Mesh Addition Based on the Depth Image (MABDI)

Lucas Chavez

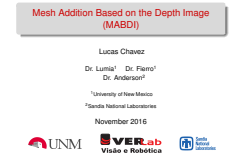
Dr. Lumia[1] Dr. Fierro[1]
Dr. Anderson[2]

[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.



- Introduce myself and committee
- My research is robotics
- and specifically environmental mapping
- meaning creating a map from sensor data
- The name of my algorithm

Outline

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

Outline

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

Overview

Motivation for this work: provide a map of the environment

Examples applications:

- Autonomous agents (robots)
- Teleoperation (human)

Overview

Motivation for this work: provide a map of the environment

Examples applications:

- Autonomous agents (robots)
- Teleoperation (human)

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

Overview

In the literature this is referred to as the SLAM problem

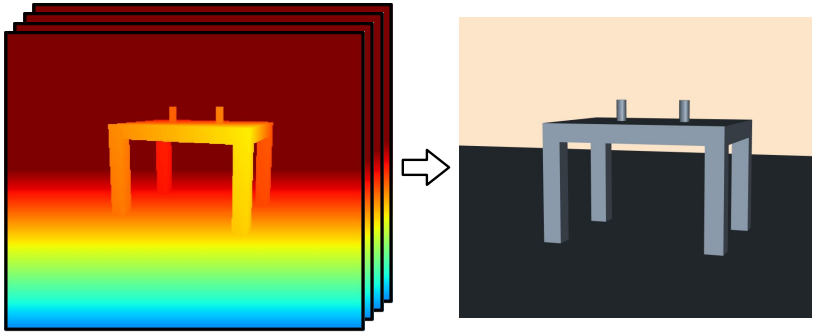
- Simultaneous Localization and Mapping
- Environmental Mapping
- Research began around 1987
- Sensing and computing technology

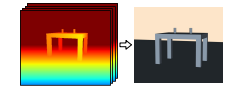
Overview

In the literature this is 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 was to transform depth images into a mesh representation
- * I will explain these two components
- 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

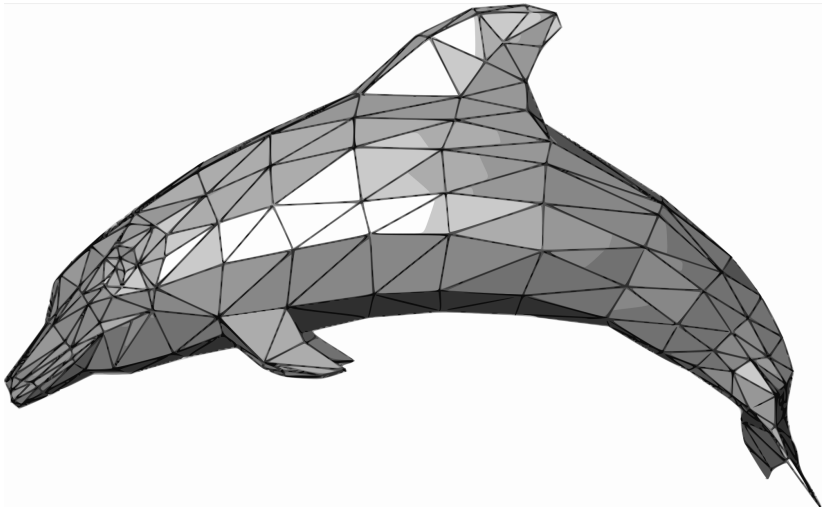
- 30 frames per second
- D - 9 million pixel values per second
- Algorithms must handle a high rate of data

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

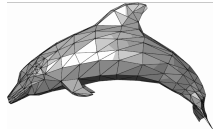
- Kinect
- First affordable sensor to provide
- high resolution spatial information

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

Mesh



Mesh



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

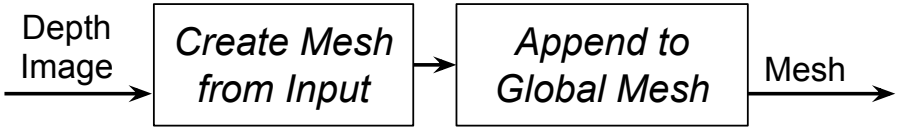
Mesh

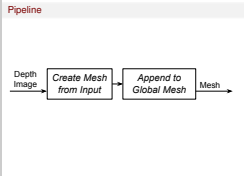
- Supported
- Computationally Inexpensive
- Low Memory Requirement

Mesh
Supported
Computationally Inexpensive
Low Memory Requirement

- Supported
- - available software, tools, research, algorithms, etc., for this type of map
- Computationally Inexpensive
- - GPUs make this computationally efficient
- Low Memory Requirement
- - Can it run on a laptop with a standard amount of RAM?

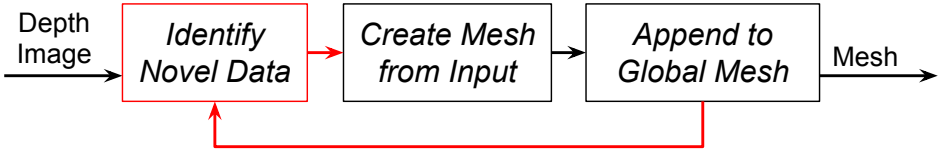
Pipeline

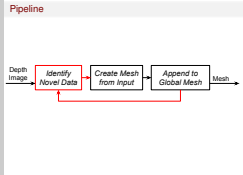




- Traditional Mesh-Based Mapping Methods
- - Take incoming data
- - Generate a mesh structure
- - Append to growing global mesh structure

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.

