BAYESIAN METHODS FOR MULTI-OBJECT RECONSTRUCTION WITH HILBERT MAP REPRESENTATIONS

by

Herbert Wright

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Kalhert School of Computing

Approved:

Tucker Hermans Thesis Faculty Supervisor	Mary Hall Chair, Kalhert School of Computing
Tom Henderson Honors Faculty Advisor	Monisha Pasupathi, PhD Dean, Honors College

Month Year

ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam hendrerit mauris erat, ac efficitur mi porta eu. Vestibulum ultricies molestie augue eget tempus. Morbi sodales nulla ex, fringilla imperdiet dui elementum nec. Vestibulum sit amet quam laoreet leo ullamcorper pretium. Aenean est metus, lobortis non eros a, semper ultricies ipsum. Cras ac lorem sit amet mi ultricies feugiat ac id sem. Duis ultricies lacus nec nulla posuere hendrerit.

TABLE OF CONTENTS

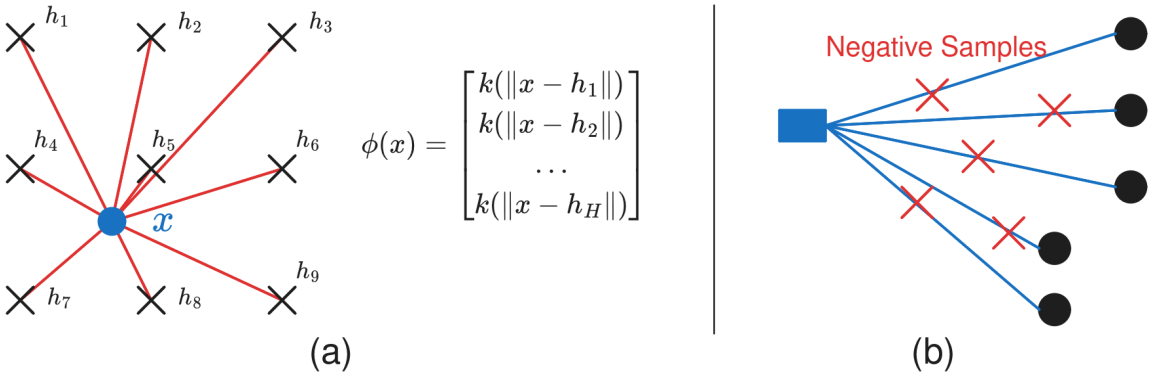

Figure 1: (a) A *hinge point* feature transform induced by a set of hinge points is used by Hilbert maps [1]; (b) these maps are built by first sampling *negative samples* along the unoccupied portions of the camera ray.

1 INTRODUCTION

2 RELATED WORKS

3 BACKGROUND

3.1 Hilbert Maps

Hilbert Maps: Introduced in [1], Hilbert maps are a method for building an occupancy map of an environment given depth observations.

Bayesian Hilbert Maps: Hilbert Maps were extended to the Bayesian setting in [2]. Instead of an individual weight vector, the weight is treated as a normally distributed random variable, $\mathbf{w} \sim P(\mathbf{w})$. Variational Bayesian logistic regression as described in[3] is then performed over data $D = \{(\phi(\mathbf{x}_i), y_i)\}_{i \in [n]}$ in order to obtain the approximate posterior distribution:

$$\hat{P}(\mathbf{w}|D) \propto Q(D|\mathbf{w}; \xi)P(\mathbf{w}) \approx P(D|\mathbf{w})P(\mathbf{w}),$$

where the variational parameter ξ is introduced. The method relies on an EM algorithm that alternates between calculating the posterior $\hat{P}(\mathbf{w}|D) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ from an approximate likelihood function and obtaining a better likelihood approximation. The specific approxi-

mation used for the likelihood takes the form of a normal distribution, and ensures that the approximated likelihood is conjugate to a normal prior $P(\mathbf{w}) = \mathcal{N}(\bar{\mu}, \bar{\Sigma})$.

