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.

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

Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

Outline

Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

Overview

Motivation is to provide situation awareness

- Autonomous agents
- Teleoperation

Overview
Motivation is to provide situation awareness
Autonomous agents
Teleoperation

- Examples Autonomous
- - Path planning, obstacle avoidance, object manipulation
- Examples Teleoperation
- - Search and Rescue, Hazardous Environments

Overview

In the literature referred to as the SLAM problem

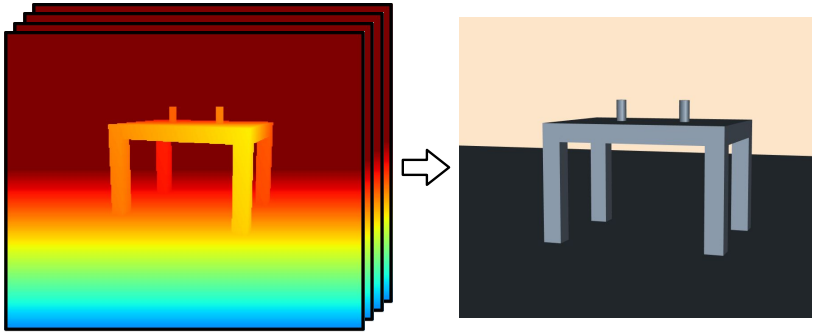
- Simultaneous Localization and Mapping
- Environmental Mapping
- Research began around 1987
- Sensing and computing technology

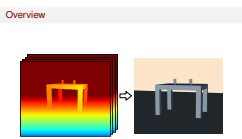
Overview

In the literature referred to as the SLAM problem
- Simultaneous Localization and Mapping
- Environmental Mapping
- Research began around 1987
- Sensing and computing technology

- Most work in this area describe the SLAM problem
- - Simultaneously locate the robot in the environment as well as map the environment
- Environmental Mapping
- - Deals specifically with the mapping part of the SLAM problem
- - This work is a contribution to Environmental Mapping
- 1987
- - Work in this area has been going on for more than 20 years
- Recent work
- - Rich and dense maps of the environment
- - Fueled by recent advances in sensing and computational power

Overview





- For this work
- Goal is to transform depth images into a mesh representation
- Depth images
- - each pixel represents distance from sensor instead of color
- Mesh representation
- - Vertices and elements
- - 3D points and the connects between those points

RBG-D Sensor

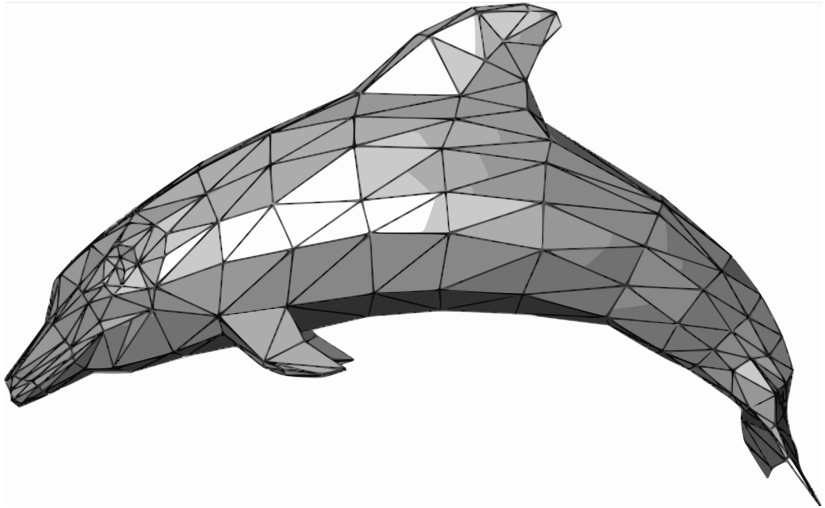
- 30 frames per second
- D - 9 million pixel values per second
- Algorithms must handle a high rate of data

RBG-D Sensor
30 frames per second
D - 9 million pixel values per second
Algorithms must handle a high rate of data

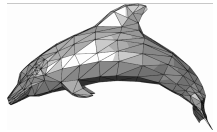
- Kinect
- First affordable sensor to provide
- high resolution spatial information

- High rate of data
- Algorithms must take this as a design consideration

Mesh



Mesh



- There are different types of maps
- Mesh is the map type chosen for this work

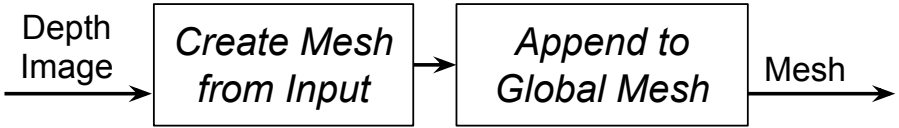
Mesh

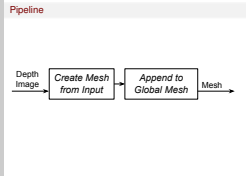
- Supported
- Computationally Inexpensive
- Low Memory Requirement

Mesh
Supported
Computationally Inexpensive
Low Memory Requirement

- Supported
- - Is there software, tools, research, algorithms, etc., for this type of map?
- Computationally Inexpensive
- - Can the algorithms run quickly on low cost computers (rather than specialized hardware)?
- Low Memory Requirement
- - Can the algorithms run on hardware with a standard amount of RAM?

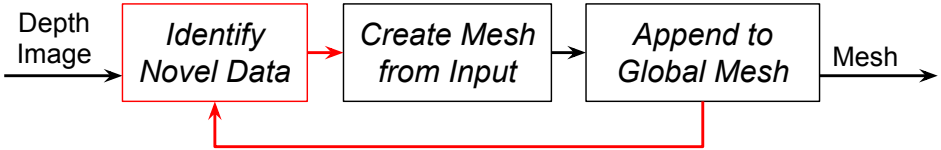
Pipeline

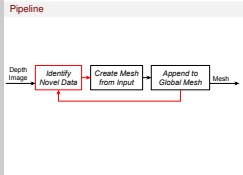




- Traditional Mesh-Based Mapping Methods
- - Take incoming data
- - Generate a mesh structure
- - Append to growing global mesh structure

Pipeline





- MABDI
- - Takes what we already know
- (the global mesh structure)
- - Uses it to throwaway points in the data that are redundant
- Ability to classify incoming data and only use what is novel
- is MABDI's contribution

Contribution

MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.

Contribution

MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.