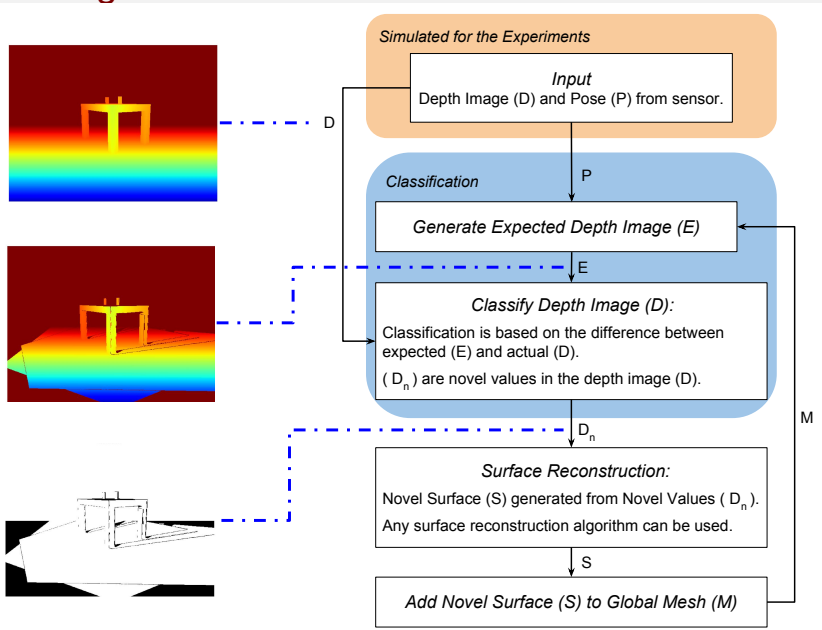
Contribution

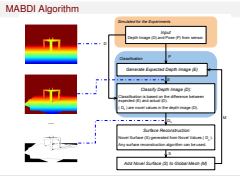
MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.

Outline

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

MABDI Algorithm





System diagram of the MABDI algorithm

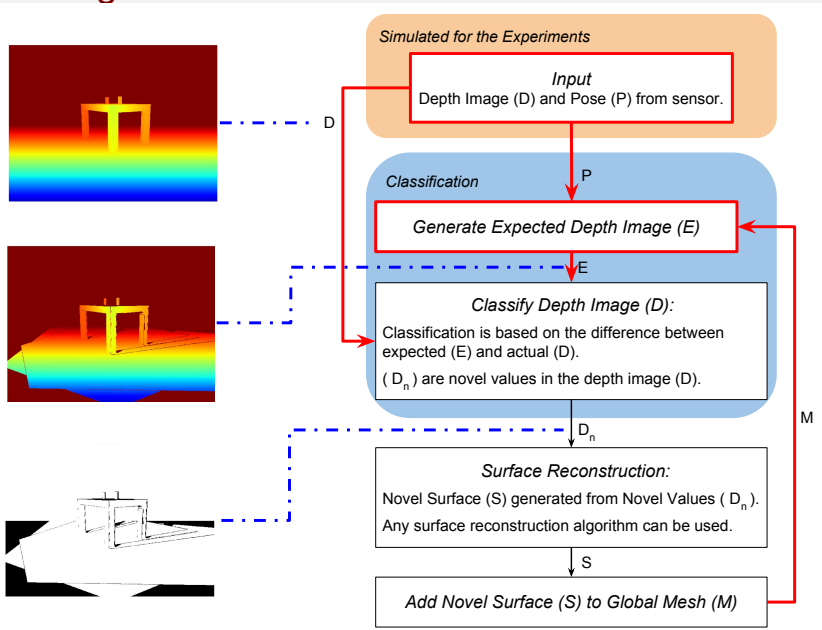
Orange -

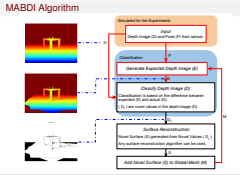
- Input to the algorithm
- has been simulated for this work.

Classification -

- These components are what make MABDI unique
- They classify data before it is added to the Global Mesh.

MABDI Algorithm





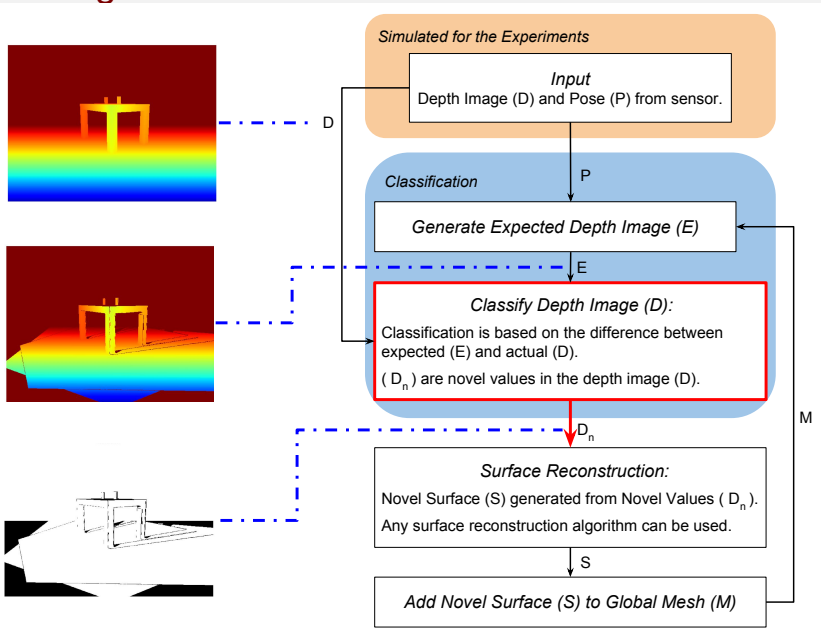
Input

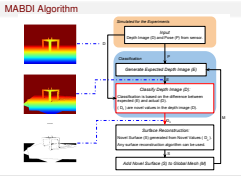
- I will cover simulation process in the next section.

Generate Expected Depth Image

- Takes the global mesh (what we know about the environment)
- And pose of sensor.
- Generates what we expect to see.
- Meaning what would a RGB-D sensor would see.
- Given what we know about the environment and the pose of the sensor.