Once the posterior weight distribution is obtained, the map m is defined by the expectation:

$$m(\mathbf{x}) = \mathbb{E}_{\mathbf{w}}[\sigma(\mathbf{w}^\top \phi(\mathbf{x}))].$$

Because there is not an analytic solution for this expectation, approximations are used. The most common approximation is

$$\mathbb{E}_{\mathbf{w}}[\sigma(\mathbf{w}^\top \phi(\mathbf{x}))] \approx \sigma\left(\frac{\mathbb{E}_{\mathbf{w}}[\mathbf{w}^\top \phi(\mathbf{x})]}{\sqrt{1 + \frac{\pi}{8}\text{Var}(\mathbf{w}^\top \phi(\mathbf{x}))}}\right), \tag{1}$$

which is easily obtained for any $\mathbf{w}$ following a normal distribution.

Extensions of BHMs include Bayesian treatment of kernel parameters and hinge point placement [4], fusing two BHMs [5], and mapping environments with moving actors [6].

3.2 Stein Variational Gradient Descent

4 VOLUMETRIC, PROBABILISTIC, AND ROBUST SEGMENTATION MAPS

This section introduces the V-PRISM method, which stands for **V**olumetric, **P**robabilistic, and **R**obust **I**nstance **S**egmentation **M**aps. In sec. 4.1, I provide a formulation for the multi-object reconstruction problem and give a brief overview of the method. The method is further explained in sec. 4.2 and sec. 4.3. Then, sec. 4.4 details experiments on the V-PRISM method.

4.1 Overview of Method

Problem Formulation. Instead of predicting an occupancy map for each object, we phrase our problem as a multi-class mapping problem. This ensures that each point in space can only be occupied by a single object. Without this constraint, reconstructions of objects can

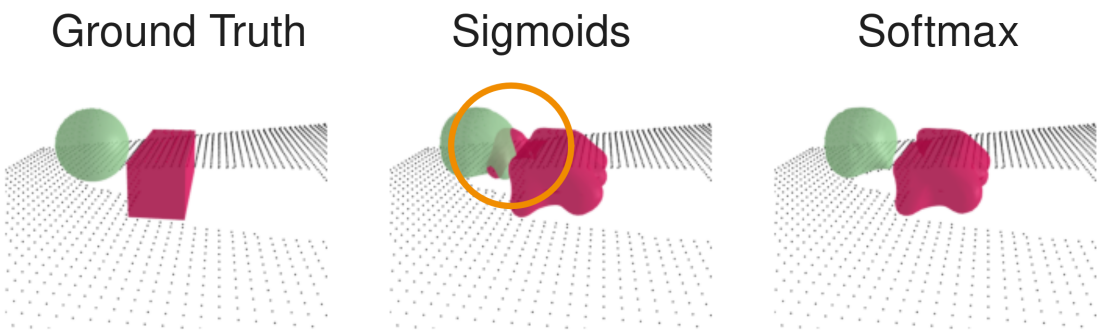

Figure 2: Running a separate sigmoid model per object can cause unwanted intersections between the reconstructions (circled). Our multi-class formulation uses a softmax model that avoids this problem

intersect each other as shown in fig. 2. Formally, we receive observations $\{(\mathbf{x}_i, y_i)\}_{i\in[n]}$ where $\mathbf{x}_i \in \mathbb{R}^d$ corresponds to an observed point with segmented class $y_i \in [c]$. We assume $y_i = 1$ denotes $\mathbf{x}_i$ being segmented to no specific object and is part of the background or table. We also assume that these observations came from a camera with a known location $\mathbf{o} \in \mathbb{R}^d$. The goal is to build a map function $m : \mathbb{R}^d \to [0, 1]^c$ such that $m(\mathbf{x})$ corresponds to the probability distribution over classes that the point $\mathbf{x}$ could belong to.

We would like our map to satisfy that $m(\mathbf{x}_i) \approx \mathbf{e}_{y_i}$ for all i, where $\mathbf{e}_{y_i}$ is the one hot encoding of y_i. We can infer that for any $\mathbf{x}_i$, because the camera ray started at $\mathbf{o}$ and terminated at $\mathbf{x}_i$, all points in between are unoccupied. We would like our map to reflect this realization. This forms the basis for the negative sampling performed in [2]. We will also assume that objects in the scene are resting on or above a planar surface. While this typically means a table, our method is agnostic to the type of surface.