Outline

Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

Important Variables

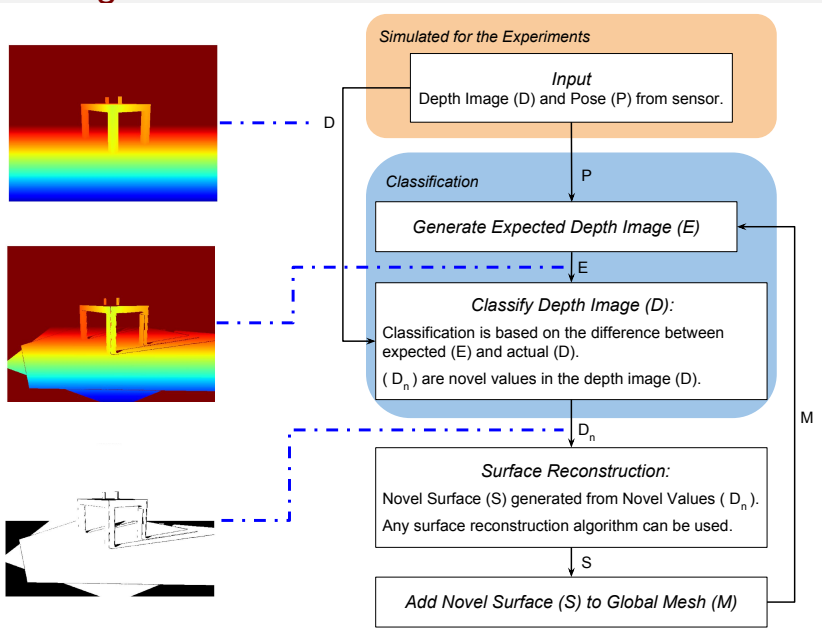
Description of the main variables

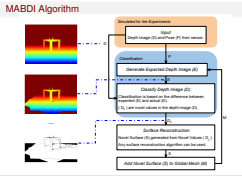
Variable Name	Description
D	Depth image from RGB-D sensor
P	Pose of the sensor
D_n	Parts of D that are *novel*
S	Novel surface generated from D_n
M	Global mesh

Important 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

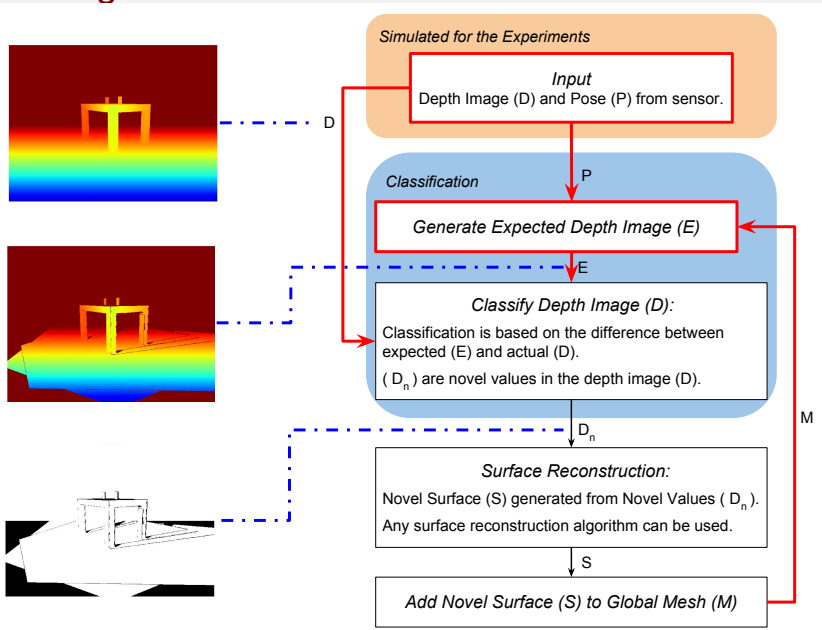
- D - an image
- P - Describes position and orientation of sensor
- D_n - Subset of point in D that have been labeled as novel
- S - Mesh structure. List of vertices and elements.
- Vertices are points and elements define connections between vertices

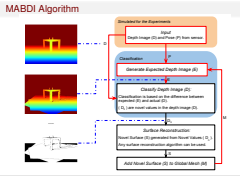
MABDI Algorithm





MABDI Algorithm





The MABDI algorithm

- Orange -
- Input to the algorithm. This has been simulated for this work.
- Classification -
- What sets MABDI apart from traditional mesh-based mapping algorithms.
- Allows us to classify data before it is added to the Global Mesh.

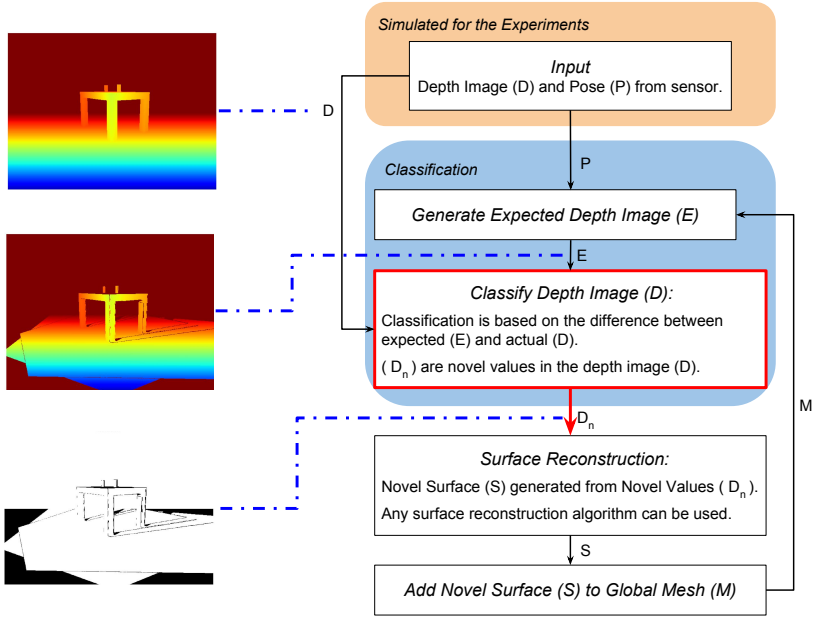
Input

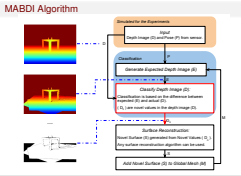
- Has been simulated in this work.
- We will cover simulation process in detail.

Generate Expected Depth Image

- Takes the global mesh (what we know about the environment)

MABDI Algorithm

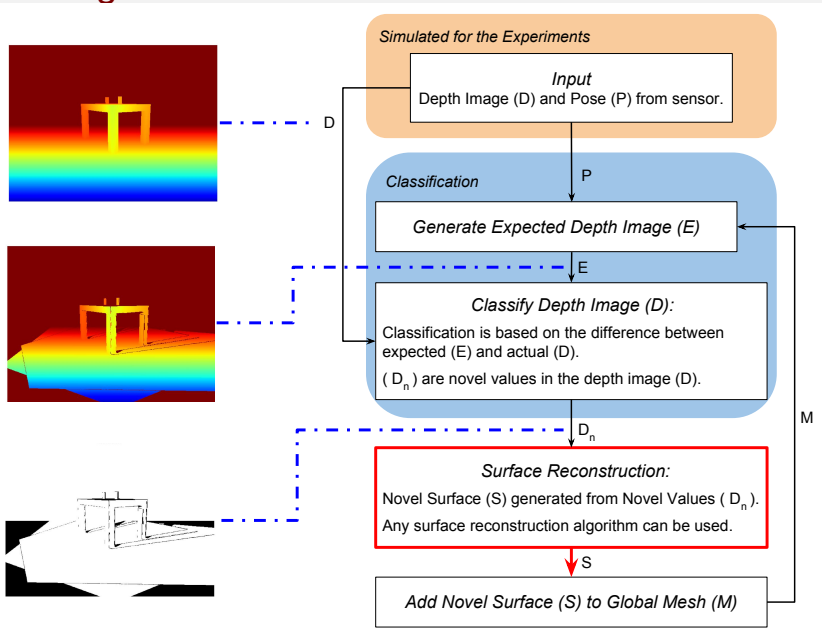


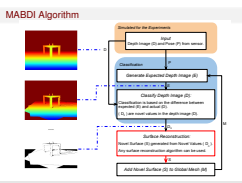


Classify Depth Image

- This is the heart of MABDI
- and is MABDI's contribution to the state-of-the-art
- Determine which points from D are novel.
- (From a new part of the environment that has not been seen before)
- Taking the absolute difference between *E* and *D* and thresholding.
- (point to equation)
- If the differences are small, those points are thrown away.
- If the differences are large, those points are kept.
- If the difference is large, the measurements are coming from a part of the environment that has not been seen before.

