Learning-aided 3D Occupancy Mapping with Mobile Robots

Kevin Doherty

May 19, 2017

Table of Contents

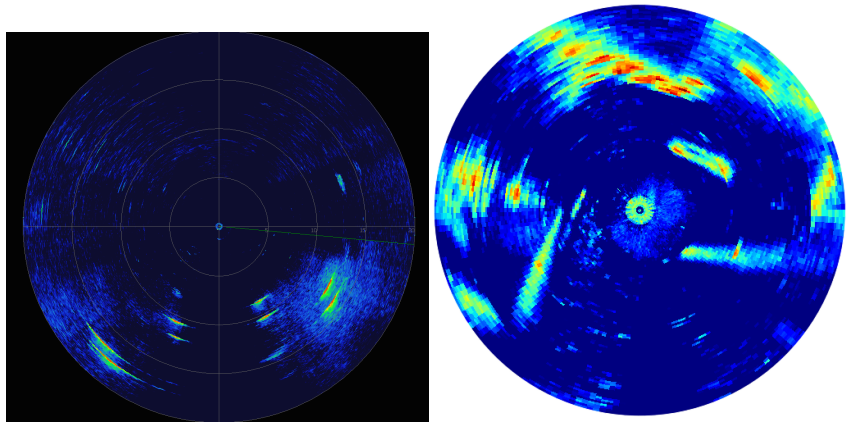
Motivation

Applications of mobile robots warrant reliable planning and navigation, but sensors often give sparse and noisy readings

Motivation

Applications of mobile robots warrant reliable planning and navigation, but sensors often give sparse and noisy readings

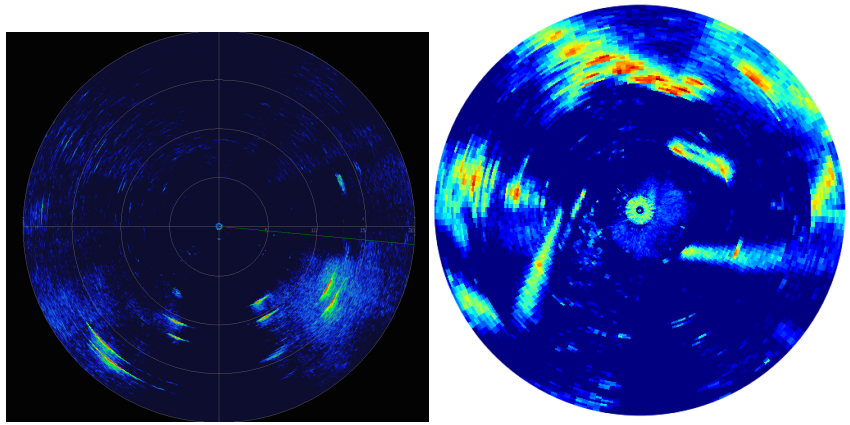
- Underwater navigation using sonar
  - Environmental disturbances
  - Low resolution
  - Multiple returns



Motivation

Applications of mobile robots warrant reliable planning and navigation, but sensors often give sparse and noisy readings

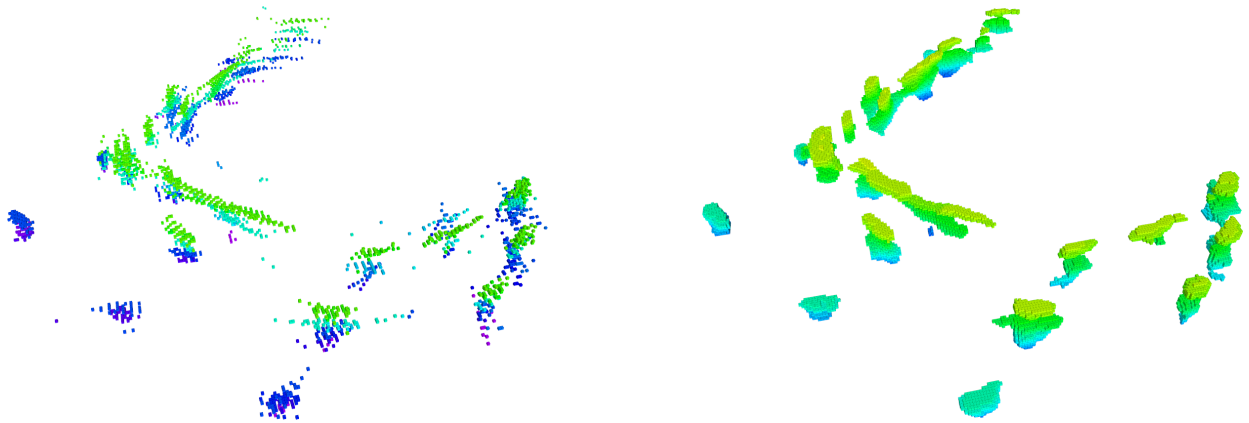
- Underwater navigation using sonar
  - Environmental disturbances
  - Low resolution
  - Multiple returns



- Traditional occupancy grid mapping is limited by strong spatial independence assumptions

Motivation

Applications of mobile robots warrant reliable planning and navigation, but sensors often give sparse and noisy readings

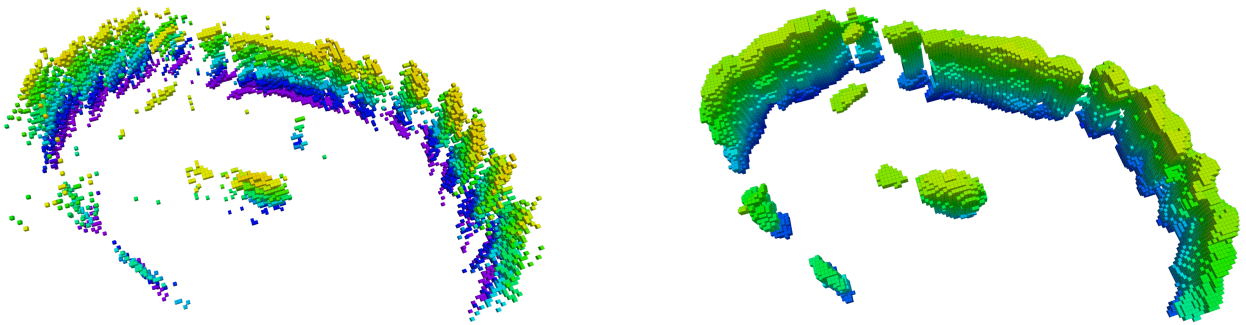


(a) Pier 84 OctoMap (b) Pier 84 GP OctoMap

- 3D maps built with single-beam scanning sonar, from Wang et al. [37]

Motivation

Applications of mobile robots warrant reliable planning and navigation, but sensors often give sparse and noisy readings



(a) USMMA OctoMap (b) USMMA GP OctoMap

- 3D maps built with single-beam scanning sonar, from Wang et al. [37]

Problem Statement

We are concerned with the problem of mapping an *a priori* unknown environment using range sensors

Problem Statement

We are concerned with the problem of mapping an *a priori* unknown environment using range sensors

- Particularly, when a range sensor provides sparse and noisy measurements

Problem Statement

We are concerned with the problem of mapping an *a priori* unknown environment using range sensors

- Particularly, when a range sensor provides sparse and noisy measurements
- A solution should be real-time, taking place online

Problem Statement

We are concerned with the problem of mapping an *a priori* unknown environment using range sensors

- Particularly, when a range sensor provides sparse and noisy measurements
- A solution should be real-time, taking place online
- **Assumptions:**
  - We will assume that the map is static
  - We relax the assumption in standard occupancy grid mapping that cells of a grid map are independent

Our Contributions

We present two learning-aided mapping methods:

Our Contributions

We present two learning-aided mapping methods:

- **Overlapping Hilbert Maps**
  - Approximate approach to Hilbert Maps, proposed by Ramos and Ott [22]
  - Train an estimator for each robot pose and fuse predictions

Our Contributions

We present two learning-aided mapping methods:

- **Overlapping Hilbert Maps**
  - Approximate approach to Hilbert Maps, proposed by Ramos and Ott [22]
  - Train an estimator for each robot pose and fuse predictions
- **Bayesian Generalized Kernel Occupancy Maps**
  - Perform Bayesian inference with conjugate prior
  - Exact recursive updates
  - Introduce spatial correlations with kernel smoothing

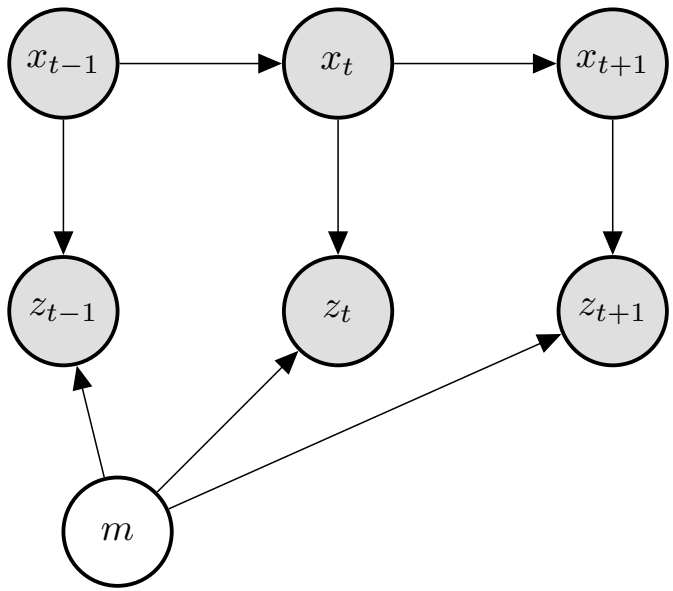
Table of Contents

Mapping with Known Poses

What is the probability of map m, comprised of cells m_i, given measurements $z_{1:t}$ taken at poses $x_{1:t}$?