Method Description. Our method builds a map $m(\mathbf{x})$ from segmented camera depth observations of a multi-object scene through two main steps. A high level overview is displayed in fig. 3. First, negative sampling is performed as described in sec. 4.3, where additional points are added to the observed ones in order to form a new labelled point cloud. During this step, the RANSAC [7] algorithm is run in order to recover the surface plane the ob-

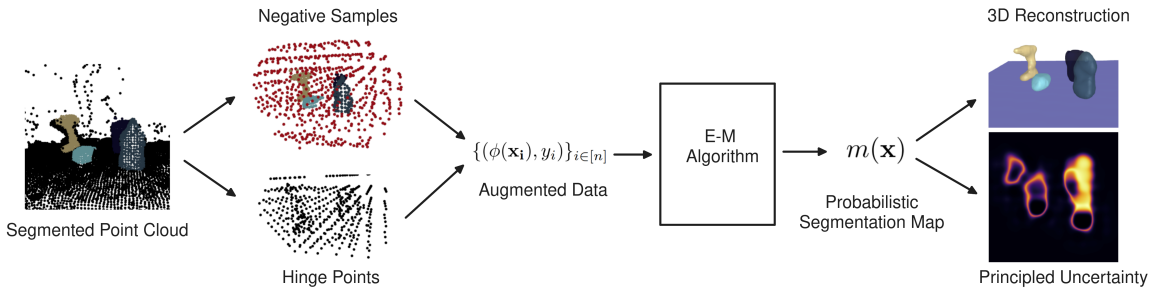

Figure 3: Overview of V-PRISM method

jects are resting on. The points are also subsampled in order to increase efficiency. We then generate a set of hinge points that are used to construct a feature transform according to **??**. This transform, along with our sampled points, creates a set of augmented data.

Once we have our transformed data, we perform Bayesian multi-class regression over the data with an expectation maximization (EM) algorithm. The specific technique makes use of mathematical ideas from [8]. The full EM algorithm and model are explored in sec. 4.2. Efficiently evaluating $m(\mathbf{x})$ for query $\mathbf{x}$ values is also covered in sec. 4.2, where we make use of an approximation proposed in [9]. The segmentation map produced maps each point in 3D space to a distribution over c classes, where one class denotes not belonging to an object and the other $c-1$ classes denote the segmented objects observed.

Once we have our map, we can use it to evaluate how likely different points are to be in occupied by different objects. This is useful in many motion planning algorithms in order to minimize unwanted collisions. We can also reconstruct the meshes of each object by running the marching cubes algorithm [10]. These meshes can be used to create a signed distance function, simulate physics, or to visualize the scene. Our map also encodes principled uncertainty about the geometry of the scene which can be used for active inference.

4.2 Softmax EM Algorithm

Training. To create a Bayesian multi-class map, we consider using a weight matrix $\mathbf{W} \in \mathbb{R}^{c \times m}$ where each row is normally distributed, giving the following likelihood function:

$$P(y = k|\mathbf{W}, \mathbf{x}) = \text{softmax}(\mathbf{W}\phi(\mathbf{x}))_k,$$

where the softmax function is defined as

$$\text{softmax}(\mathbf{W}\phi(\mathbf{x}))_k = \frac{\exp(\mathbf{W}\phi(\mathbf{x})_k)}{\sum_{i=1}^{c} \exp(\mathbf{W}\phi(\mathbf{x})_i)}.$$

Because a conjugate prior for the softmax likelihood doesn't exist, we must use variational inference to find a posterior Gaussian distribution. In our case, we will maximize a lower bound on the likelihood. A useful inequality for this is given in [8], and is stated in the following theorem:

Theorem 1: From [8]. Let $\mathbf{z} \in \mathbb{R}^c$, $\alpha \in \mathbb{R}$, and $\xi \in \mathbb{R}^c_+$. Then the following inequality holds:

$$\ln \sum_{k=0}^{c} \exp(\mathbf{z}_k) \leq \alpha + \sum_{k=0}^{c} \frac{\mathbf{z}_k - \alpha - \xi_k}{2} + \lambda(\xi_k)((\mathbf{z}_k - \alpha)^2 - \xi_k^2) + \ln(1 + \exp(\xi_k)),$$

where $\lambda(\xi_k) = ((1 + \exp(-\xi_k))^{-1} - (1/2))/2\xi_k$.