MABDI Algorithm

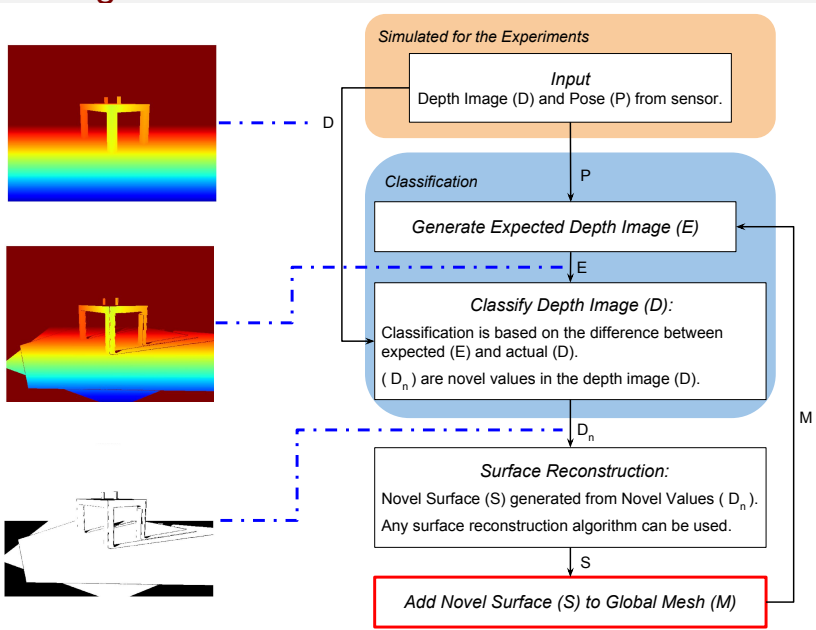


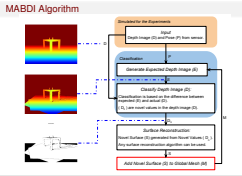


Surface Reconstruction

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

MABDI Algorithm





- Add Novel Surface to Global Mesh
- Append surface to the global mesh
- that is continuously being updated

Implementation Specific Details

Initial Mesh

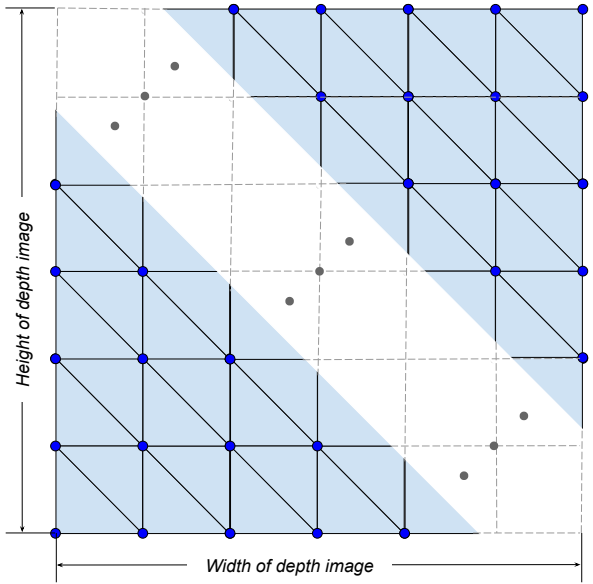
- Surface Reconstruction component is responsible for creating S from D_n
- Our Method:
- Define topology in 2D, on the depth image
- Project to 3D
- Remove elements

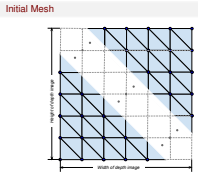
Initial Mesh
Surface Reconstruction component is responsible for creating S from D_n
Our Method:
Define topology in 2D, on the depth image
Project to 3D
Remove elements

- Responsible for creating a surface S from the novel points D_n
- S is a mesh data structure that consists of a list of vertices and elements
- - Vertices are points
- - Elements define connections between vertices

- Generate initial mesh with all points in D
- 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

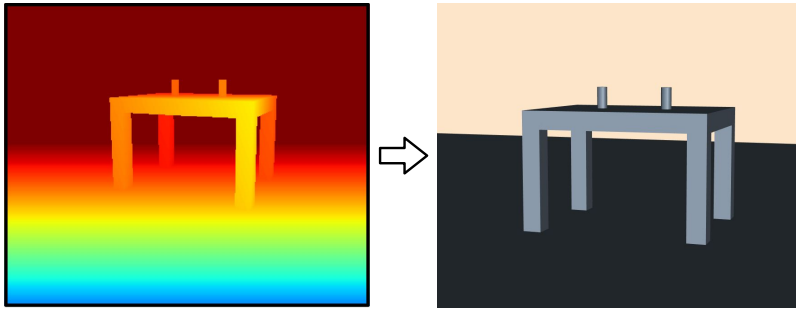
Initial Mesh



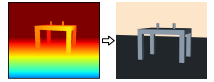


- Imagine this is the depth image
- - Every blue point is a pixel in the depth image
- - Corresponds to 3D point in space
- Depth image
- - Not a set of unorganized points
- - Has structural information
- - This allows us to define a topology in 2D that is preserved when projected to 3D
- We then take every blue dot and project them into 3D space
- preserving the connections between vertices

Initial Mesh



Initial Mesh



- Imagine mesh being defines using every pixel in depth image
- Then projected to 3D space
- - Ignore the background for now
- Note there will be no surface
- behind the cup,
- under the table,
- anywhere on the floor that the sensor doesn't see

Removing elements

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

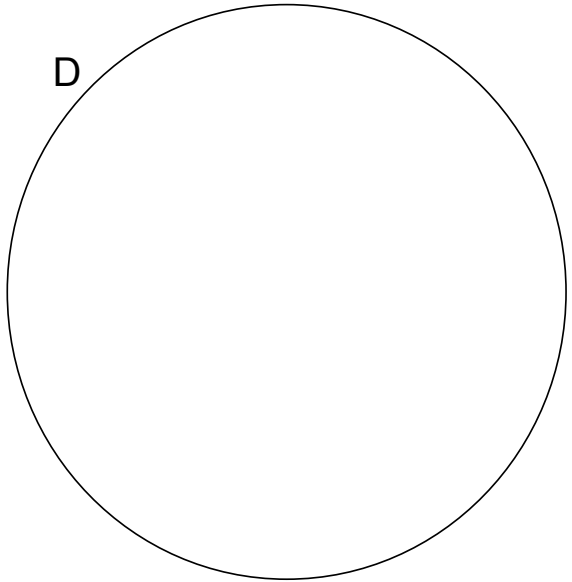
- D_{known}
- $D_{boundary}$
- $D_{invalid}$

Removing elements

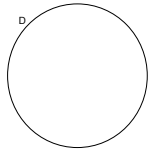
Elements are removed from the S if they touch pixels from the sets:

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

Removing elements

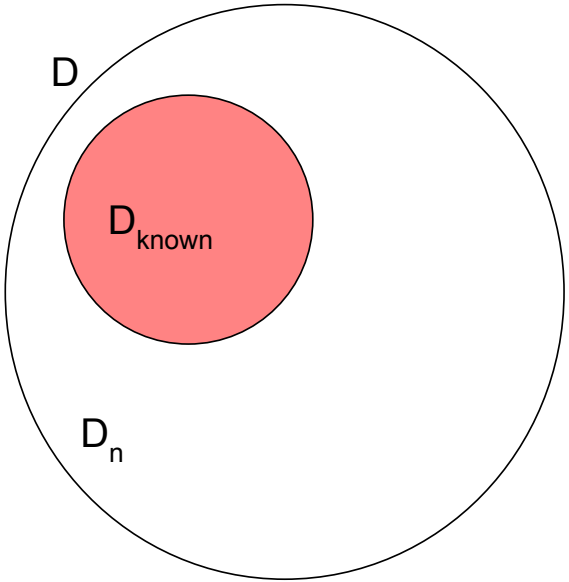


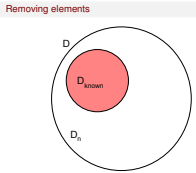
Removing elements



- Inside the circle represents the set
- containing every point from D

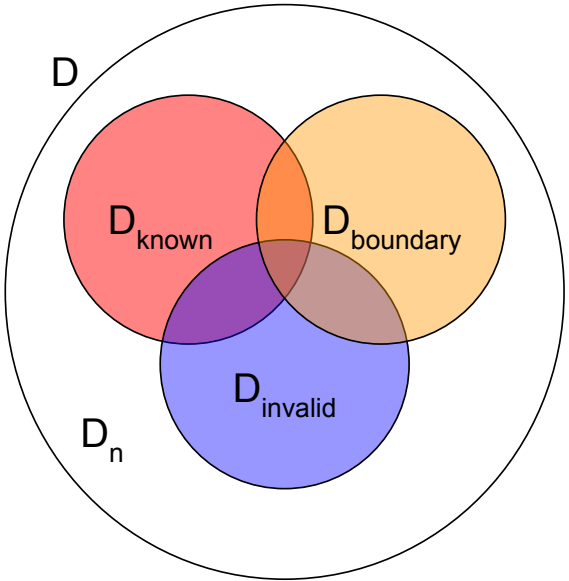
Removing elements



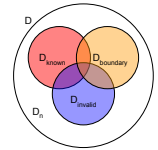


- D_n (novel) is everything that the categorization process said is novel.
- (point to equation)

Removing elements

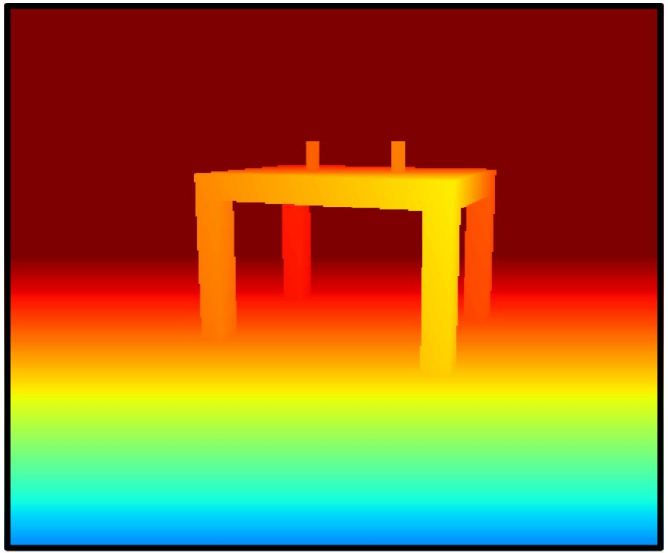


Removing elements

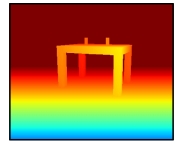


- We define two additional set of points to be thrown away.
- $D_{boundary}$
- - remove elements defined by points that lie on completely different surfaces
- $D_{invalid}$
- - elements that are out of range of the sensor

Removing elements



Removing elements



- We define two additional set of points to be thrown away.
- $D_{boundary}$
- - (point out neighboring pixels on leg and floor)
- $D_{invalid}$
- - (point out background)

Removing elements - D_{known}

$$D_n = |D - E| > threshold$$

$$D_{known} = D \setminus D_n$$

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

- Define novel and known points formally
- D_n - Same equation as discussed "Classify Depth Image" component
- D_{known}
- - All the points that not novel
- - Those points that have a small difference

Removing elements - $D_{boundary}$

$$K = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$$

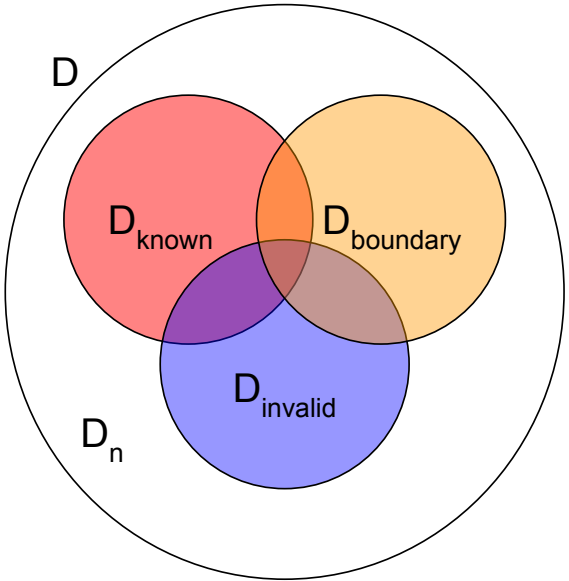
$$D_{boundary} = (D * K) > threshold$$

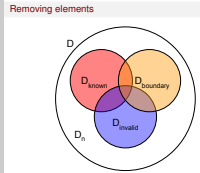
Removing elements - $D_{boundary}$
$K = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$
$D_{boundary} = (D * K) > threshold$

- Lie on different surfaces
- Like the pixel neighbors floor and leg that we discussed

- Two dimensional, differencing convolution filter is passed over D
- This filter has a magnified response at points where the difference between neighboring pixels is large
- Remembering pixel values signify depth