Mapping with Known Poses

What is the probability of map m, comprised of cells m_i, given measurements $z_{1:t}$ taken at poses $x_{1:t}$?

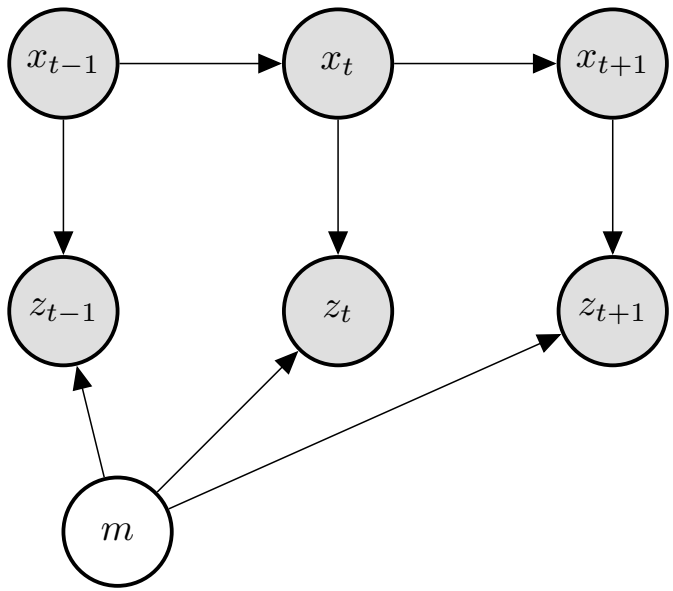


Mapping with Known Poses

What is the probability of map m, comprised of cells m_i, given measurements $z_{1:t}$ taken at poses $x_{1:t}$?

Assume m_i independent:

$$p(m|x_{1:t}, z_{1:t}) = \prod_{i=1}^{N} p(m_i|x_{1:t}, z_{1:t})$$

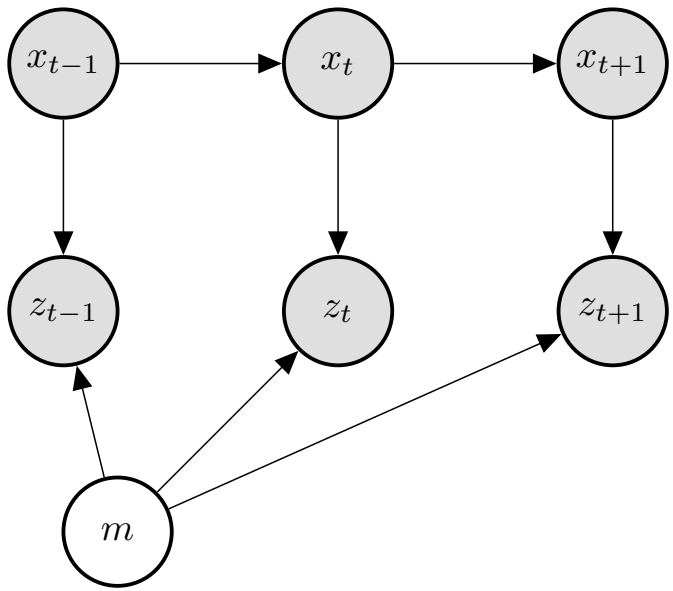


Mapping with Known Poses

What is the probability of map m, comprised of cells m_i, given measurements $z_{1:t}$ taken at poses $x_{1:t}$?

Assume m_i independent:

$$p(m|x_{1:t}, z_{1:t}) = \prod_{i=1}^{N} p(m_i|x_{1:t}, z_{1:t})$$



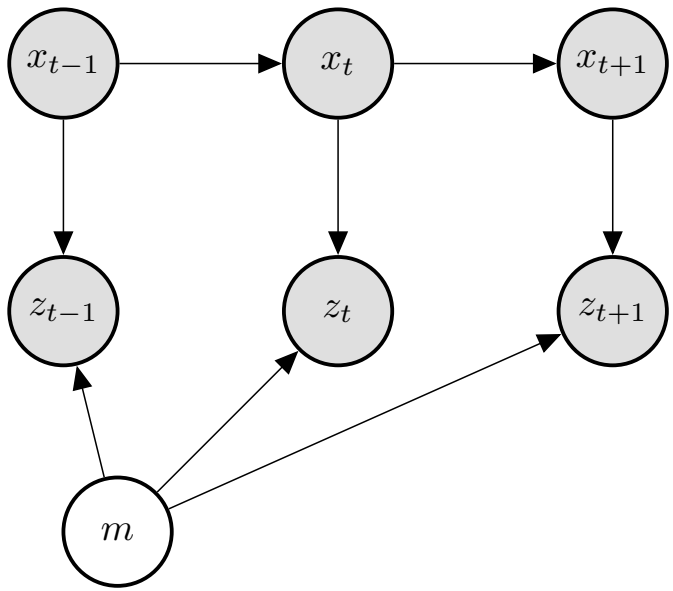
- Estimate each $p(m_i|z_{1:t}, x_{1:t})$ individually

Mapping with Known Poses

What is the probability of map m, comprised of cells m_i, given measurements $z_{1:t}$ taken at poses $x_{1:t}$?

Assume m_i independent:

$$p(m|x_{1:t}, z_{1:t}) = \prod_{i=1}^{N} p(m_i|x_{1:t}, z_{1:t})$$



- Estimate each $p(m_i|z_{1:t}, x_{1:t})$ individually

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})}$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$= \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Markov)}$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$
\begin{aligned}
p(m_i|z_{1:t}, x_{1:t}) &= \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} && \text{(Bayes)} \\
&= \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} && \text{(Markov)} \\
&= \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})} && \text{(Bayes)}
\end{aligned}
$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$= \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Markov)}$$

$$= \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$p(\neg m_i|z_{1:t}, x_{1:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$= \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Markov)}$$

$$= \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$p(\neg m_i|z_{1:t}, x_{1:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i|z_{1:t-1}, x_{1:t-1})} \frac{p(\neg m_i)}{p(m_i)}$$

Mapping with Known Poses

We can derive the recursive update as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$= \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Markov)}$$

$$= \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})} \quad \text{(Bayes)}$$

$$p(\neg m_i|z_{1:t}, x_{1:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i|z_{1:t-1}, x_{1:t-1})} \frac{p(\neg m_i)}{p(m_i)}$$

$$\log \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} + \log \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} - \log \frac{p(m_i)}{1 - p(m_i)}$$
$$= l_{t,i} + l_{1:t-1,i} - l_0$$

Mapping with Known Poses

Occupancy Grid Mapping Algorithm

Given new (state, measurement) pair (x_t, z_t):

- Traverse every cell in the grid
  - If a cell is visible, update its log-odds
  - Otherwise, leave it alone

Mapping with Known Poses

Occupancy Grid Mapping Algorithm

Given new (state, measurement) pair (x_t, z_t):

- Traverse every cell in the grid
  - If a cell is visible, update its log-odds
  - Otherwise, leave it alone

What can be improved about this approach?

Mapping with Known Poses

Occupancy Grid Mapping Algorithm

Given new (state, measurement) pair (x_t, z_t):

- Traverse every cell in the grid
  - If a cell is visible, update its log-odds
  - Otherwise, leave it alone

What can be improved about this approach?

- Better inverse sensor model? Heuristic? Learned?

Mapping with Known Poses

Occupancy Grid Mapping Algorithm

Given new (state, measurement) pair (x_t, z_t):

- Traverse every cell in the grid
  - If a cell is visible, update its log-odds
  - Otherwise, leave it alone

What can be improved about this approach?

- Better inverse sensor model? Heuristic? Learned?
- Map is static (Markov assumption)

Mapping with Known Poses

Occupancy Grid Mapping Algorithm

Given new (state, measurement) pair (x_t, z_t):

- Traverse every cell in the grid
  - If a cell is visible, update its log-odds
  - Otherwise, leave it alone

What can be improved about this approach?

- Better inverse sensor model? Heuristic? Learned?
- Map is static (Markov assumption)
- Cells are independent

Learning Occupancy Maps

Historically, two avenues pursued:

Learning Occupancy Maps

Historically, two avenues pursued:

- Learning the inverse sensor model [30]
  - Avoids heuristics
  - Learn the characteristics of particular sensor
  - Doesn't consider readings outside the measurement cone

Learning Occupancy Maps

Historically, two avenues pursued:

- Learning the inverse sensor model [30]
  - Avoids heuristics
  - Learn the characteristics of particular sensor
  - Doesn't consider readings outside the measurement cone
- Considering correlations between all cells [31]

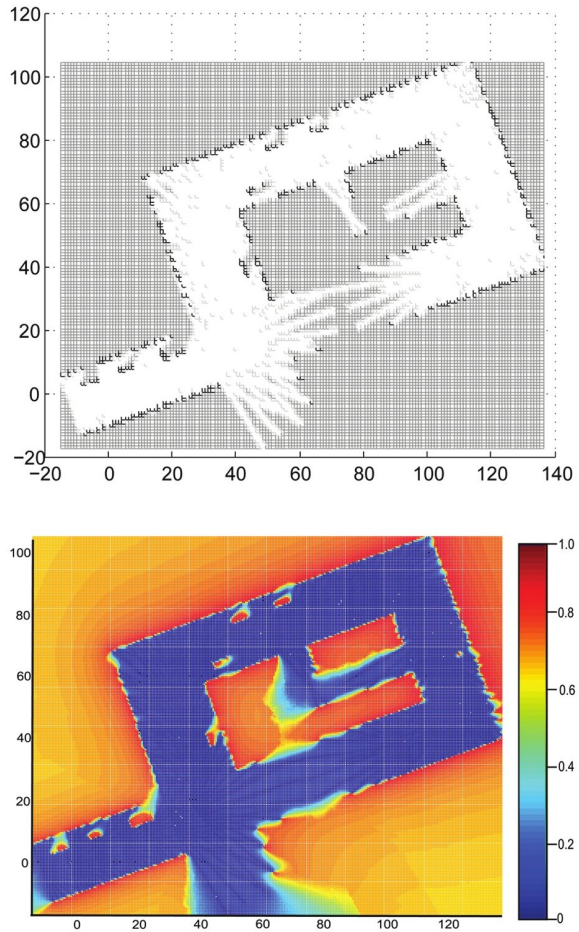
$$m^* = \arg\max_m \ \log p(m|z_{1:t}, x_{1:t})$$