Applying Theorem 1 to $\mathbf{z} = \mathbf{W}\phi(\mathbf{x})$, we can bound the likelihood by introducing the two variational parameters α and ξ with the inequality,

$$\ln P(y = k|\mathbf{W}, \mathbf{x}) \geq \ln Q(y = k|\mathbf{W}, \mathbf{x}; \alpha, \xi).$$

We can maximize this lower bound and use it as an approximation to the true likelihood by

solving the following:

$$\arg\max_{\alpha,\xi} \mathbb{E}_{\mathbf{W}} \left[\ln Q(y = y_i|\mathbf{x}_i, \mathbf{W}; \alpha, \xi)\right].$$

This can be analytically solved for $\mathbf{W}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$, yielding the following optimal values found in [8]:

$$\alpha_i = \frac{\frac{1}{2}(\frac{c}{2} - 1) + \sum_{k=1}^{c} \lambda(\xi_k)\mu_k^\top \phi(\mathbf{x}_i)}{\sum_{k=1}^{c} \lambda(\xi_k)}, \tag{2}$$

$$\xi_{i,k}^2 = \phi(\mathbf{x}_i)^\top \Sigma_k \phi(\mathbf{x}_i) + (\mu_k^\top \phi(\mathbf{x}_i))^2 + \alpha_i^2 - 2\alpha_i \mu_k^\top \phi(\mathbf{x}_i). \tag{3}$$

Due to the inequality used, $P(y = k|\mathbf{W}, \mathbf{x}; \alpha, \xi)$ is normally distributed for any α, ξ and will be conjugate to our prior weight distribution. Thus, we have a closed-form for the approximate posterior distribution, $P(\mathbf{W}|y = k, \mathbf{x}) = \mathcal{N}(\hat{\mu}, \hat{\Sigma})$. The update equations mirror those found in [8] and are as follows:

$$\hat{\Sigma}_k^{-1} = \bar{\Sigma}^{-1} + 2\sum_{i=1}^{n} \lambda(\xi_{i,k})\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \tag{4}$$

$$\hat{\mu}_k = \hat{\Sigma}_k \left[\bar{\Sigma}_k^{-1}\bar{\mu}_k + \sum_{i=1}^{n} \left(y_{i,k} - \frac{1}{2} + 2\alpha_i \lambda(\xi_{i,k})\right)\phi(\mathbf{x}_i)\right]. \tag{5}$$

We can use eq. 2, eq. 3, eq. 4, and eq. 5 to create an EM algorithm to iterate between calculating our posterior distribution and optimizing our variational parameters, shown in fig. 4. The size of $\hat{\Sigma}_k$ scale quadratically with the feature dimension.

Inference. In order to make predictions about new points we need to evaluate the following expectation:

$$\hat{P}(y = k|\mathbf{x}) = \mathbb{E}_{\mathbf{W}} \left[\text{softmax}(\mathbf{W}\phi(\mathbf{x}))\right]_k. \tag{6}$$

There is not a closed form solution to this expectation, so we must approximate it. While we could use sampling to estimate the expectation, we instead use a more computationally efficient approximation.

Algorithm 1 V-PRISM

Input:

Observed, segmented points $o = \{(\mathbf{x}_i, y_i)\}_{i \in [n']}$

Prior means $\{\bar{\mu}_k\}_{k \in [c]}$ and covariances $\{\bar{\Sigma}_k\}_{k \in [c]}$

$\mathcal{D} \leftarrow \text{NEGATIVESAMPLE}(o)$

$\phi \leftarrow \text{HINGEPOINTTRANSFORM}(o)$

$\xi_{i,k} \leftarrow 1$ for $i \in [m], k \in [c]$

$\alpha_i \leftarrow 0$ for $i \in [m]$

for p iterations **do**

  $\hat{\Sigma}^{-1} \leftarrow \bar{\Sigma}^{-1} + 2\sum_i |\lambda(\xi_i)| \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top$

  $\hat{\mu}_k \leftarrow \hat{\Sigma}\left(\bar{\Sigma}^{-1}\bar{\mu} + \sum_i (y_i - \frac{1}{2} + 2\alpha_i \lambda(\xi_{i,k}))\phi(\mathbf{x}_i)\right)$

  $\alpha_i \leftarrow \text{UPDATEALPHA}(\xi_i, \mathbf{x}_i, \hat{\mu}, \hat{\Sigma})$