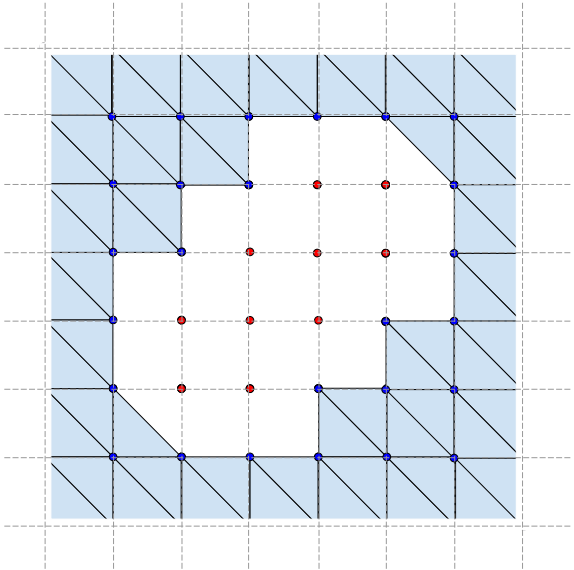
Removing elements



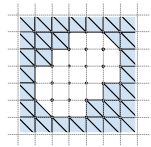


- Each colored circle represents points that we are going to remove

Removing elements

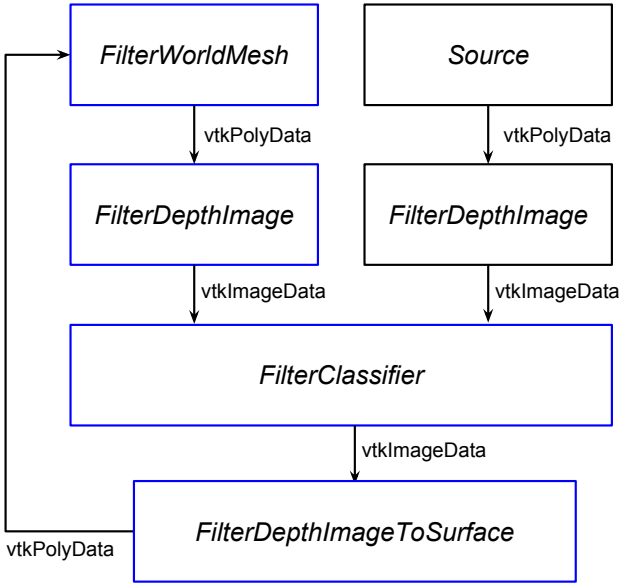


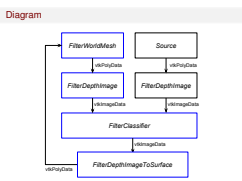
Removing elements



- Points from the colored circle are represented as the red points
- Technically, these points have already been projected to 3D
- but this is the best way to visualize this concept

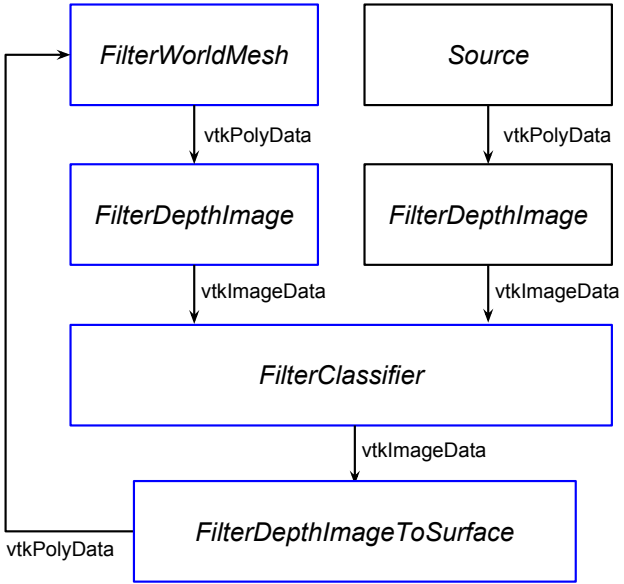
Diagram

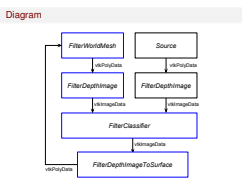




- In blue are the core components
- In black components for the simulation of a real environment

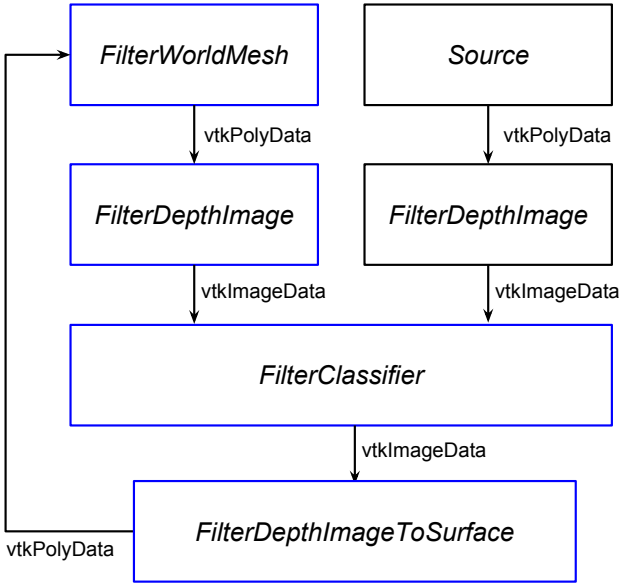
Diagram

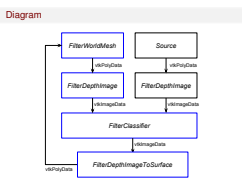




- *Source*
- - Define the environment that is used for the
- *FilterDepthImage*
- - Render the incoming vtkPolyData in a window and
- - output the depth buffer from the window as a vtkImageData
- - output has pose information of the sensor.

Diagram



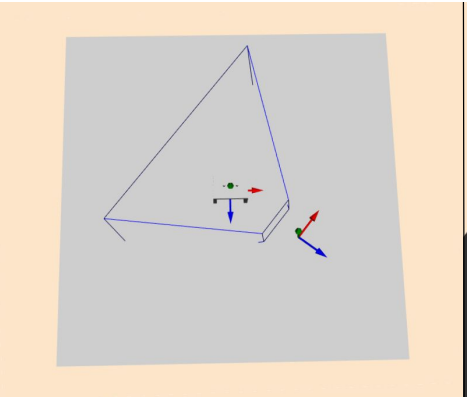


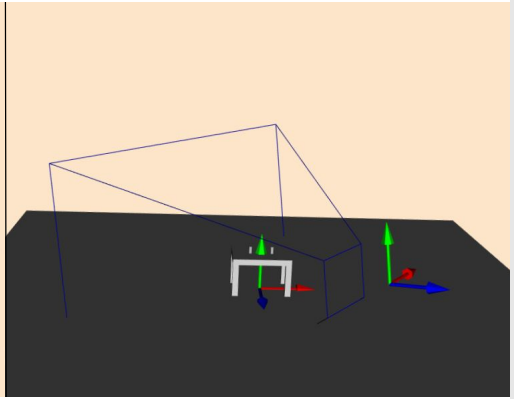
- *FilterClassifier*
- - Implements the true innovation of MABDI
- - take difference of actual and expected
- - outputs a new depth image where the data that is not novel is marked to be thrown away
- *FilterDepthImageToSurface*
- - Performs surface reconstruction
- - surface is output as a vtkPolyData.
- *FilterWorldMesh*
- - Here we simply append the incoming novel surface to a growing global mesh that is also output as a vtkPolyData

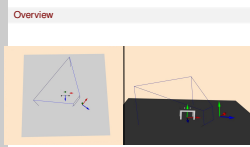
Outline

Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

Overview







- Third person view of simulation environment

  Two main coordinate systems
- A coordinate system fixed to the environment called the global coordinate system
- One attached the origin of the sensor's viewing frustum

  red, green, and blue arrows represent the x, y, and z axis respectively

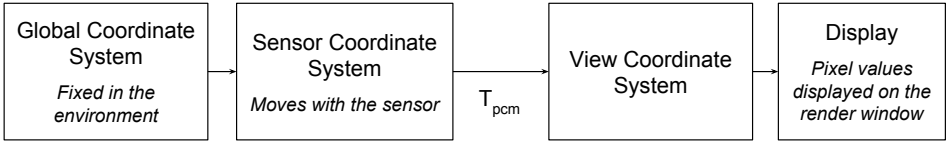
Simulating a RGB-D Sensor

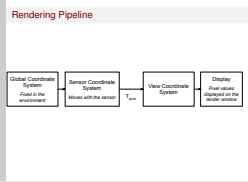
- Adding noise to the depth image
- Sensor path

Simulating a RGB-D Sensor
Adding noise to the depth image
Sensor path

- First we will discuss
- Adding noise to the depth image to simulate a depth sensor in a real environment
- Second we will discuss
- The path the sensor takes in each experiment