  - Iterative cell flipping approach
  - Intractable for expansive or finely-discretized 3-dimensional grids

Gaussian Process Occupancy Maps

Proposed by O'Callaghan and Ramos [17], learn a model from measurement data, abandon discretization

- Formulate map inference as spatial interpolation between measurements
- Reason about correlations between measurements, not cells
- Map is *continuous*
- Provides uncertainty estimates



Gaussian Process Occupancy Maps

Given:

- Training data $X = \{x_1, x_2, \ldots, x_N \mid x_i \in \mathbb{R}^3\}$ comprised of the 3D coordinates of our "occupied" (sensor hits) and "free" points (sensor ray samples)
- Corresponding labels $y = \{y_1, y_2, \ldots, y_N \mid y_i \in \{-1, 1\}\}$

A Gaussian process is a distribution over functions f:

$$f(x) \sim \mathcal{GP}(m(x),\ k(x, x'))$$

Where we have:

- Mean function $m(x)$, typically taken to be 0
- Covariance function, or kernel, $k(x, x')$, a measure of similarity between x and x'

Goal: Infer latent function values f_* for query points X_*:

$$f_* | X, y, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*))$$

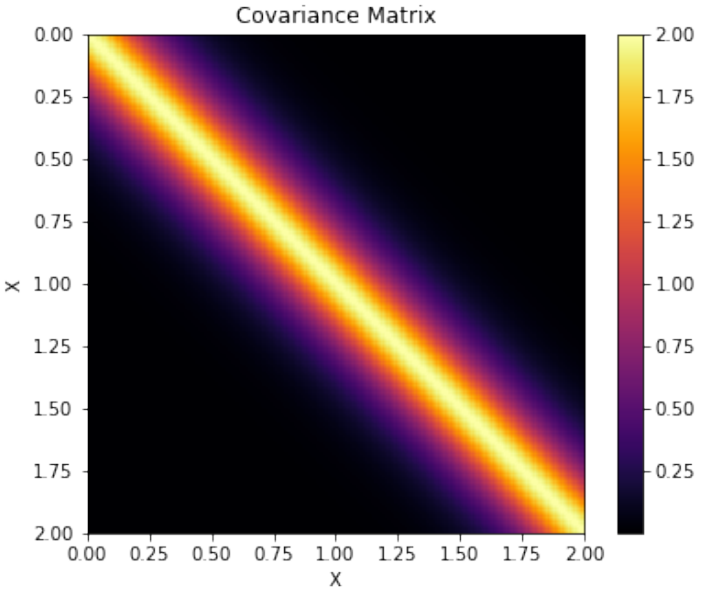
Gaussian Process Occupancy Maps

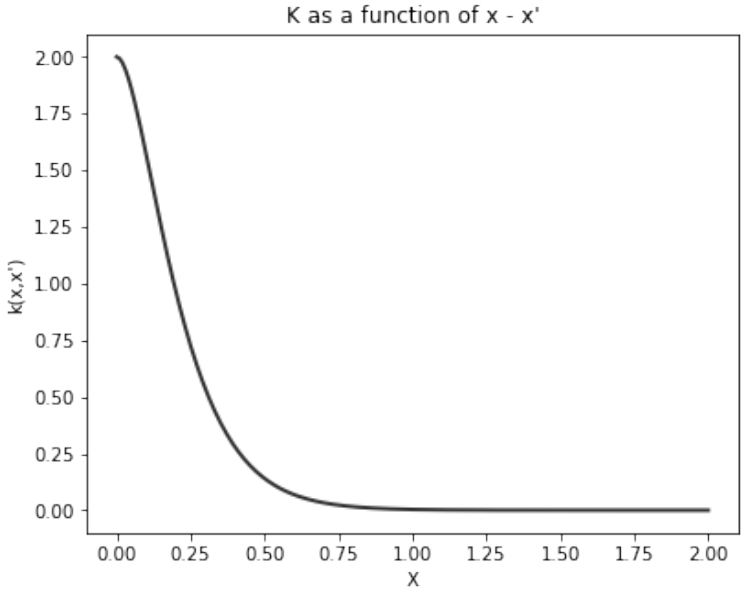
The mean and covariance of the latent function values can be computed as follows:

$$\bar{f}_* = K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} y$$
$$\text{cov}(f_*) = K(X_*, X_*) - K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} K(X, X_*).$$

What's the kernel?

Gaussian Process Occupancy Maps





What's the kernel? Matérn $\nu = 3/2$ is a popular choice:

$$k(x, x') = \sigma_f^2 \left(1 + \frac{\sqrt{3}\|x - x'\|}{l}\right) \exp\left(\frac{-\sqrt{3}\|x - x'\|}{l}\right)$$

σ_f^2 represents the prior signal variance and l controls the length-scale of the kernel

Gaussian Process Occupancy Maps

Obtains a mean and covariance; mean not guaranteed to be in the range [0,1], "squash" GP output to obtain something we can interpret as a probability

$$p(y_{*i} = 1|X, y) = \frac{1}{1 + \exp(-\gamma\omega_{*i})}$$

where γ is a positive constant, $\omega_{*i} = \sigma^2_{min}\mu_{*i}/\sigma^2_{*i}$, and σ^2_{min} is the minimum predicted variance

Gaussian Process Occupancy Maps

We can update the Gaussian process regression model using the Bayesian committee machine [33] as follows:

$$\hat{m}(f_*|\mathcal{D}) = \widehat{\text{cov}}(f_*|\mathcal{D}) \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i)^{-1} m(f_*|\mathcal{D}_i)$$

$$\widehat{\text{cov}}(f_*|\mathcal{D})^{-1} = -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i)^{-1}$$

Gaussian Process Occupancy Maps

Why not just use Gaussian process occupancy maps?

Gaussian Process Occupancy Maps

Why not just use Gaussian process occupancy maps?

- Prediction requires inversion of a matrix the size of the training data; in $\mathcal{O}(N^3)$

Gaussian Process Occupancy Maps

Why not just use Gaussian process occupancy maps?

- Prediction requires inversion of a matrix the size of the training data; in $\mathcal{O}(N^3)$
- Too slow for real-time mapping scenarios

Hilbert Maps

Proposed by Ramos and Ott, same idea as Gaussian process regression, but some modifications

- Nonlinear logistic regression

$$p(y_* = 1|x_*, x, w) = \frac{1}{1 + \exp(w^T k(x_*, x))}$$

- Approximate radial basis function kernel (Nyström method)

$$k(x, x') = \exp(-\gamma \|x - x'\|^2)$$

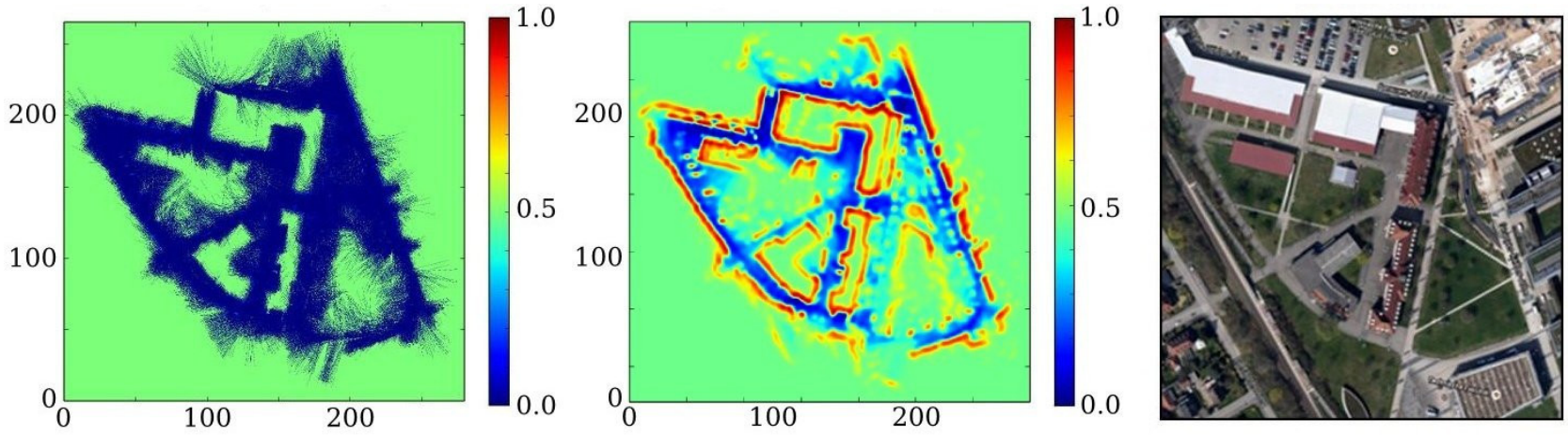
$$\phi_{\text{Nystöm}}(x) = D^{-1/2} V^T (k(x, \hat{x}_1), \ldots, k(x, \hat{x}_m))$$

- Train iteratively with stochastic gradient descent: minimize

$$E(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha \|w\|_1$$

$$L(y_i, f(x_i)) = \log(\exp(-y_i(x_i^T w + b)) + 1)$$

Hilbert Maps



Pros:

- Faster than GPOM (Linear time complexity, but big coefficient attached)
- Comparable accuracy

Cons:

- No uncertainty estimates
- Still intractable for real-time 3D mapping

Table of Contents

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality
- How much bearing should data from several feet/yards/miles have on a query point?

