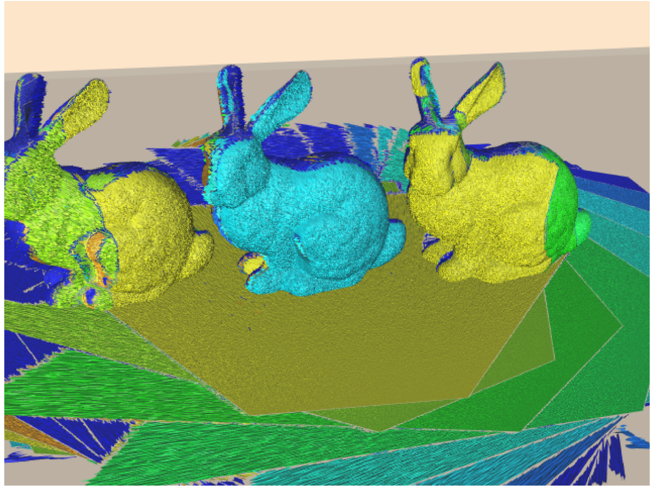
(d) (e) (f)

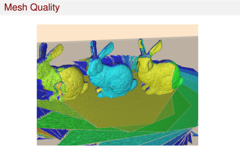
2016-11-09

Mesh Addition Based on the Depth Image (MABDI)
└Results
└During the Experiment


During the Experiment

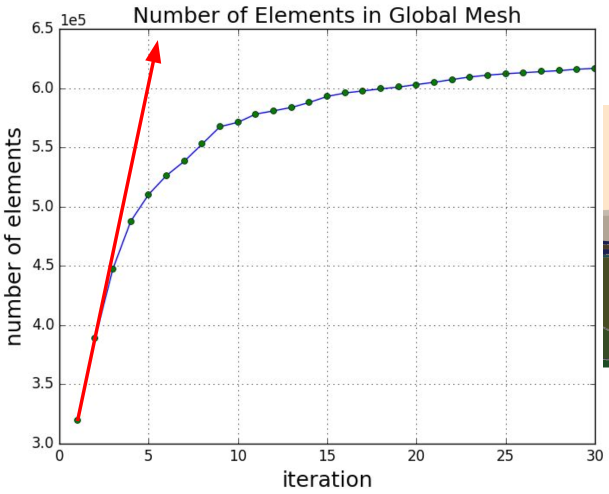
Mesh Quality

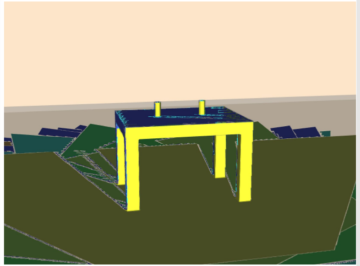


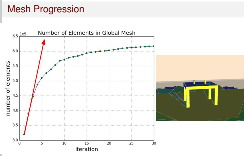


- There are gaps in the mesh that occur typically along the boundaries of where the novel surface S is appended to the global mesh M. This behavior is common for Surface Reconstruction methods as those discussed in Section **??**. Algorithms exist for merging these gaps as a post processing step such as Turk's Zippered Polygon Meshes [**?**]. The aforementioned methods are typical for single object reconstruction. Traditional mesh-based environmental mapping algorithms simply append overlapping layers of mesh resulting in no gaps but a heavily redundant representation with a high memory cost.
- The mesh is noisy. This noisiness is due to the simplicity of our implementation's surface reconstruction method as discussed in Section **??**. Our method simply connects neighboring points in the point cloud without additional steps such as Laplacian smoothing [**?**]. Our reconstruction method was sufficient for

Mesh Progression

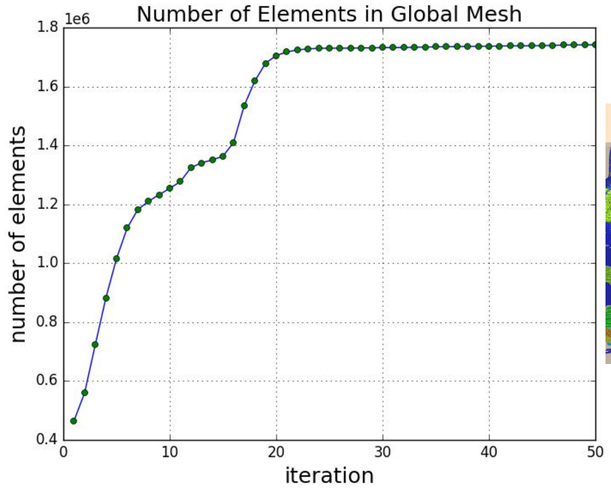


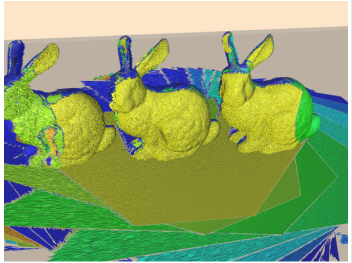


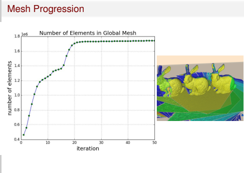


- Figure **??** shows the resultant mesh and mesh progression for the first experiment. The plot highlights the major difference between MABDI and traditional mesh-based environmental mapping methods. Traditional methods would have a plot similar to that indicated by the red arrow on the graph because these methods have no ability to identify or remove redundant mesh elements. Due to MABDI's algorithmic design, MABDI has the intrinsic ability to identify points in the depth image corresponding to parts of the environment that are already known by the global mesh M. MABDI then simply does not use those points for surface reconstruction and consequently does not create redundant mesh elements. For this reason, the number of elements in M levels off as the environment becomes more known.