  $\xi_{i,k} \leftarrow \text{UPDATEXI}(\alpha_i, \mathbf{x}_i, \hat{\mu}, \hat{\Sigma})$

end for

return $\hat{\mu}, \hat{\Sigma}$

Figure 4: Psuedo-code for the V-PRISM algorithm

As described in [9], we can write the softmax in terms of the sum of sigmoidal terms with the following equality:

$$\text{softmax}(\mathbf{a})_k = \frac{1}{2 - c + \sum_{i \neq k} \sigma(\mathbf{a}_k - \mathbf{a}_i)^{-1}},$$

where c is the number of classes. This is then used as motivation for the approximating the expectation with

$$\mathbb{E}_{\mathbf{W}}\left[\text{softmax}(\mathbf{W}\phi(\mathbf{x}))\right]_k \approx \frac{1}{2 - c + \sum_{i \neq k} \mathbb{E}[\sigma(\tilde{\mathbf{z}}_i)]^{-1}},$$

with $\tilde{\mathbf{z}}_i = [\mathbf{W}\phi(\mathbf{x})]_k - [\mathbf{W}\phi(\mathbf{x})]_i$. When combined with the sigmoidal approximation in Equation (1), this becomes an easily computable approximation to Equation (6).

4.3 Negative Sampling

Similar to many mapping methods, V-PRISM requires sampling negative unoccupied points along depth camera rays. The traditional negative sampling used, mentioned in sec. 3.1, is meant for mapping environments where the robot is in an enclosed space and each camera ray is detecting a wall or sufficiently large object. This sampling performs poorly when the goal is to map a relatively small object resting on a tabletop or other surface. To fully utilize the tabletop structure within the environment, we propose a new negative sampling method designed for object-centric mapping. Our sampling method rests on two main realizations:

1. Along the ray, negative samples are most useful when near known objects.
2. Points below a surface plane cannot be occupied by objects resting entirely on or above that surface.

We assume we have a segmented point cloud of the scene $\{(\mathbf{x}_i, y_i)\}_{i\in[n']}$ where each y_i corresponds to the segmentation label of the respective $\mathbf{x}_i$. We also assume a known position of the camera $\mathbf{o}$. Our sampling method begins by finding the center of the smallest axis-aligned bounding box that contains all of the segmented points for each individual object in the scene. We denote these centers with $\mathbf{o}_k$. We then perform stratified uniform sampling along each ray, only keeping points that are within r_{obj} distance from at least one $\mathbf{o}_k$. Sampled points within the desired radius of a center are labeled as unoccupied and added to the collection of points for the algorithm.

Next, we run RANSAC [7] on the observed point cloud to recover the table plane. Once we have the plane, we uniformly randomly sample points within r_{obj} from each object center and keep any such points that fall below the plane. These points are labelled as unoccupied and added to our collection.

Finally, we perform grid subsampling as described in [11] with each label in parallel in order to reduce the number of points our algorithm is fed. In practice, we choose different

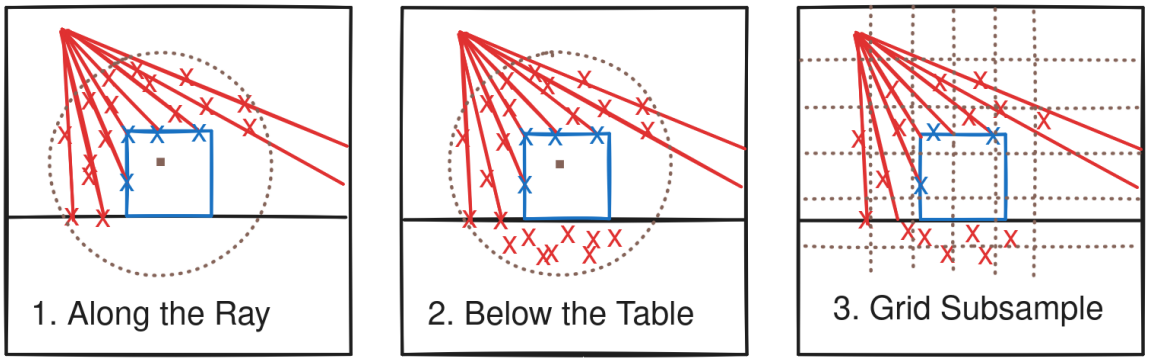

Figure 5: Overview of our sampling method. 1. We perform stratified sampling along camera rays within r_{obj} of the object. 2. Points are sampled below the table within r_{obj} of the object. 3. Grid subsampling is performed.

resolutions to subsample empty points and points on object surfaces. This can dramatically increase the efficiency of our method by removing redundant points. The entire negative sampling process is shown in fig. 5.