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality
- How much bearing should data from several feet/yards/miles have on a query point?
  - In practice, the kernel produces values that are nearly zero for this data

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality
- How much bearing should data from several feet/yards/miles have on a query point?
  - In practice, the kernel produces values that are nearly zero for this data
  - Don't need to reason about training data that is far away (outside some radius R)

Overview

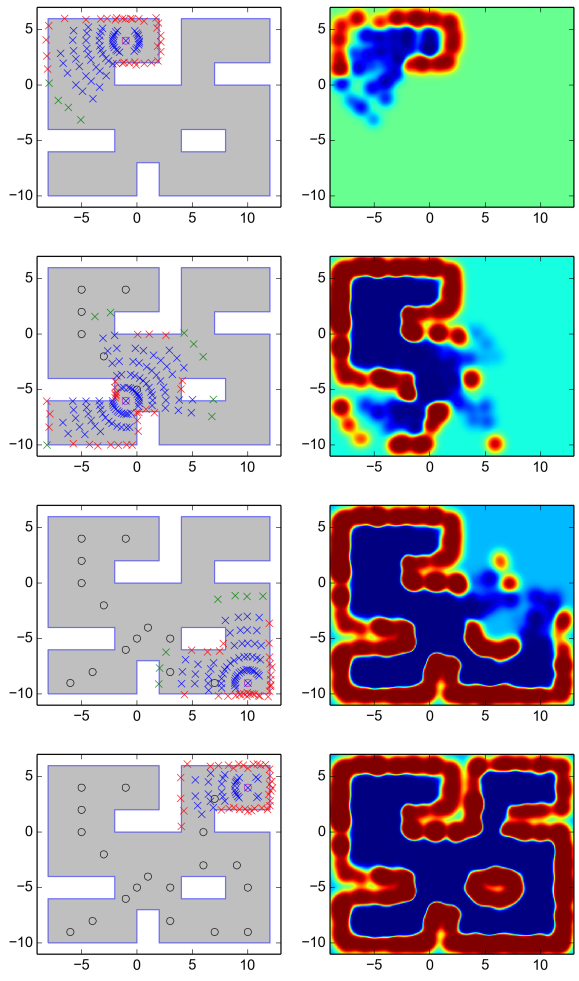
In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality
- How much bearing should data from several feet/yards/miles have on a query point?
  - In practice, the kernel produces values that are nearly zero for this data
  - Don't need to reason about training data that is far away (outside some radius R)
- Train different classifier on each range scan

Overview

In practice, there are a few approximations we can make to speed up computation of Hilbert maps:

- Once we have a trained classifier, where should we query?
  - Solution should emphasize locality
- How much bearing should data from several feet/yards/miles have on a query point?
  - In practice, the kernel produces values that are nearly zero for this data
  - Don't need to reason about training data that is far away (outside some radius R)
- Train different classifier on each range scan
  - **What do we do when a query falls within distance R of multiple classifiers?**



Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$

Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$
- Estimate the probability of occupancy of map cell m_i given the data by combining logistic regression outputs $p(m_i|\mathcal{D}_{1:t})$

Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$
- Estimate the probability of occupancy of map cell m_i given the data by combining logistic regression outputs $p(m_i|\mathcal{D}_{1:t})$

$$p(m_i|\mathcal{D}_{1:t}) = \left[1 + \frac{1 - p(m_i|\mathcal{D}_{1:t-1})}{p(m_i|\mathcal{D}_{1:t-1})} \frac{1 - p(m_i|\mathcal{D}_t)}{p(m_i|\mathcal{D}_t)} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}$$

Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$
- Estimate the probability of occupancy of map cell m_i given the data by combining logistic regression outputs $p(m_i|\mathcal{D}_{1:t})$

$$p(m_i|\mathcal{D}_{1:t}) = \left[1 + \frac{1 - p(m_i|\mathcal{D}_{1:t-1})}{p(m_i|\mathcal{D}_{1:t-1})} \frac{1 - p(m_i|\mathcal{D}_t)}{p(m_i|\mathcal{D}_t)} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}$$

- Doesn't this assume grid cell independence?

Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$
- Estimate the probability of occupancy of map cell m_i given the data by combining logistic regression outputs $p(m_i|\mathcal{D}_{1:t})$

$$p(m_i|\mathcal{D}_{1:t}) = \left[1 + \frac{1 - p(m_i|\mathcal{D}_{1:t-1})}{p(m_i|\mathcal{D}_{1:t-1})} \frac{1 - p(m_i|\mathcal{D}_t)}{p(m_i|\mathcal{D}_t)} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}$$

- Doesn't this assume grid cell independence?
- Spatial dependence accounted for by logistic regression classifier

Probabilistic Local Map Fusion

- Assume discretized grid map $m_i \in \mathcal{M}$
- Estimate the probability of occupancy of map cell m_i given the data by combining logistic regression outputs $p(m_i|\mathcal{D}_{1:t})$

$$p(m_i|\mathcal{D}_{1:t}) = \left[1 + \frac{1 - p(m_i|\mathcal{D}_{1:t-1})}{p(m_i|\mathcal{D}_{1:t-1})} \frac{1 - p(m_i|\mathcal{D}_t)}{p(m_i|\mathcal{D}_t)} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}$$

- Doesn't this assume grid cell independence?
- Spatial dependence accounted for by logistic regression classifier
- We don't *need* to discretize this; treat the logistic regression as spatial interpolation, combine predictions at query time

Combining Logistic Regression Classifiers

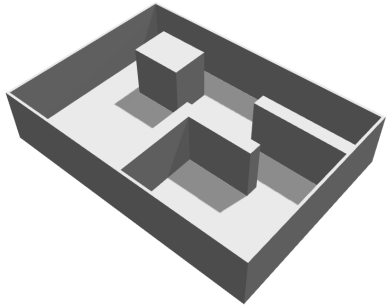
$$
\begin{aligned}
p(m_i|\mathcal{D}_1) &= \frac{1}{1+\exp(w_1^T k(X_1, x_*))} \\
p(\neg m_i|\mathcal{D}_1) &= 1 - p(m_i|\mathcal{D}) = \frac{\exp(w_1^T k(X_1, x_*))}{1+\exp(w_1^T k(X_1, x_*))} \\
p(m_i|\mathcal{D}_{1:2}) &= \left[1 + \frac{1-p(m_i|\mathcal{D}_1)}{p(m_i|\mathcal{D}_1)} \frac{1-p(m_i|\mathcal{D}_2)}{p(m_i|\mathcal{D}_2)} \frac{p(m_i)}{1-p(m_i)}\right]^{-1} \\
&= \left[1 + \exp(w_1^T k(X_1, x_*)) \exp(w_2^T k(X_2, x_*))\right]^{-1} \\
&= \frac{1}{1+\exp(w_{new}^T k(X_{new}, x_*))}
\end{aligned}
$$

where $w_{new} = [w_1, w_2]$ and $X_{new} = [X_1; X_2]$.

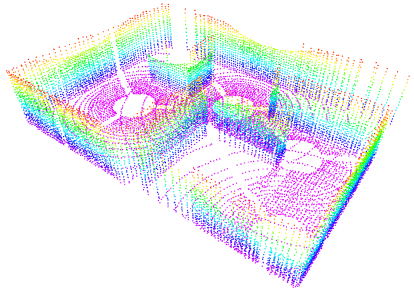
Computational Results

- Tested on 2 simulated datasets: "structured" and "unstructured" in Gazebo [12]
- Time and qualitative map evaluation performed for Freiburg Corridor dataset [34]
- Quantitatively compared our proposed method to our own implementation of Hilbert maps (as it was originally proposed) and OctoMap [7]

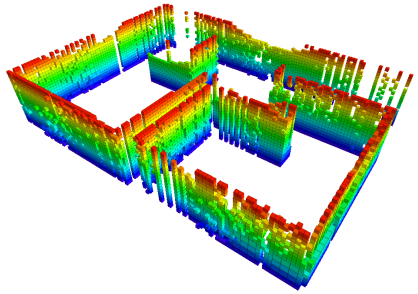
Computational Results



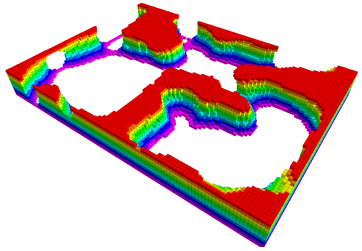
(a) The ground truth map



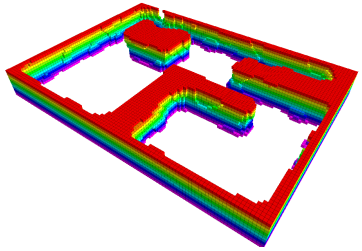
(b) The raw sensor data



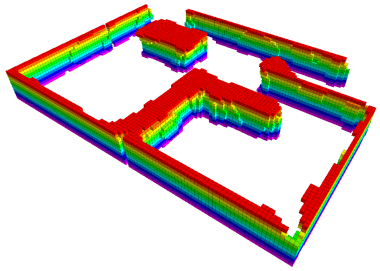
(c) The result of OctoMap



(d) Global HM ($m = 100$)

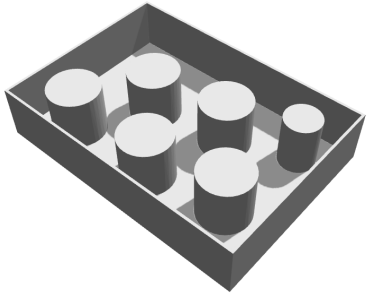


(e) Global HM ($m = 1000$)