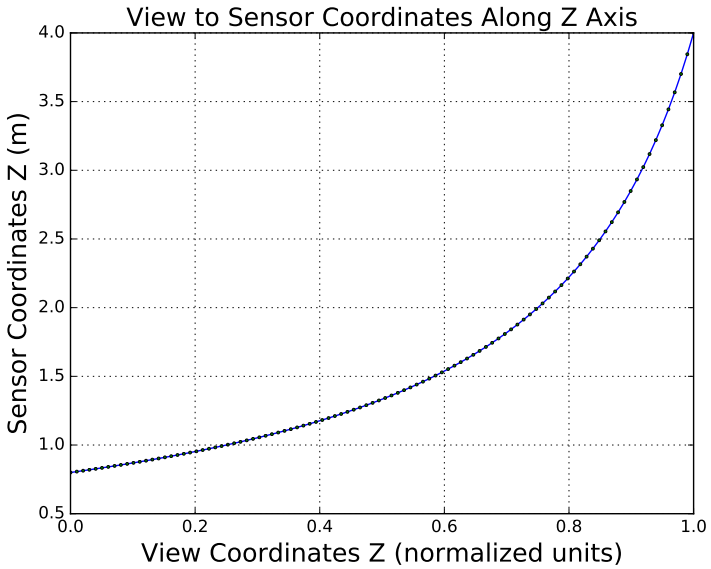
Rendering Pipeline

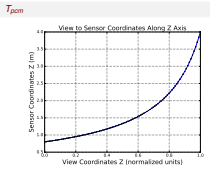




- Pinhole camera model
- transforms geometry in the sensor's coordinate system to homogenous coordinates
- z-component of the homogenous coordinates is what defines the depth image

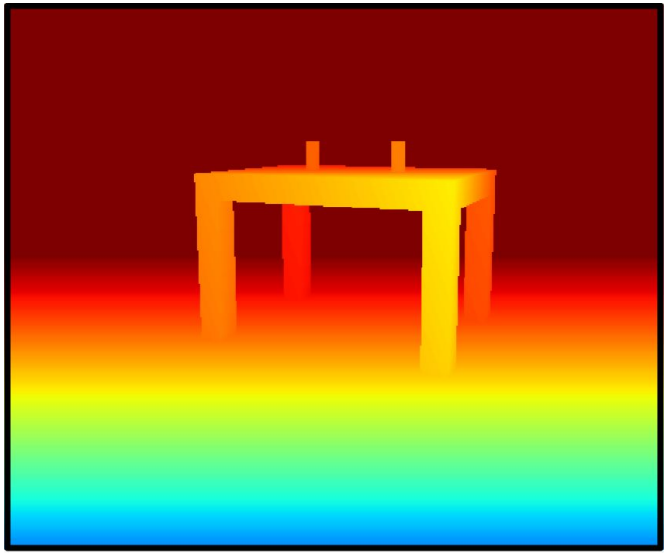
T_{pcm}



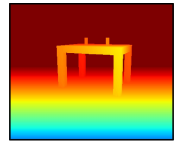


- The pinhole camera transformation, T_{pcm},
- Creates a non-linear relationship between
- - values in the depth image
- - and their corresponding location in the sensor's coordinate system
- Graph was created by:
- - Taking center pixel in depth image
- - Varying value 0-1 and projecting to camera's coordinate system

Adding Noise



Adding Noise



- The depth image

Adding Noise

$$D_{noisy}(i,j) = D(i,j) + \mathcal{N}(\mu{=}0, \sigma{=}0.002)$$

Adding Noise
$D_{noisy}(i,j) = D(i,j) + \mathcal{N}(\mu{=}0, \sigma{=}0.002)$

- We sample a normal distribution and add the value to each pixel

Adding Noise

"Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications"
2012, Kourosh Khoshelham and Sander Elberink

$$\sigma_z = 1.425\text{e}{-5} \times Z^2$$

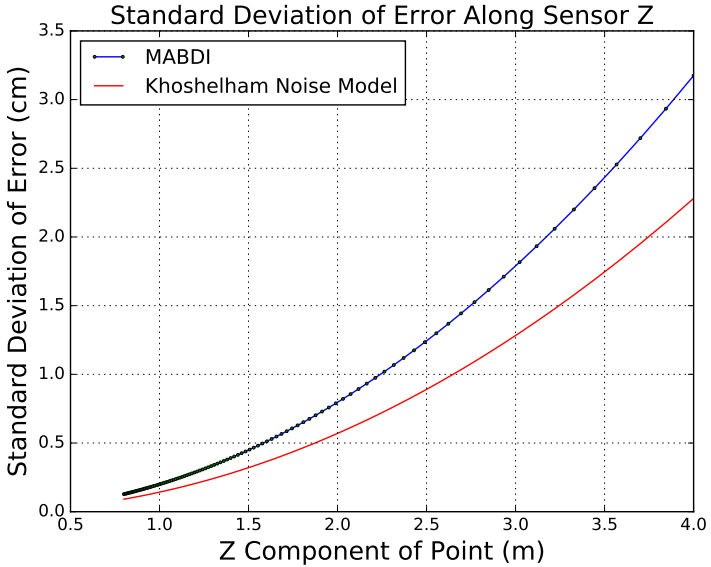
Adding Noise

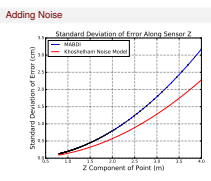
"Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications"
2012, Kourosh Khoshelham and Sander Elberink

$\sigma_z = 1.425\text{e}{-5} \times Z^2$

- Researchers have created error models to describe the standard deviation of measurement error found in various RGB-D sensors
- Point in camera's coordinate system
- Point has some z value Z
- σ_z - standard deviation of error
- as a function of Z

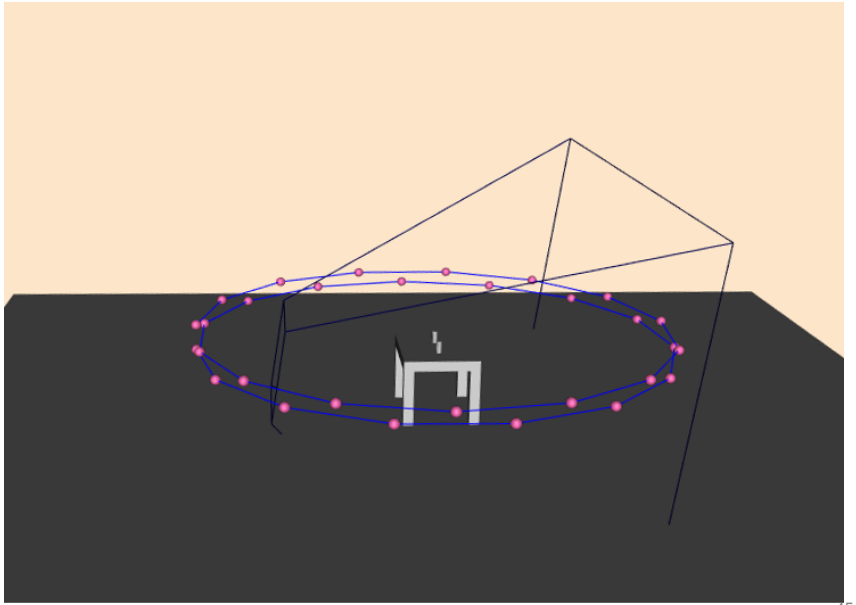
Adding Noise

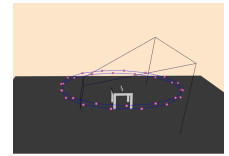




- Each line shows how the measurement's standard deviation of error changes as the point moves along the z axis in the sensor's coordinate system.
- The standard deviation of error simulated in our experiments is larger than that defined by Khoshelham's model for points within the sensor's range.