MABDI Algorithm

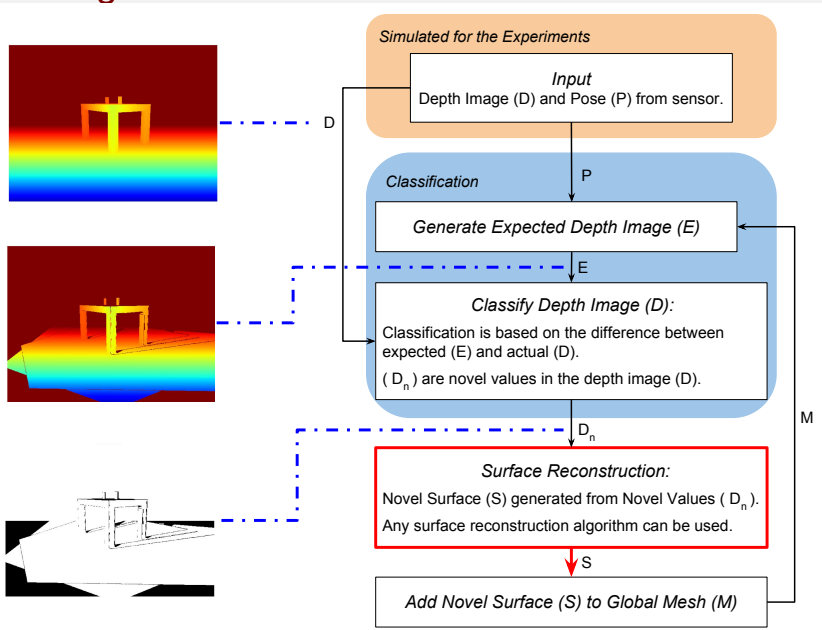


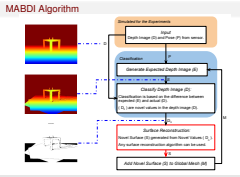


Classify Depth Image

- This is the heart of MABDI
- and is MABDI's contribution to the state-of-the-art
- It determines 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

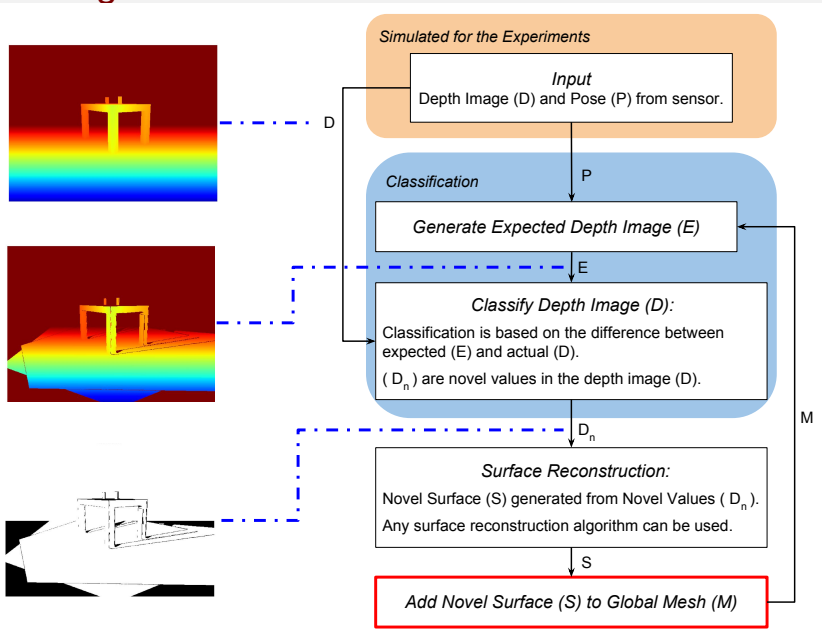


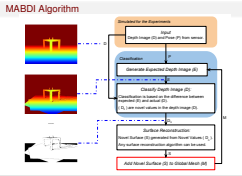


Surface Reconstruction

- 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

Initial Mesh

Surface Reconstruction component is responsible for creating the novel surface S from the novel points D_n

Our Method:

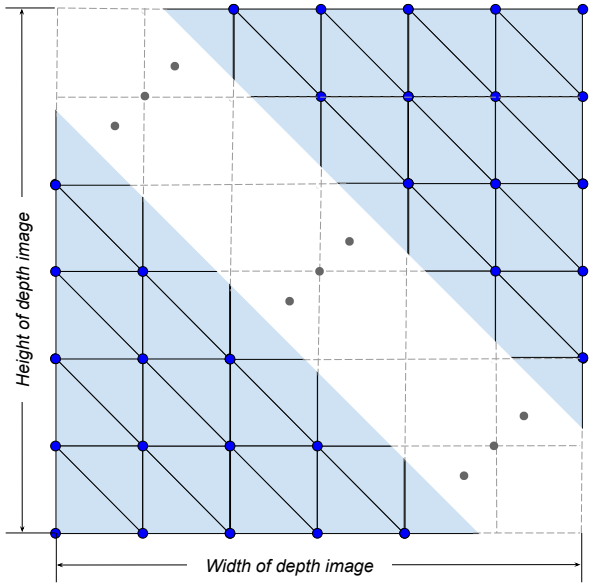
- Define topology in 2D, on the depth image
- Project to 3D
- Remove elements

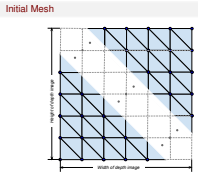
Initial Mesh
Surface Reconstruction component is responsible for creating the novel surface S from the novel points D_n
Our Method:
- Define topology in 2D, on the depth image
- Project to 3D
- Remove elements

- I just described the MABDI algorithm
- The algorithm is simply an idea
- I am going to describe how I implemented this idea