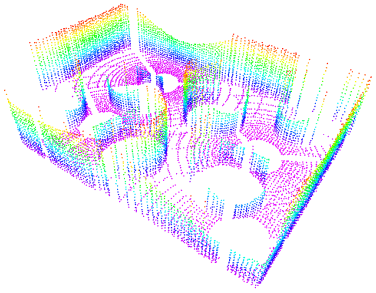


(f) Overlapping HM

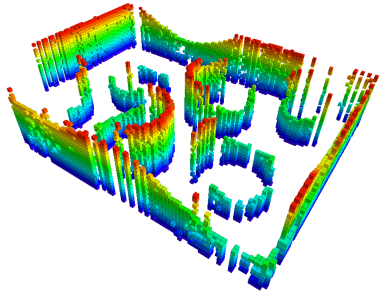
Computational Results



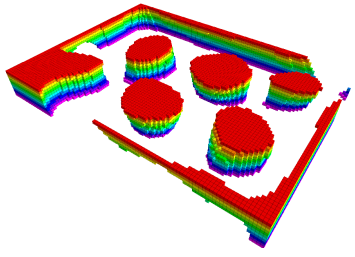
(a) The ground truth map



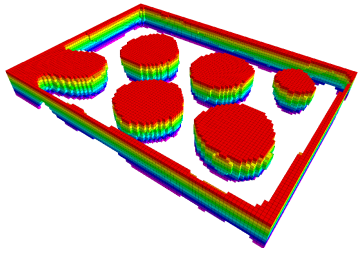
(b) The raw sensor data



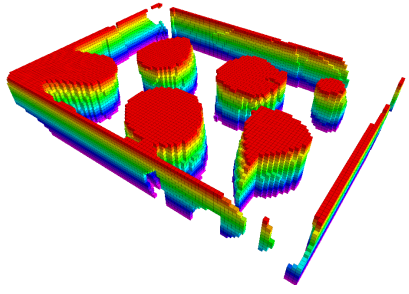
(c) The result of OctoMap



(d) Global Hilbert maps ($m = 100$)

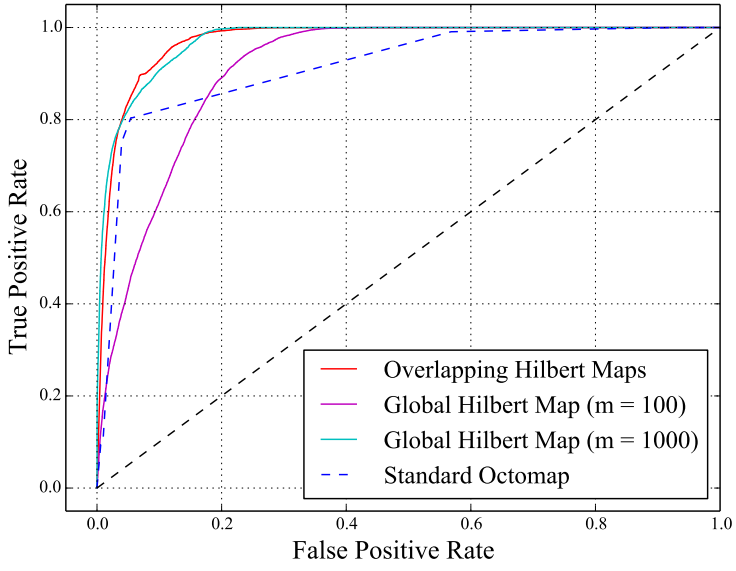


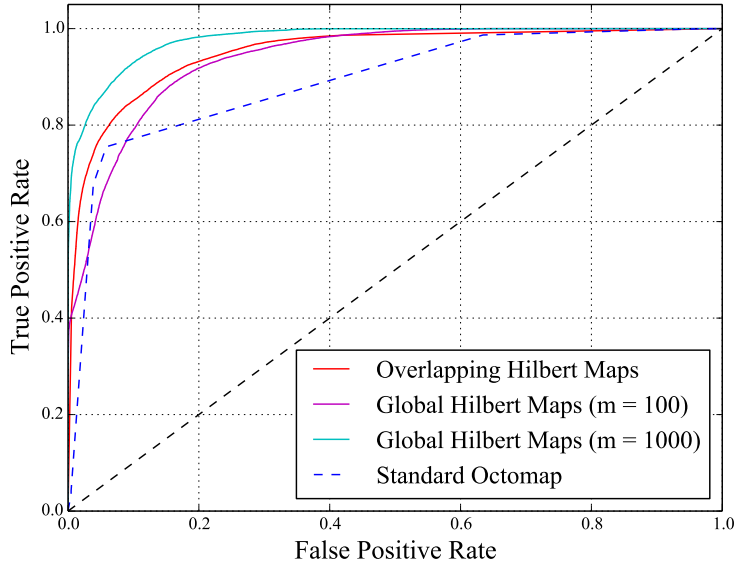
(e) Global Hilbert maps ($m = 1000$)

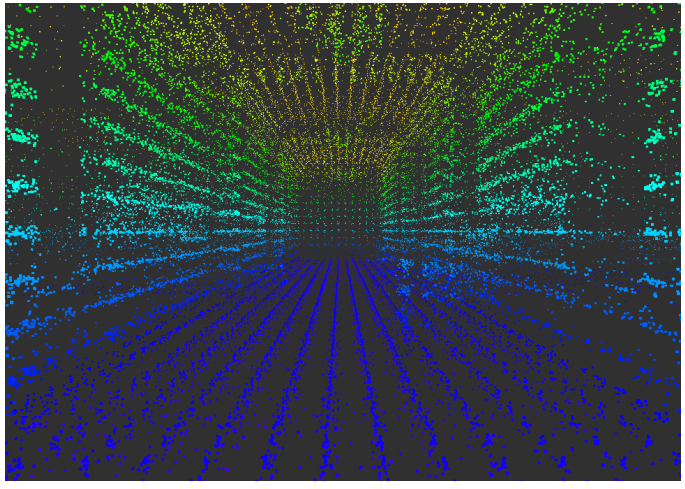


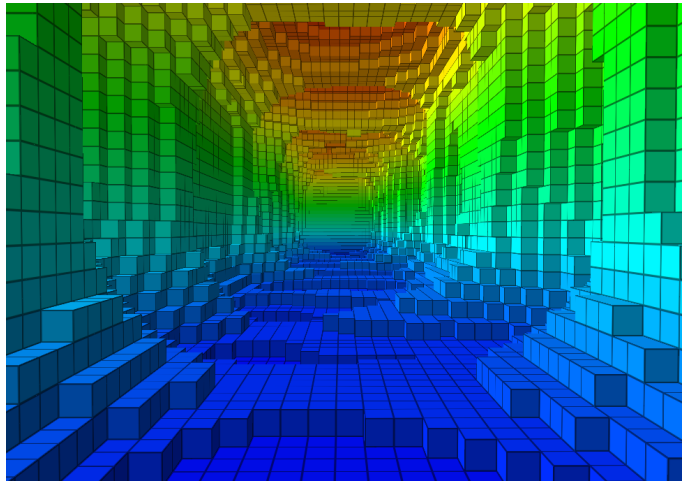
(f) Overlapping Hilbert maps

Computational Results









Computational Results

Dataset	Scans	Pts/Scan	Test Pts/Scan	Method	Time/Scan	Time (s)
Structured	12	3500	29892	OHM	1.89	22.68
				GHM-100	N/A	8.70
				GHM-1000	N/A	331.64
				OctoMap	0.02	0.2
Unstructured	12	3500	29892	OHM	1.83	21.96
				GHM-100	N/A	8.52
				GHM-1000	N/A	332.82
				OctoMap	0.01	0.14
FR-079	66	4943	371170	OHM	14.6	963.6
				GHM-100	N/A	1656.95
				GHM-1000	N/A	11914.96
				OctoMap	0.1	6.7

Summary

We have presented overlapping Hilbert maps

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers
- Faster, but not quite there (!!)

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers
- Faster, but not quite there (!!)
  - Implemented pretty naïvely in Python

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers
- Faster, but not quite there (!!)
  - Implemented pretty naïvely in Python
  - That's an issue

What next?

- How can we do this in real-time?

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers
- Faster, but not quite there (!!)
  - Implemented pretty naïvely in Python
  - That's an issue

What next?

- How can we do this in real-time?
- Do we *really* need to *train* all these models, or are all the features pretty much equally useful?

Summary

We have presented overlapping Hilbert maps

- Local approximation to Hilbert maps
- Combine predictions from multiple classifiers
- Faster, but not quite there (!!)
  - Implemented pretty naïvely in Python
  - That's an issue

What next?

- How can we do this in real-time?
- Do we *really* need to *train* all these models, or are all the features pretty much equally useful?
- Can we better take advantage of the idea of *local* spatial correlations?
- What about prediction uncertainty?

Table of Contents

Overview

Three techniques used:

Overview

Three techniques used:

- Generalized kernel inference approach by Vega-Brown et al. [35]
  - Kernel smoother
  - Reverts to specified prior in regions with limited training data
  - Exact recursive updates

Overview

Three techniques used:

- Generalized kernel inference approach by Vega-Brown et al. [35]
  - Kernel smoother
  - Reverts to specified prior in regions with limited training data
  - Exact recursive updates
- Sparse kernel by Melkumyan and Ramos [13]
  - Enables exact inference with efficient spatial data structures like k-d trees

Overview

Three techniques used:

- Generalized kernel inference approach by Vega-Brown et al. [35]
  - Kernel smoother
  - Reverts to specified prior in regions with limited training data
  - Exact recursive updates
- Sparse kernel by Melkumyan and Ramos [13]
  - Enables exact inference with efficient spatial data structures like k-d trees
- Test data octrees by Wang et al. [36]
  - Fast query-time performance
  - Prune away redundant map cells to reduce memory consumption

Nonparametric Bayesian Kernel Inference

Occupancy is Bernoulli distributed, with parameter $\theta = p(m_i|\mathcal{D}_{1:t})$.