Sensor Path





- blue line indicates the path
- pink points indicate where the sensor stops along the path.
- path circles the objects in the environment twice. Helical path was chosen
- - it returns to a part of the environment that has already been mapped and is thus “known” to the algorithm
- - helix so 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

- Parameters that control the simulated environment
- with chosen values for each run

  Environment
- *Table*
- - consists of a table and two cups placed on the table. The table is 1 meter tall.
- - *Bunnies*
- - three bunnies around 1.5 meters tall. Created with Stanford Bunny a well known data set in computer graphics.

  Noise - adds noise to the depth image of the simulated sensor

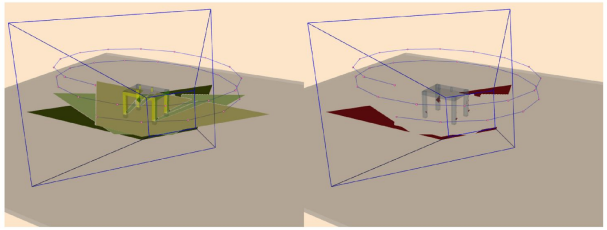
  Dynamic - adds an object during the simulation
- In this experiment a third bunny is added half-way through the simulation

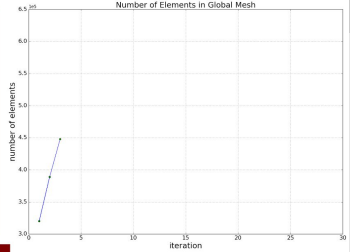
Outline

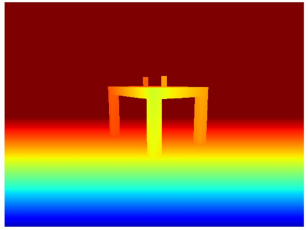
Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

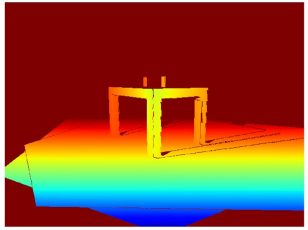
During the Experiment

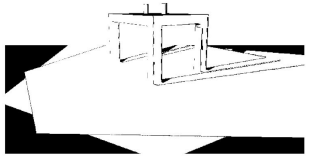
(a) (b) (c)



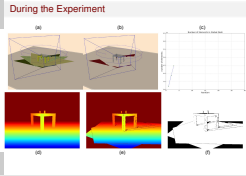






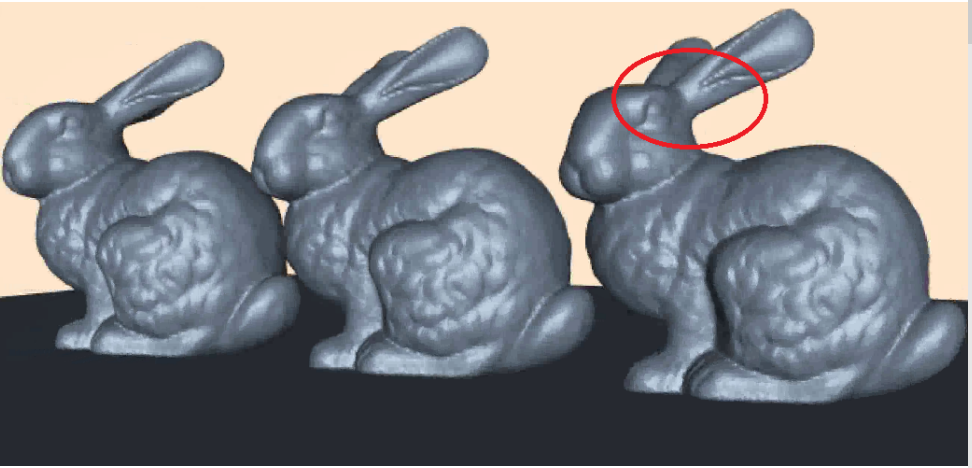


(d) (e) (f)

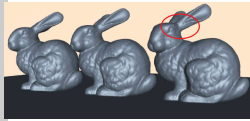


- Input - **??**(d) shows the depth image D generated from the simulated sensor. **??**(a) shows us two important aspects to consider about D. First, the pose P of the sensor is shown by looking at the sensor's view frustum, indicated by the blue wireframe. Second, the only environmental information 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

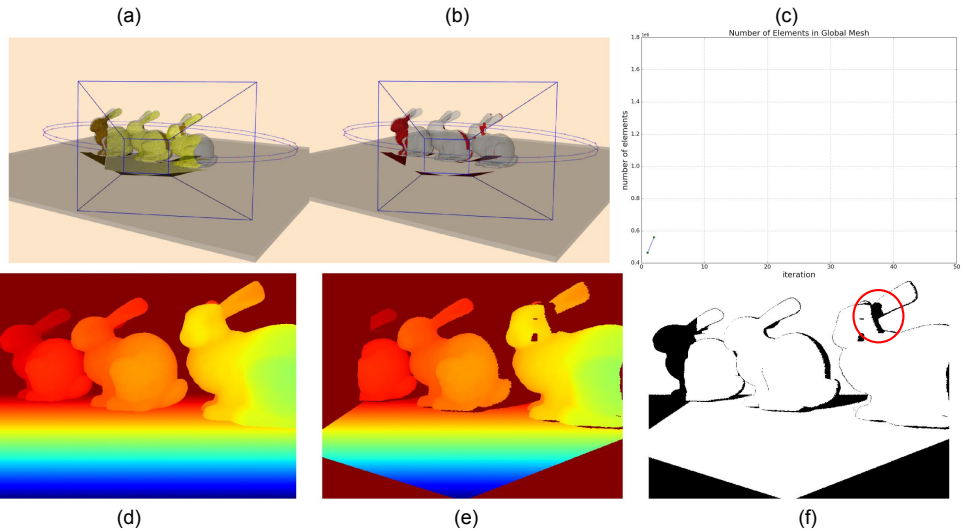


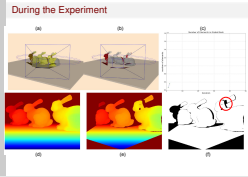
During the Experiment



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

During the Experiment

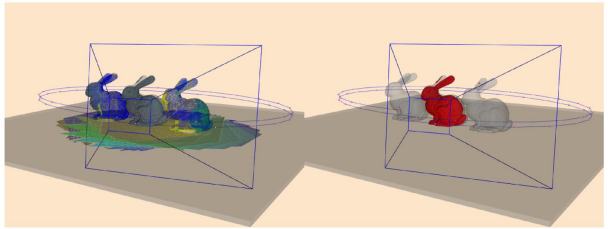


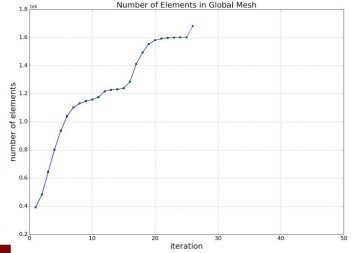


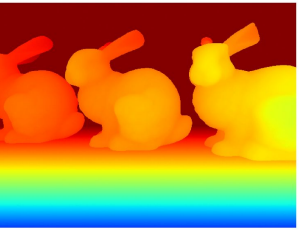
- **??**(a) shows the global mesh M. The yellow portion of the mesh constitutes the entirety of M after the first iteration. We can see the novel portion of the environment was not represented in M after the first iteration due to occlusion.
- **??**(d) shows the depth image D from the new sensor pose 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