- topology or structure

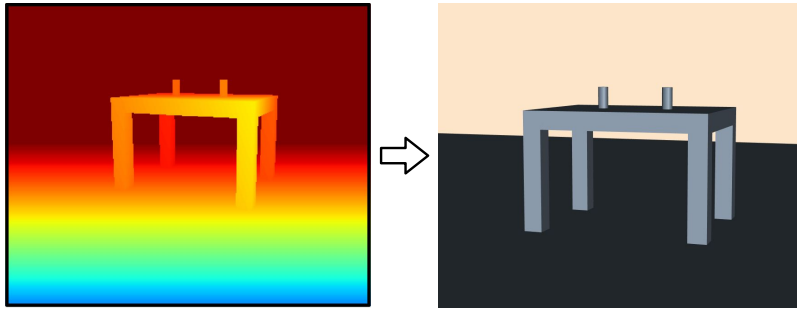
Initial Mesh



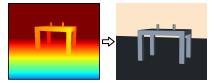


- Imagine this is the depth image
- - Every blue point is a pixel in the depth image
- - Corresponds to 3D point in space
- Depth image
- - Not a set of unorganized points
- - Has structural information
- - This allows us to define a topology in 2D that is preserved when projected to 3D
- I then take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

Initial Mesh

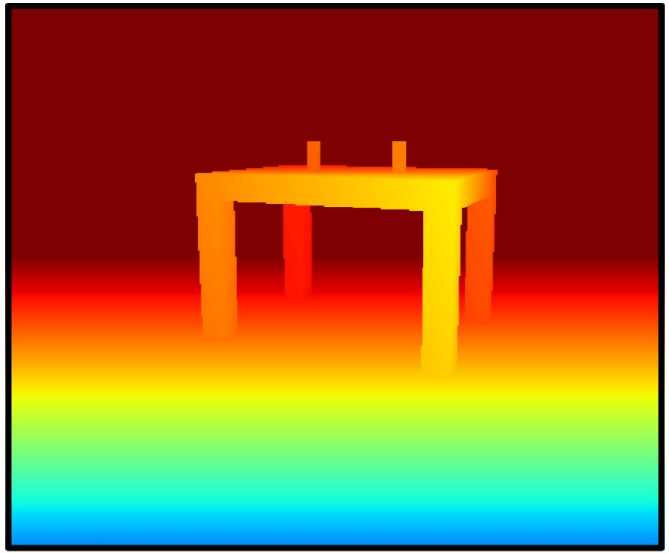


Initial Mesh

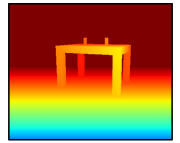


- Imagine mesh being defines using every pixel in depth image
- Then projected to 3D space
- - Ignore the background for now
- Note there will be no surface
- behind the cup,
- under the table,
- anywhere on the floor that the sensor doesn't see

Removing elements

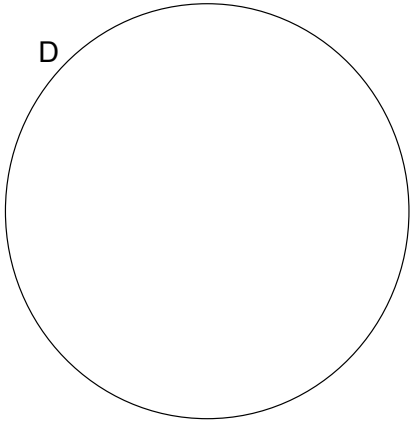


Removing elements

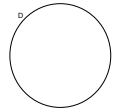


- I begin removing elements by first identifying points to be removed from the depth image

Removing elements

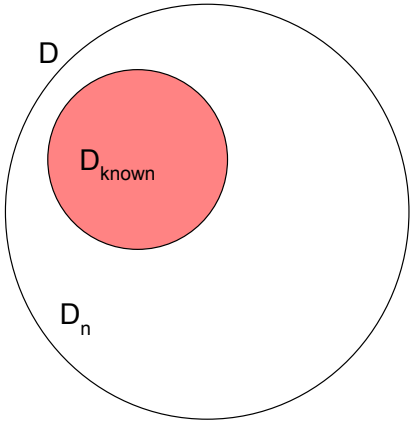


Removing elements



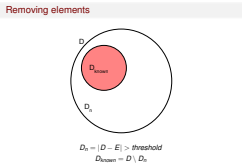
- Inside the circle represents every point from D

Removing elements



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

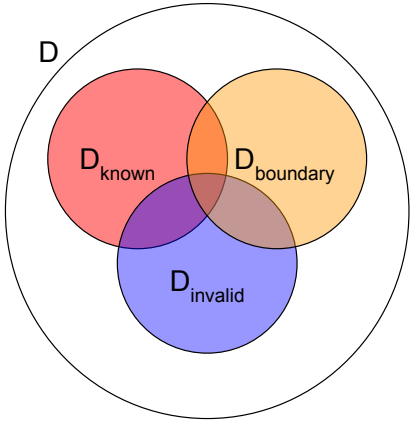
$$D_{known} = D \setminus D_n$$



- D_n (novel) is the set of points that the categorization process said is novel
- Categorization process is defined by this equation

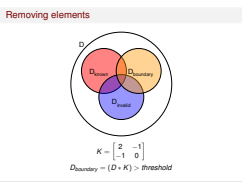
  absolute difference between what the sensor saw and what we expect to see

Removing elements



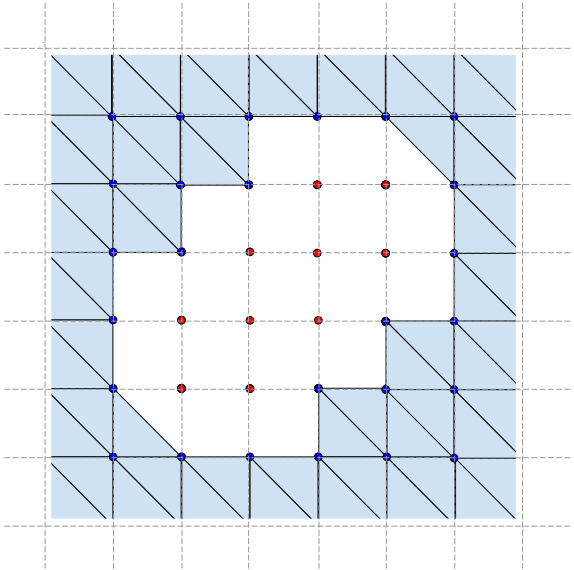
$$K = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix}$$