Nonparametric Bayesian Kernel Inference

Occupancy is Bernoulli distributed, with parameter $\theta = p(m_i|\mathcal{D}_{1:t})$.With $\theta \sim Beta(\alpha, \beta)$, the predicted mean and variance of θ are as follows:

$$\mathbf{E}[\theta] = \frac{\alpha}{\alpha + \beta}$$

$$\mathbf{Var}[\theta] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

where α and β are hyperparameters.

Nonparametric Bayesian Kernel Inference

Occupancy is Bernoulli distributed, with parameter $\theta = p(m_i|\mathcal{D}_{1:t})$.With $\theta \sim Beta(\alpha, \beta)$, the predicted mean and variance of θ are as follows:

$$\mathbf{E}[\theta] = \frac{\alpha}{\alpha + \beta}$$

$$\mathbf{Var}[\theta] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

where α and β are hyperparameters.We apply the recursive updates:

$$\alpha_t = \alpha_{t-1} + \bar{y}_t$$
$$\beta_t = \beta_{t-1} + \bar{k}_t - \bar{y}_t$$

$$\bar{y}_t = \sum_{i=1}^{N} k(x_i, x_*)y_i, \;\; \bar{k}_t = \sum_{i=1}^{N} k(x_i, x_*)$$

Nonparametric Bayesian Kernel Inference

Features of the new method:

- α_0 and β_0 represent priors on the positive (occupied) and negative (unoccupied) classes

Nonparametric Bayesian Kernel Inference

Features of the new method:

- α_0 and β_0 represent priors on the positive (occupied) and negative (unoccupied) classes
- When we have lots of nearby training data, approximates Nadaraya-Watson kernel smoother:

$$\hat{m}(x_*) = \frac{\sum_{i=1}^{N} k(x_i, x_*) y_i}{\sum_{i=1}^{N} k(x_i, x_*)}$$

Nonparametric Bayesian Kernel Inference

Features of the new method:

- α_0 and β_0 represent priors on the positive (occupied) and negative (unoccupied) classes
- When we have lots of nearby training data, approximates Nadaraya-Watson kernel smoother:

$$\hat{m}(x_*) = \frac{\sum_{i=1}^{N} k(x_i, x_*) y_i}{\sum_{i=1}^{N} k(x_i, x_*)}$$

- Bonus: We get variances associated with our predictions!

Sparse Kernel

The sparse kernel we used [13] is defined as

$$k(x, x') = \begin{cases} \sigma_0 \left[\frac{2+\cos(2\pi \frac{d}{l})}{3} (1 - \frac{d}{l}) + \frac{1}{2\pi} \sin(2\pi \frac{d}{l}) \right] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases}$$

where $\sigma_0 > 0$ is a constant parameter of the kernel, $l > 0$ is the scale, and d is the distance between x and x'.

Sparse Kernel

The sparse kernel we used [13] is defined as

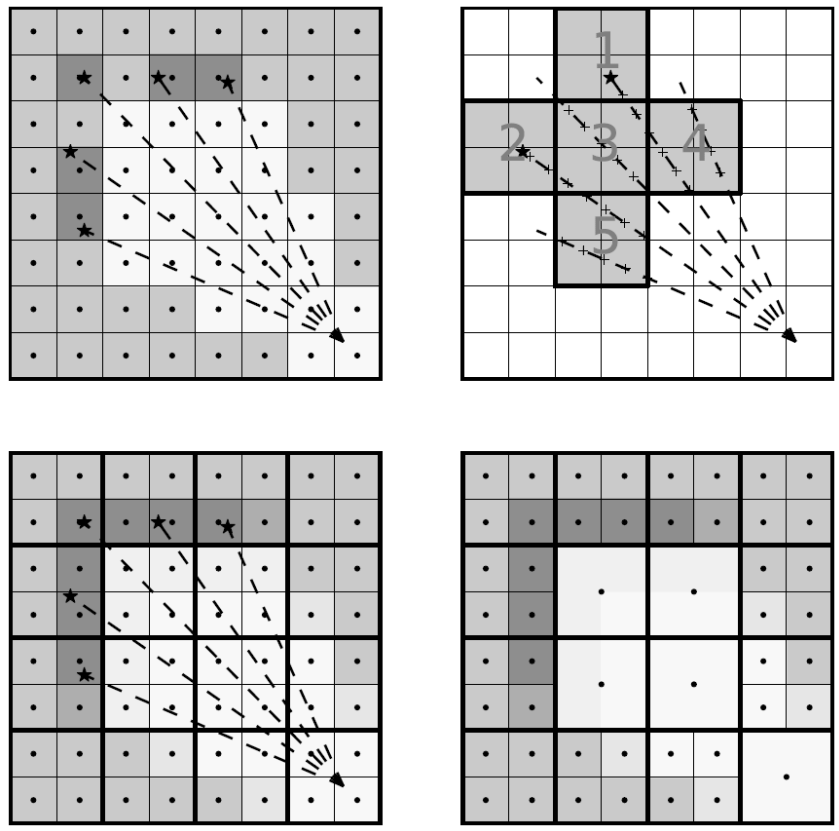
$$k(x, x') = \begin{cases} \sigma_0 \left[\frac{2+\cos(2\pi \frac{d}{l})}{3} (1 - \frac{d}{l}) + \frac{1}{2\pi} \sin(2\pi \frac{d}{l}) \right] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases}$$

where $\sigma_0 > 0$ is a constant parameter of the kernel, $l > 0$ is the scale, and d is the distance between x and x'.

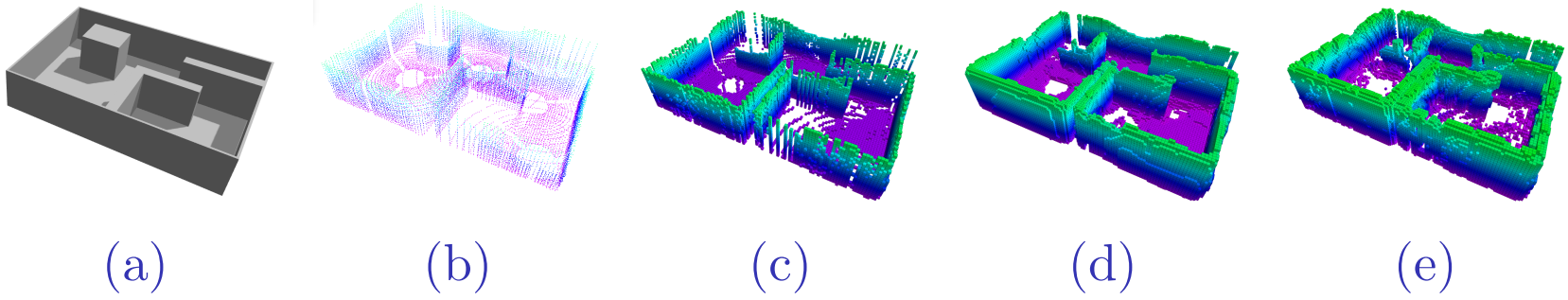
- Efficiently and exactly compute $\bar{y}$ and $\bar{k}$ in $\mathcal{O}(\log N)$ time using a k-d tree
- Overall computational complexity is $\mathcal{O}(M \log N)$, where M is the number of test points and N is the number of training points

Test-data Octrees

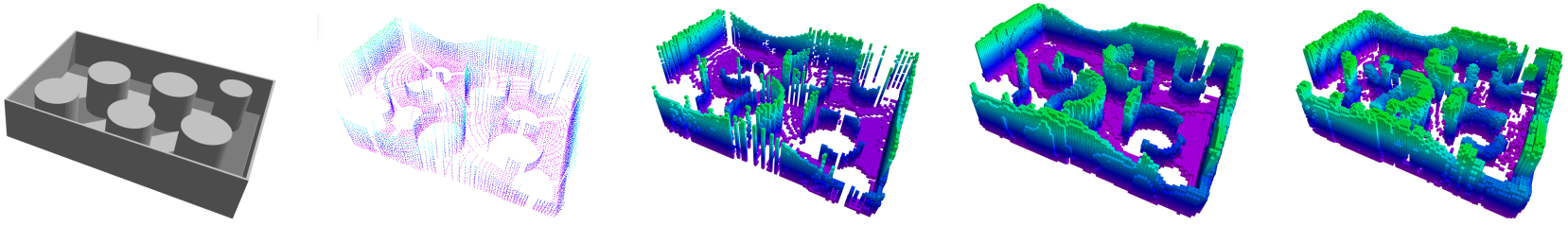
- Consider only points in "extended block"
- Quickly retrieve all relevant grid cells
  - and all relevant training data!
- Prune neighboring grid cells sharing the same state