Mesh Progression

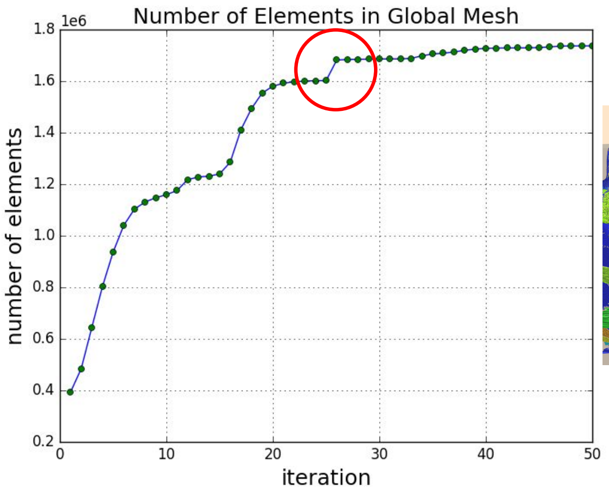


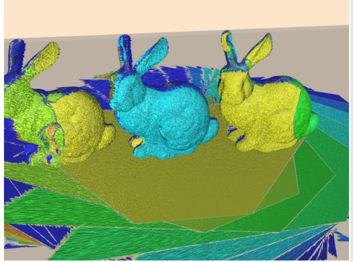




- Figure **??** shows us the resultant mesh after the second experiment. Here we can see that MABDI is reactive to the environment. In the preceding experiment, the environment was symmetrical. In this experiment, the environment is not symmetrical and we can see the effects by looking at the progression of the global mesh M. First let us note that the sensor circles the objects twice during the experiment and in total travels 720° during the 50 iterations. We notice when the sensor gets to 90° (around iteration 7) the number of elements begins to level off and then increases again as the sensor travel to 270° (around iteration 19). This behavior occurs because the information rich perspectives of the environment occur at 0° and 180°. There is less for the sensor to look at when viewing the environment from the sides. In this way, MABDI is reactive as the sensor moves to parts of the environment that are rich in information. Consequently, the mesh grows rapidly based on the

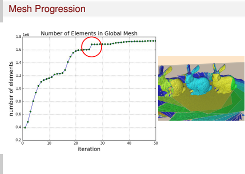
Mesh Progression





2016-11-09

Mesh Addition Based on the Depth Image (MABDI)
└Results
└Mesh Progression



- Figure **??** shows us the resultant mesh after the third experiment. In this experiment the middle bunny was added during the twenty-sixth iteration. This object addition had two effects on the global mesh. First, it created a sudden jump in the plot as highlighted by the red circle. Second, the middle bunny is colored blue in the resultant mesh, signifying that it was added to M during a different iteration than the bunnies on the left and the right. Both of these effects indicate that MABDI was able to successfully identify the new bunny as novel and incorporate the bunny in to the global mesh within one iteration.

Outline

1 Introduction

2 Approach
- Implementation
- Software Design

3 Experimental Setup
- Simulation Overview
- Simulating a RGB-D Sensor
- Sensor Path
- Simulation Parameters

4 Results

Outline

Introduction
Approach
Implementation
Software Design
Experimental Setup
Simulation Overview
Simulating a RGB-D Sensor
Sensor Path
Simulation Parameters
Results

Motivation

The goal of MABDI is to identify data from the sensor that has not yet been represented in the map and use this data to add to the map. MABDI does this by leveraging the difference between what we are actually seeing and what we expect to see. MABDI can work in conjunction with any current mesh-based surface reconstruction algorithms, and can be thought of as a general means to provide introspection to those types of reconstruction methods.

The MABDI implementation was able to successfully perform in a realistic simulation environment. The results show how novel sensor data was successfully classified and used to add to the global mesh. Also, the MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor. This performance means that it is capable of real-world applications. Currently MABDI is only designed to handle object addition, but the idea can be extended to handle both object addition and removal as discussed in Section **??**. This would give the

Motivation

The goal of MABDI is to identify data from the sensor that has not yet been represented in the map and use this data to add to the map. MABDI does this by leveraging the difference between what we are actually seeing and what we expect to see. MABDI can work in conjunction with any current mesh-based surface reconstruction algorithms, and can be thought of as a general means to provide introspection to those types of reconstruction methods.
The MABDI implementation was able to successfully perform in a realistic simulation environment. The results show how novel sensor data was successfully classified and used to add to the global mesh. Also, the MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor. This performance means that it is capable of real-world applications. Currently MABDI is only designed to handle object addition, but the idea can be extended to handle both object addition and removal as discussed in Section ??. This would give the