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

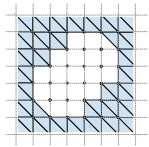


- I define two additional set of points to be thrown away.
- $D_{boundary}$
- - remove elements defined by points that lie on completely different surfaces
- - pixel neighbors floor and leg (point out difference in values)
- - To achieve this a two dimensional, differencing convolution filter is passed over D
- - Filter has a magnified response at points where the difference between neighboring pixels is large
- $D_{invalid}$
- - elements that are out of range of the sensor

Removing elements



Removing elements

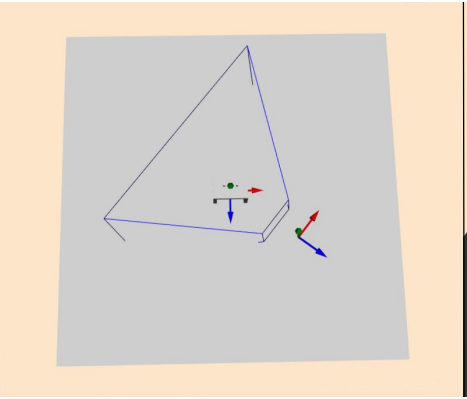


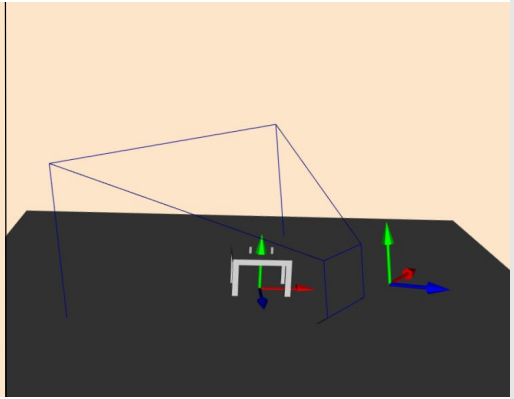
- Points from the colored circles are represented as the red points
- Technically, these points have already been projected to 3D
- but this is the best way to visualize this concept

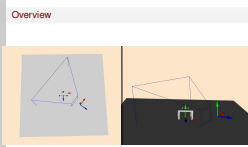
Outline

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

Overview







- ** All experiments done in a simulation environment
- 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

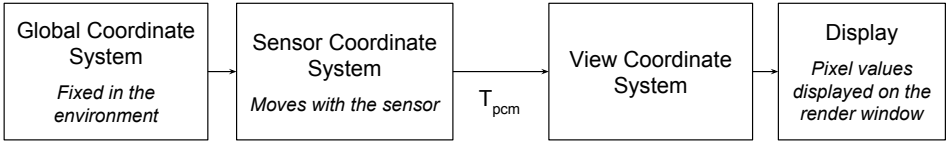
Simulating a RGB-D Sensor

- Adding noise to the depth image
- Sensor path

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

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

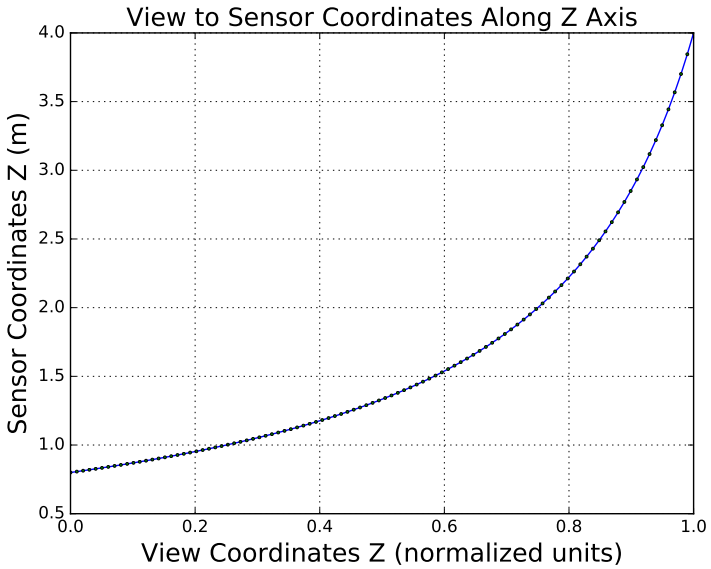
Rendering Pipeline

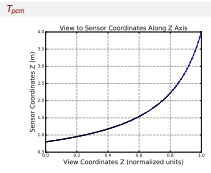


Rendering Pipeline

- 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

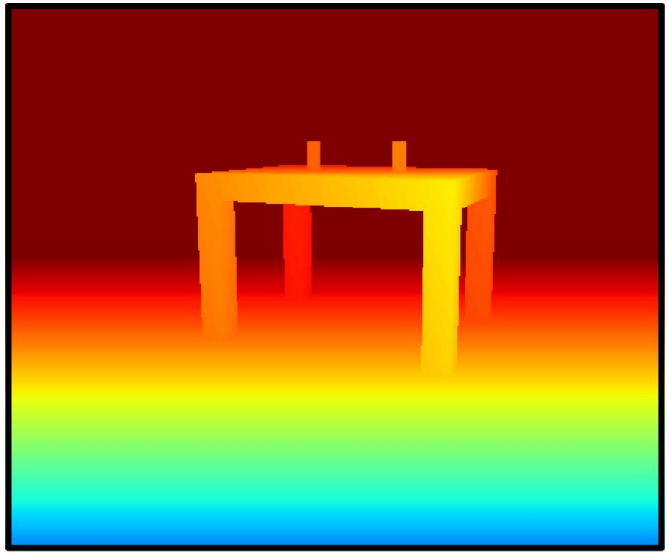
T_{pcm}



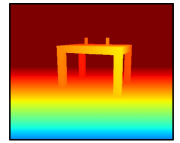


- The pinhole camera transformation, T_{pcm},
- Creates a non-linear relationship between
- - values in the depth image
- - and their corresponding location in the sensor's coordinate system
- Graph was created by:
- - Taking center pixel in depth image
- - Varying value 0-1 and projecting to camera's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