Computational Results



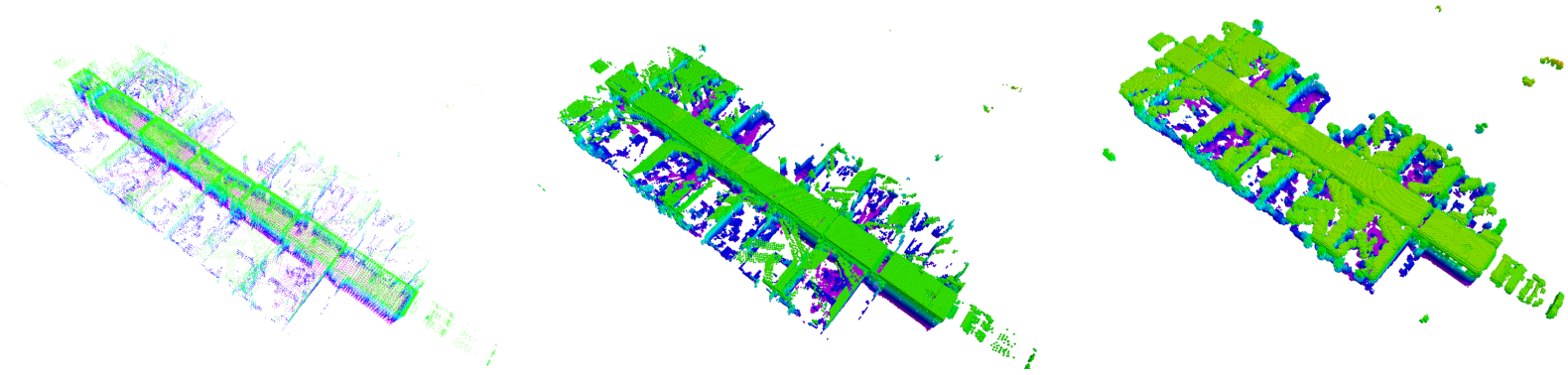
(a) (b) (c) (d) (e)



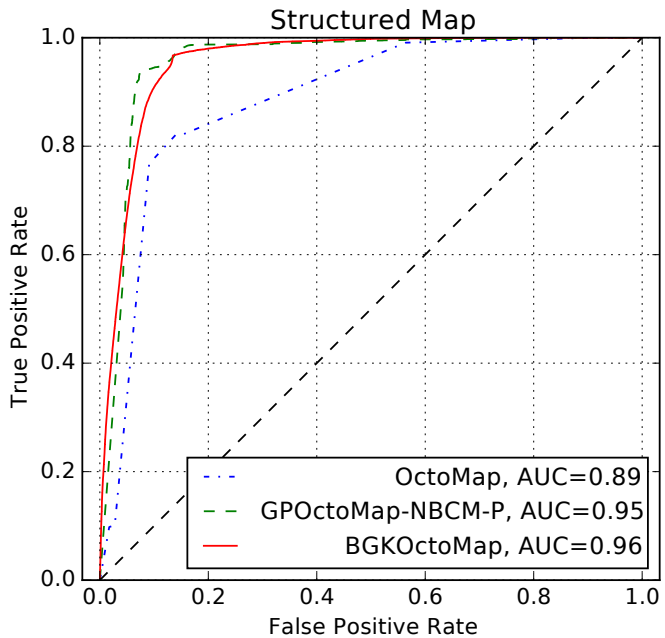
(a) (b) (c) (d) (e)

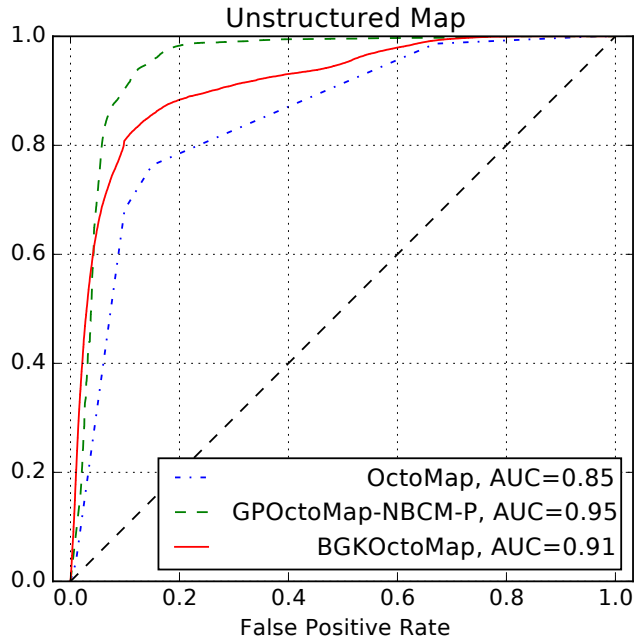
(a) Gazebo model, (b) Raw sensor data (“hit” points),
(c) OctoMap, (d) GPOctoMap, (e) Our method, BGKOctoMap

Computational Results



(a) (b) (c)



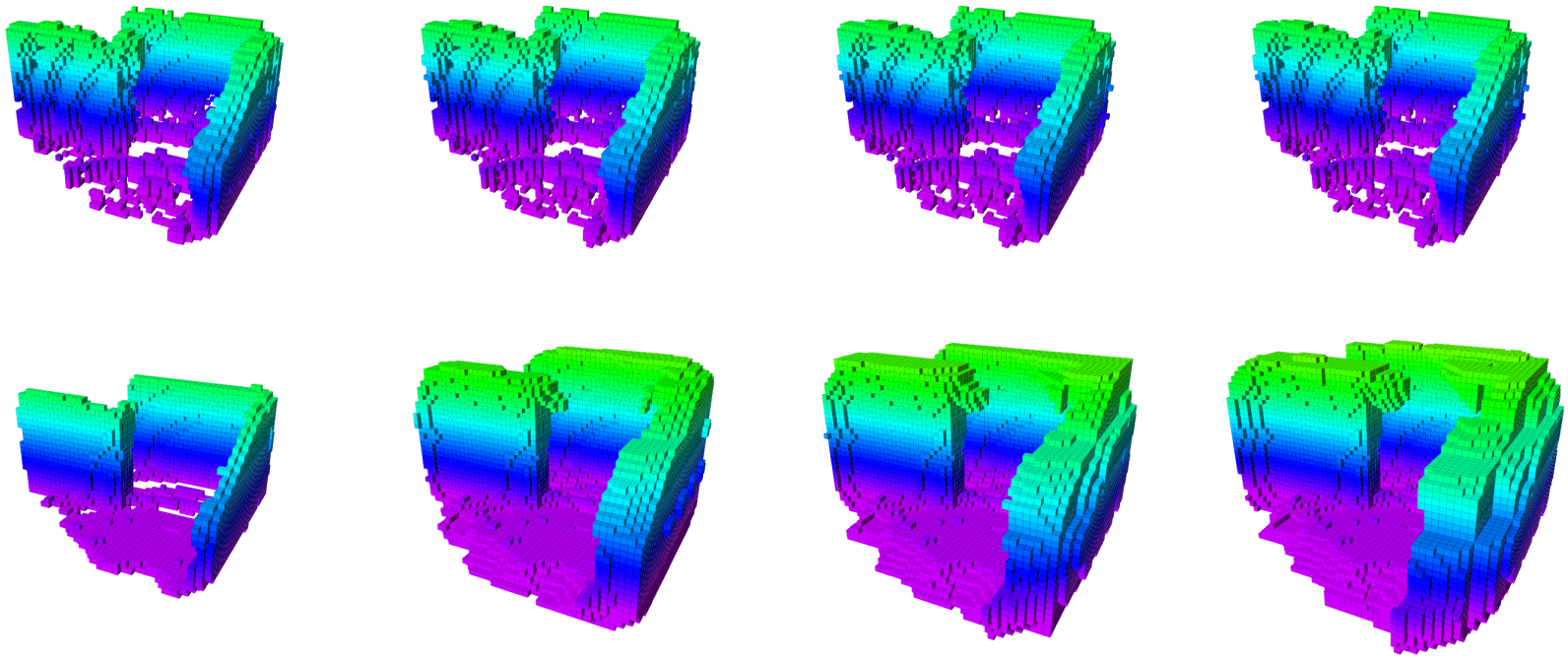


Computational Results

Dataset	Scans	Pts/Scan	Sampled Pts/Scan	Method	Time/Scan	Time (s)
Structured	12	3500	1506	BGK	**0.021**	**0.25**
				GPOM	0.091	1.1
				OctoMap	0.027	0.32
Unstructured	12	3500	1506	BGK	0.019	0.23
				GPOM	0.075	0.90
				OctoMap	**0.018**	**0.22**
FR-079	66	89445	7601	BGK	**0.15**	**9.6**
				GPOM	0.28	18.4
				OctoMap	0.15	10.1

Computational Results

Comparison to BCM updates:



Results of updating both BGKOctoMap (Top) and GPOctoMap-NBCM (Bottom) after the introduction of 1, 15, 30, and 60 scans containing the same data (Left to Right).

Long-term Mapping

BGKOctoMap is stable when mapping the same region

Long-term Mapping

BGKOctoMap is stable when mapping the same region

- Why isn't the BCM?

Long-term Mapping

BGKOctoMap is stable when mapping the same region

- Why isn't the BCM?
  - Provably:

$$\lim_{K \to \infty} p(y_* = 1 | X, y) = \begin{cases} 1, & \text{if } \mu_1 > 0 \\ 0.5, & \text{if } \mu_1 = 0 \\ 0, & \text{if } \mu_1 < 0 \end{cases}$$

in the case of the same data (X, y) input to the BCM K times and "squashed" (derivation in thesis, for the curious)

Recent Work

BGKOctoMap is comparable in speed to GPOctoMap, has attractive long-term mapping benefits

Recent Work

BGKOctoMap is comparable in speed to GPOctoMap, has attractive long-term mapping benefits

- Prediction of GPOctoMap is more accurate

Recent Work

BGKOctoMap is comparable in speed to GPOctoMap, has attractive long-term mapping benefits

- Prediction of GPOctoMap is more accurate
- How can we improve accuracy of BGKOctoMap?