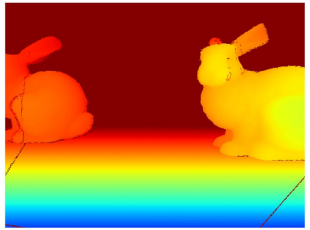
During the Experiment

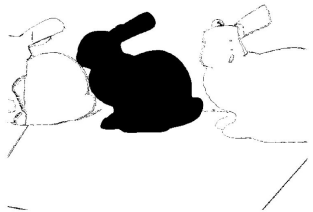
(a) (b) (c)



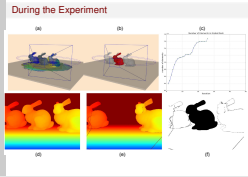




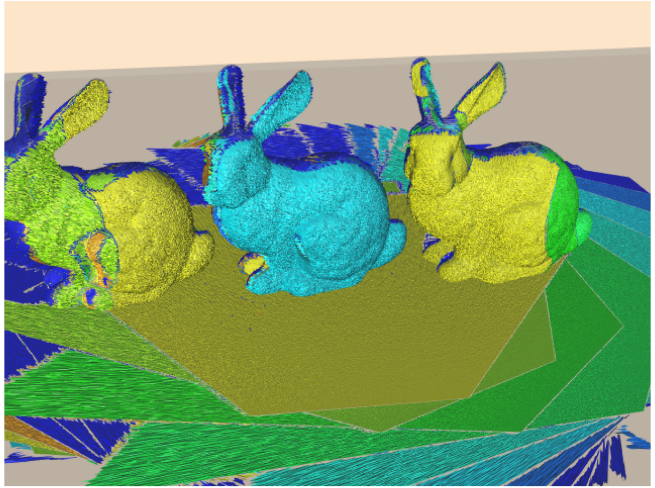




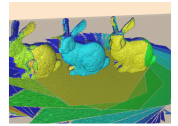
(d) (e) (f)



Mesh Quality

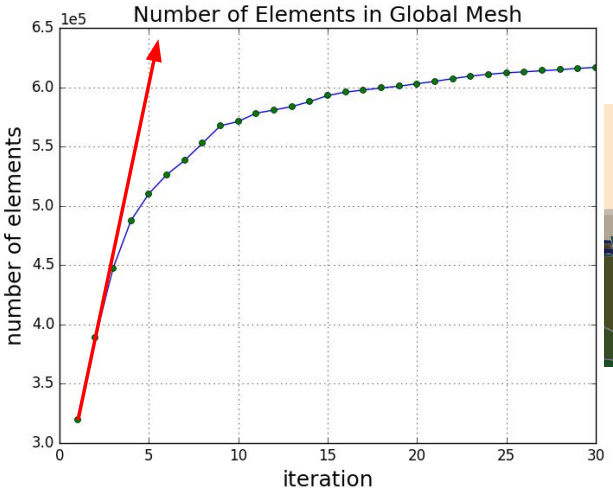


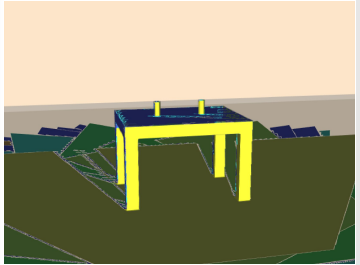
Mesh Quality

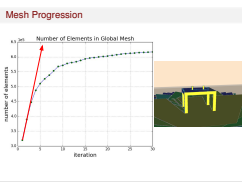


- There are gaps in the mesh that occur typically along the boundaries of where the novel surface S is appended to the global mesh M. This behavior is common for Surface Reconstruction methods as those discussed in Section **??**. Algorithms exist for merging these gaps as a post processing step such 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

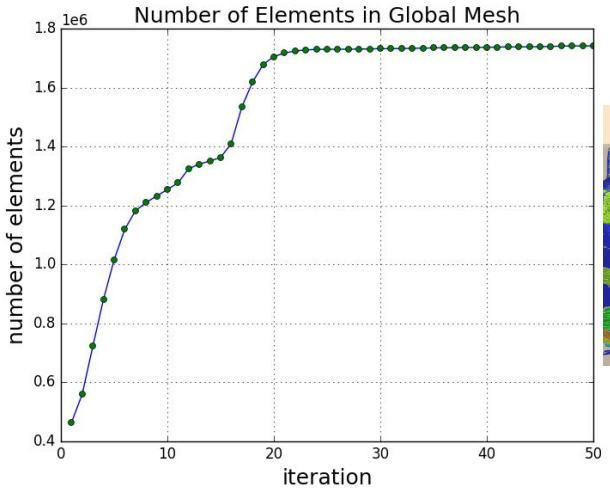


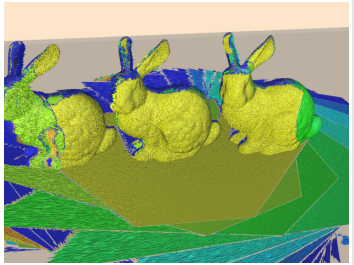


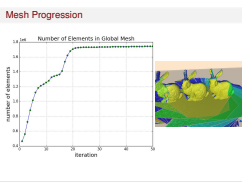


- Figure **??** shows the resultant mesh and mesh progression for the first experiment. The plot highlights the major difference between MABDI and traditional mesh-based environmental mapping methods. Traditional methods would have a plot similar to that indicated by the red arrow on the graph because 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

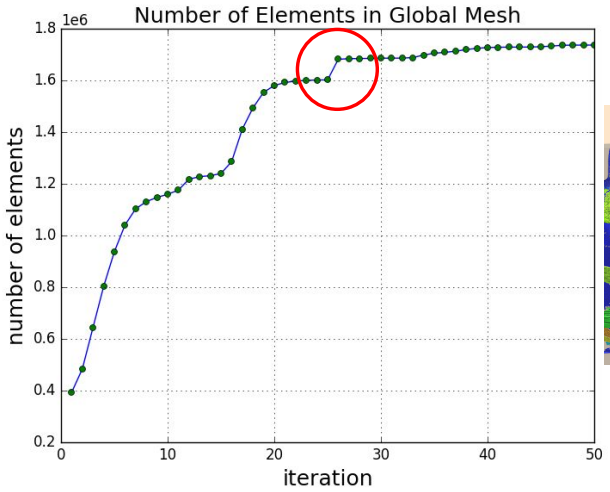


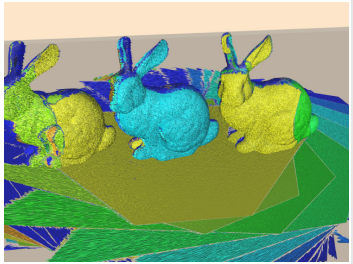


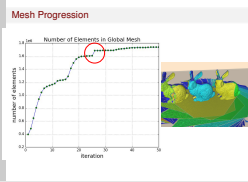


- Figure **??** shows us the resultant mesh after the second experiment. Here we can see that MABDI is reactive to the environment. In the preceding experiment, the environment was symmetrical. In this experiment, the environment is not symmetrical and we can see the effects by looking at the progression 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







- 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

Outline
Introduction
Overview
RBG-D Sensor
Map
Contribution
Approach
Algorithm
Surface Reconstruction
Software Design
Experimental Setup
Simulated Sensor
Simulation Parameters
Results
Conclusion

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