Adding Noise



Adding Noise



- The depth image

- I am going to describe how to take a "noiseless" depth image
- and add noise to mimic a real RGB-D sensor in the real world

Adding Noise

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

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

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

Adding Noise

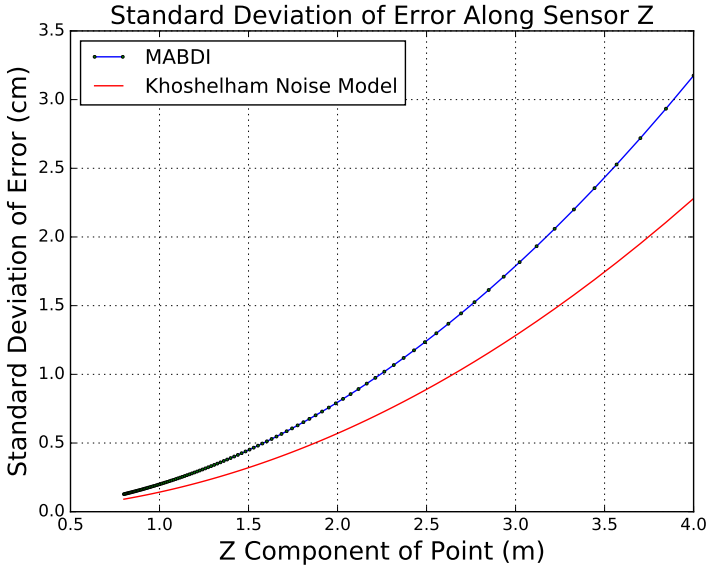
K. Khoshelham and S. O. Elberink, "Accuracy and resolution of Kinect depth data for indoor mapping applications." *Sensors (Basel, Switzerland)*

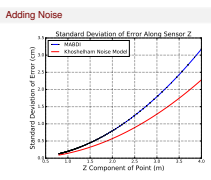
$$\sigma_z = 1.425\text{e}{-5} \times Z^2$$

Adding Noise
K. Khoshelham and S. O. Elberink, "Accuracy and resolution of Kinect depth data for indoor mapping applications." *Sensors (Basel, Switzerland)*
$\sigma_z = 1.425\text{e}{-5} \times Z^2$

- 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

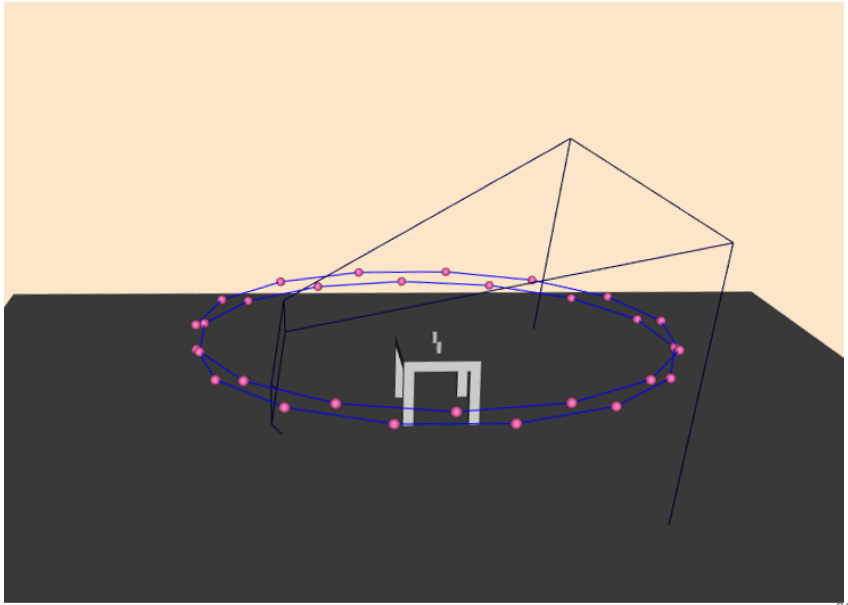
Adding Noise

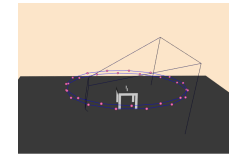




- Each line shows how the measurement's standard deviation of error changes as the point moves along the z axis in the sensor's coordinate system.
- The standard deviation of error simulated in our experiments is larger than that defined by Khoshelham's model for points within the sensor's range.

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.

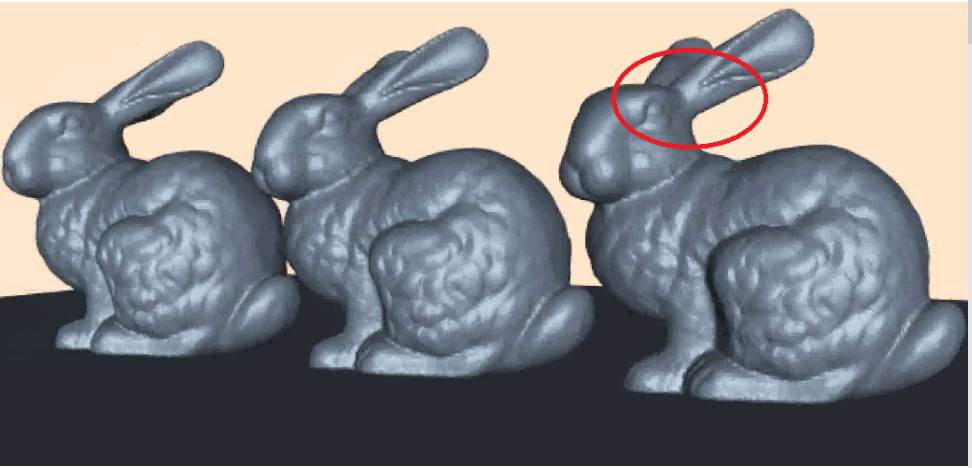
  Noise - adds noise to the depth image of the simulated sensor