Recent Work

BGKOctoMap is comparable in speed to GPOctoMap, has attractive long-term mapping benefits

- Prediction of GPOctoMap is more accurate
- How can we improve accuracy of BGKOctoMap?
- Previous work used free space samples linearly interpolated along sensor rays

Recent Work

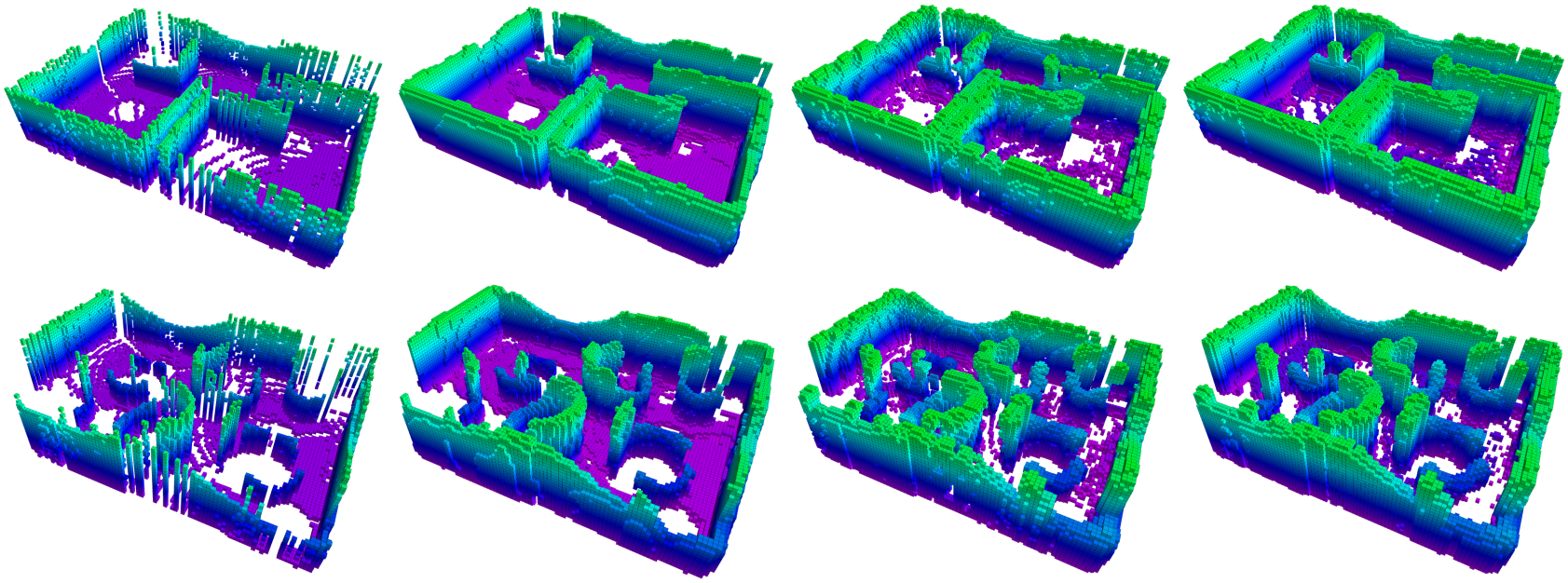
BGKOctoMap is comparable in speed to GPOctoMap, has attractive long-term mapping benefits

- Prediction of GPOctoMap is more accurate
- How can we improve accuracy of BGKOctoMap?
- Previous work used free space samples linearly interpolated along sensor rays
- Our recent work has been done with the point-to-line distance used by O'Callaghan and Ramos [17]:

$$x_{free} = \begin{cases} P, & \text{if } d < 0 \\ P + d\frac{PQ}{|PQ|}, & \text{if } 0 \leq d \leq 1 \\ Q, & \text{if } d > 1 \end{cases}$$

$$d = \frac{PQ \cdot Px_*}{|PQ|}.$$

Recent Work



Preliminary results using point-to-line distance for free-space representation. Left to Right: OctoMap, GPOctoMap, BGKOctoMap, BGKOctoMap using point-to-line distance. Top: Structured environment; Bottom: Unstructured environment.

Summary

Summary of BGKOctoMap

- Uses nonparametric Bayesian kernel inference
  - No explicit training step
  - Uncertainty estimates provided
- Sparse kernels
- Test data octrees
- Reliable long-term performance
- Predictably transitions to a prior “unknown” state of 0.5 mean with high variance when there is no training data

Table of Contents

Summary of Contributions

Two approaches to learning=aided 3D occupancy mapping have been presented:

- Overlapping Hilbert maps
  - Substantial time improvement over Hilbert maps
  - Comparable accuracy to HM
  - Still not real-time ready
  - No uncertainty estimates
- BGK occupancy maps
  - Real-time viable
  - Comparable accuracy to GPOM
  - Provides uncertainty estimates
  - Improved stability mapping same region long-term, compared to GPOM+BCM

Learning-aided 3D Mapping Library (LA3DM) on Github:
`https://github.com/RobustFieldAutonomyLab/la3dm`

Avenues for Future Work

- Dynamic maps: we didn't touch this

Avenues for Future Work

- Dynamic maps: we didn't touch this
- Implementation onboard for underwater mapping!

Avenues for Future Work

- Dynamic maps: we didn't touch this
- Implementation onboard for underwater mapping!
- *Doing things* in these maps
  - Planning
  - Navigation
  - Exploration (use of uncertainty estimates)

Avenues for Future Work

- Dynamic maps: we didn't touch this
- Implementation onboard for underwater mapping!
- *Doing things* in these maps
  - Planning
  - Navigation
  - Exploration (use of uncertainty estimates)
- Learning the kernel length scale
  - Making the kernel length scale dependent on local data

Questions

References I

[1] C.M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3D Hilbert maps," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1011-1018, 2016.

[3] K. Doherty, J. Wang, and B. Englot, "Bayesian Generalized Kernel Inference for Occupancy Map Prediction," *Proceedings of the IEEE International Conference on Robotics and Automation*, Accepted, To Appear, May 2017.

[4] A. Elfes, "Sonar-based real-world mapping and navigation." *IEEE Journal on Robotics and Automation*, vol. 3(3), pp. 249-265, 1987.

References II

[5] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22(6), pp. 46-57, 1989.

[6] V. Guizilini and F. Ramos, "Large-scale 3D scene reconstruction with Hilbert Maps." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3247-3254, 2016.

[7] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34(3), pp. 189-206, 2013.

References III

[8] M. G. Jadidi, J. V. Miro, R. Valencia, and J. Andrade-Cetto, "Exploration on continuous Gaussian process frontier maps," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 6077-6082, 2014.

[9] M. G. Jadidi, J. V. Miro and G. Dissanayake, "Warped Gaussian Processes Occupancy Mapping With Uncertain Inputs," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 680-687, April 2017.

[10] S. Kim and J. Kim, "GPmap: A unified framework for robotic mapping based on sparse Gaussian processes," *Proceedings of the 9th International Conference on Field and Service Robotics*, 2013.

References IV

[11] S. Kim and J. Kim,"Recursive Bayesian Updates for Occupancy Mapping and Surface Reconstruction," *Proceedings of the Australasian Conference on Robotics and Automation*, 2014.

[12] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2149-2154, 2004.

[13] A. Melkumyan and F. Ramos, "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets," *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, vol. 9, pp. 1936-1942, 2009.

References V

[14] H.P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 116-121, 1985.

[15] E.A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9(1), pp. 141-142, 1964.

[16] S. O'Callaghan and F. Ramos, "Continuous occupancy mapping with integral kernels," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1494-1500, 2011.

[17] S. O'Callaghan and F. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31(1), pp. 42-62, 2012.

References VI

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python." *The Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[19] V. Peretroukhin, W. Vega-Brown, N. Roy, and J. Kelly, "PROBE-GK: Predictive robust estimation using generalized kernels," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 817-824, 2016.

[20] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." *Advances in large margin classifiers*, 10(3) pp. 61-74, 1999.

References VII

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, 2009.

[22] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," *Proceedings of Robotics: Science and Systems*, 2015.

[23] C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for Machine Learning*, Cambridge, MA: The MIT Press, 2006.

[24] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian Learning for Safe High-Speed Navigation in Unknown Environments," *Proceedings of the 17th International Symposium on Robotics Research*, 2015.

References VIII

[25] R.B. Rusu and S. Cousins, "3D is here: Point cloud library (pcl)," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1-4, 2011.

[26] B. Scholkopf and A.J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, 2001.

[27] R. Senanayake, L. Ott, S. O'Callaghan, F.T. Ramos, "Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments," *Advances in Neural Information Processing Systems*, 2016.

[28] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 9-15, 2012.

References IX

[29] S. Srivastava and N. Michael, "Approximate continuous belief distributions for precise autonomous inspection," *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 74-80, 2016.

[30] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence* 99.1 (1998): 21-71.

[31] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots* 15.2 (2003): 111-127.

[32] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, Cambridge, MA: The MIT Press, 2005.

[33] V. Tresp, "A Bayesian Committee Machine, *Neural Computation*, vol. 12(11), pp. 2719-2741, 2000.

References X

[34] University of Freiburg OctoMap 3D Scan Dataset http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/

[35] W.R. Vega-Brown, M. Doniec, and N.G. Roy, "Nonparametric Bayesian inference on multivariate exponential families," *Advances in Neural Information Processing Systems*, pp. 2546-2554, 2014.

[36] J. Wang and B. Englot, "Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1003-1010, 2016.

References XI

[37] J. Wang, S. Bai, and B. Englot "Underwater Localization and 3D Mapping of Submerged Structures with a Single-Beam Scanning Sonar," *Proceedings of the IEEE International Conference on Robotics and Automation*, Accepted, To Appear, May 2017.

[38] G.S. Watson, "Smooth regression analysis," *Sankhya: The Indian Journal of Statistics, Series A*, vol. 26(4), pp. 359-372, 1964.

[39] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," *Proceedings of Neural Information Processing Systems*, 2000.