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

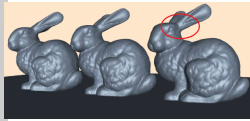
Outline

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

During the Experiment



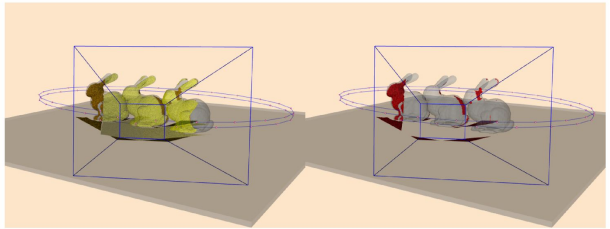
During the Experiment

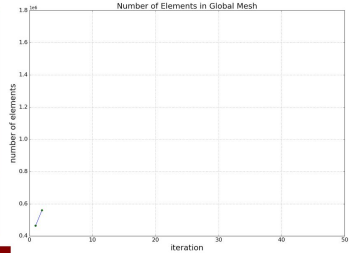


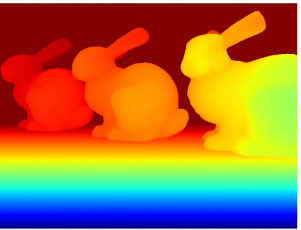
- Novel portion of the environment.
- To be discussed during Experiment 2.

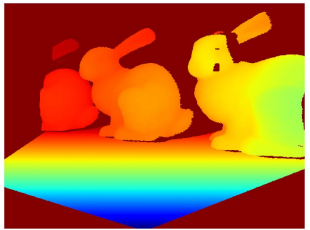
During the Experiment

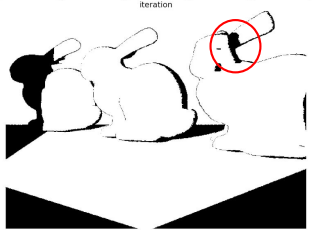
(a) (b) (c)



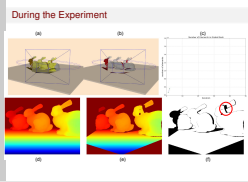








(d) (e) (f)



Yellow portion is the entirety of M after the first iteration

- Due to occlusion, novel portion not represented

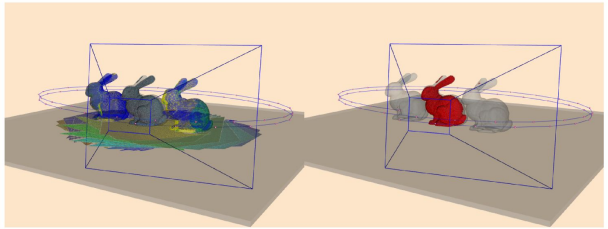
  Sensor sees portion on this iteration

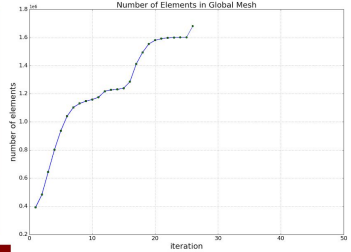
  We don't expect to see portion (not in M)
- classification process successfully identifies (highlighted by a red circle)

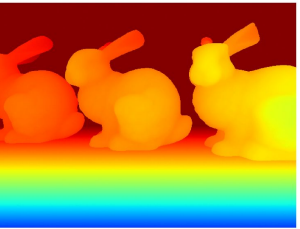
  Novel surface S now represents the novel portion
- Finally
- novel portion is represented by the global mesh M.

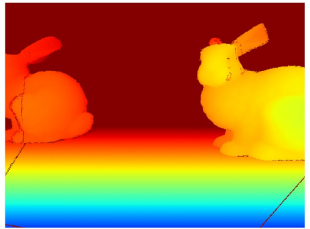
During the Experiment

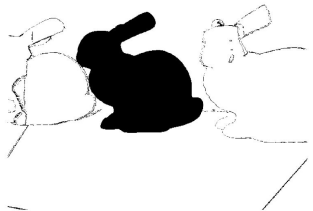
(a) (b) (c)



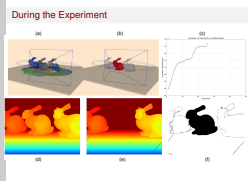








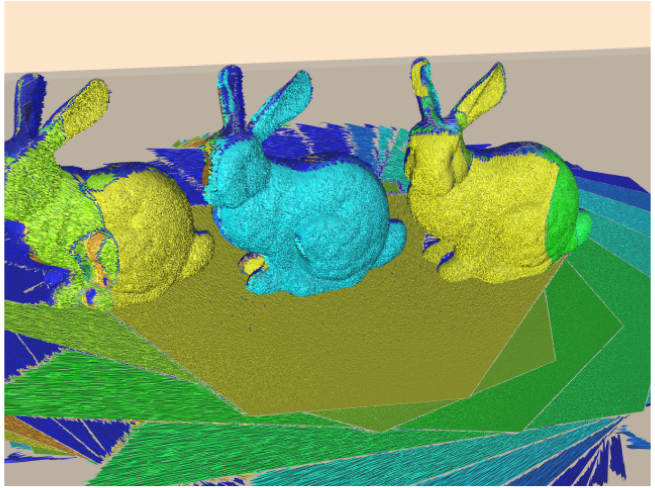
(d) (e) (f)



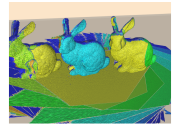
- Bunny added during this iteration
- *D* shows the new bunny
- *E* does not show the new bunny
- classification process successfully identifies
- novel points are used to generate the novel surface *S*
- *S* is appended to *M*

  S has large number of elements for this particular iteration
- plot shows resulting jump in the number of elements contained with *M*

Mesh Quality



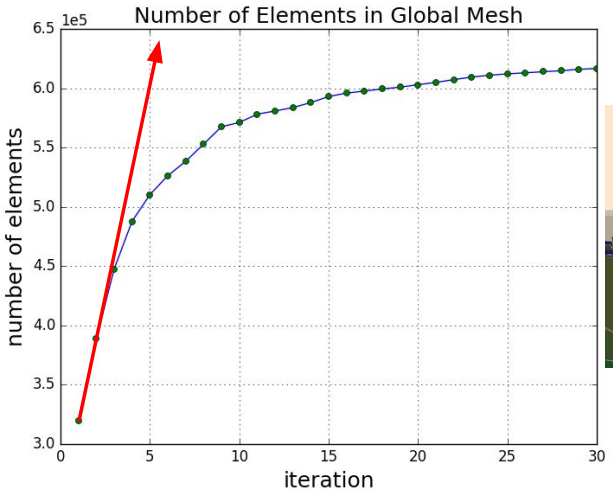
Mesh Quality

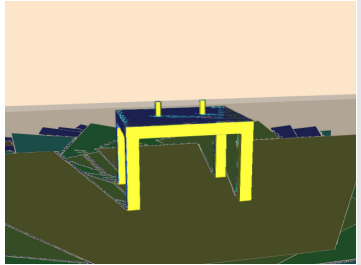


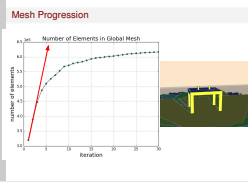
- Gaps in the mesh
- - Traditional methods have overlapping layers
- - So you don't notice gaps
- The mesh is noisy.
- - Our method simply connects neighboring points in the point cloud without additional steps such as Laplacian smoothing

- My reconstruction method was sufficient for demonstrating the usefulness of the MABDI algorithm
- - mesh has the same magnitude of noise as the sensor's simulated noise

Mesh Progression







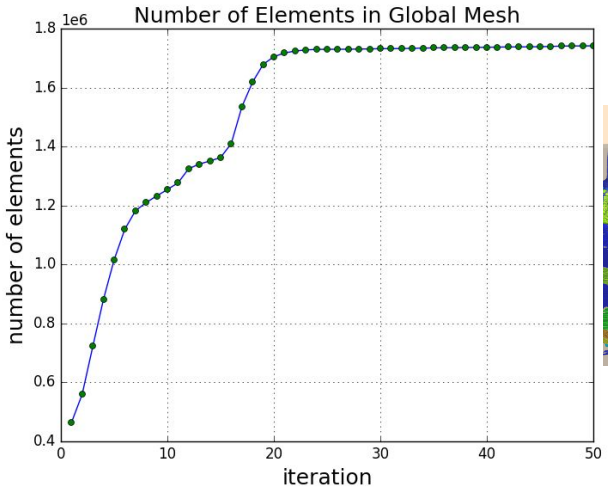
Traditional methods

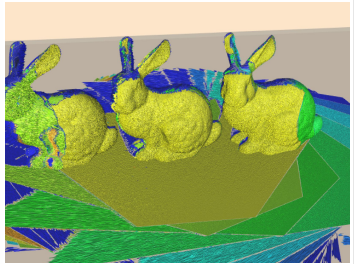
- would have a plot similar to that indicated by the red arrow on the graph

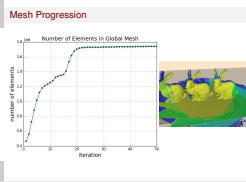
MABDI

- levels off as the environment becomes more known

Mesh Progression

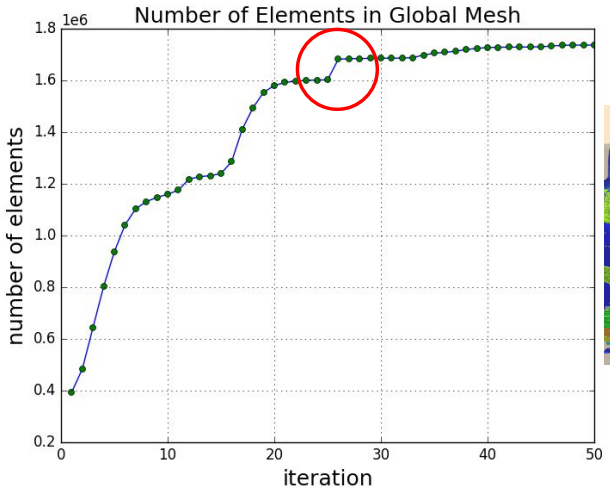


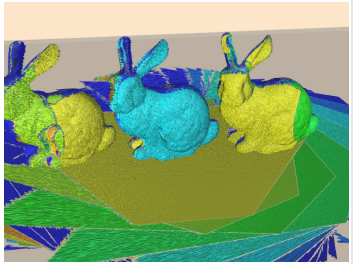


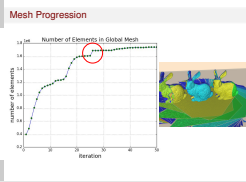


- MABDI is reactive as the sensor moves to parts of the environment that are rich in information
- the mesh grows rapidly based on the needs of the environment

Mesh Progression







- Large jump in graph corresponding to new bunny

Outline

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

Conclusion

- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

Conclusion
- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it before it is added to the global mesh.
- 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.

- Currently MABDI is only designed to handle object addition
- can be extended to handle both object addition and removal
- allowing ability to handle highly dynamic environments

Conclusion

- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

Conclusion
- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

- Currently MABDI is only designed to handle object addition
- can be extended to handle both object addition and removal
- allowing ability to handle highly dynamic environments

Conclusion

- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

Conclusion
- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

- Currently MABDI is only designed to handle object addition
- can be extended to handle both object addition and removal
- allowing ability to handle highly dynamic environments

Conclusion

- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

Conclusion
- The MABDI algorithm runs at around 2Hz on a consumer grade laptop with an Intel i7 processor.
- MABDI's algorithmic design identifies redundant information and removes it *before* it is added to the global mesh.
- 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.

- Currently MABDI is only designed to handle object addition
- can be extended to handle both object addition and removal
- allowing ability to handle highly dynamic environments

Thank you for your attention

Thank you for your attention

- I want to thank my advisor and my committee
- All my friends that are here
- Friends and family that have supported me

- I am proud of my work
- Again, thank you for your support and attention