The resulting points are then transformed to construct our set of augmented data. The transform used is induced by a set of hinge points according to **??**. In practice, we choose a set of hinge points consisting of a fixed grid around the scene as well as a fixed number of random points sampled from the surface points of each object.

4.4 Experiments

We perform experiments aimed to answer the following questions: (1) Does our method result in accurate reconstructions? (2) Does our sampling method improve map quality for object-centric mapping? (3) Is our method robust to unknown, noisy scenes? 4. Does our map accurately capture uncertainty about the scene geometry? We implement V-PRISM in PyTorch and run our algorithm on an NVIDIA GeForce RTX 2070 GPU.

Baselines: We compare our method to two different baselines. The first is a voxel-based heuristic that labels observed unoccupied voxels as unoccupied, observed occupied voxels as their corresponding segmentation label, and unobserved voxels with the same label as the nearest observed voxel. To prevent incorrect predictions below the table plane, we also run

Method	ShapeNet Scenes		YCB Scenes		Objaverse Scenes	
	IoU ↑	Chamfer ↓	IoU ↑	Chamfer ↓	IoU ↑	Chamfer ↓
Voxel	0.198	0.014	0.324	0.018	0.336	0.024
PointSDF	**0.360**	**0.010**	0.460	0.015	0.347	0.025
V-PRISM	0.309	0.011	**0.500**	**0.012**	**0.464**	**0.018**

Table 1: Quantitative experiments comparing our method to two baseline methods on procedurally generated scenes from benchmark mesh datasets.

RANSAC during our baseline and label all voxels under the plane as unoccupied. We refer to this approach as the **Voxel** baseline. The second baseline is a learning-based approach using a state of the art neural network architecture for continuous object reconstructions in robotics. We take the PointSDF architecture from [12] and replace the final activation with a sigmoid function to predict occupancy probabilities. We train this model on a dataset of scenes generated in simulation. The scenes are composed of a subset of the ShapeNet [13] dataset. Training it on these scenes instead of the original dataset PointSDF was trained on allows it to better function under occlusion and different scales. We refer to this baseline as **PointSDF**.

Metrics: We use two main metrics for comparison: **intersection over union (IoU)** and **Chamfer distance**. IoU is calculated by evaluating points in a fixed grid around each object. Chamfer distance is calculated by first reconstructing the predicted mesh by running the marching cubes algorithm [10] on a level set of $\hat{P}(y = 1|x) = \tau$ for a chosen τ of the prediction function. Then, points are sampled from both the predicted mesh and ground truth mesh and the Chamfer distance is calculated between these two point clouds.

Procedurally Generated Scenes: we evaluate our method against the two baseline methods on procedurally generated scenes, from large object datasets. We generate a scene by randomly picking a mesh and placing it at a random pose within predefined bounds with a random scale. We draw meshes from the ShapeNet [13], YCB [14], and Objaverse [15] datasets. We generate 100 scenes for each dataset with up to 10 objects in each scene. Objects are placed relatively close together in order to ensure significant occlusion in the

Method	ShapeNet Scenes		YCB Scenes		Objaverse Scenes	
	IoU ↑	Chamfer ↓	IoU ↑	Chamfer ↓	IoU ↑	Chamfer ↓
w/ BHM Sampling	0.156	0.031	0.313	0.030	0.326	0.035
V-PRISM (ours)	**0.309**	**0.011**	**0.500**	**0.012**	**0.464**	**0.018**
w/o Under the Table	0.291	0.019	**0.500**	0.014	0.439	0.024
w/o Stratified Sampling	0.145	0.024	0.294	0.023	0.291	0.029

Table 2: Ablation experiments on our negative sampling method.

scenes. Once the poses have been selected, we simulate physics for a fixed period of time to ensure objects can come to rest.

Our first experiment on simulated scenes compares our method with the two baselines. Similar to [16], we use a level set other than $\tau = 0.5$ for constructing the mesh with the neural network. We found $\tau = 0.3$ to provide the best reconstructions for our version of PointSDF. For other methods, we use $\tau = 0.5$. We report the IoU and Chamfer distance in Table 1. PointSDF outperforms other methods on the ShapeNet scenes, where the meshes are drawn from the same mesh dataset that it was trained on. On other datasets, our method outperforms PointSDF. This aligns with other work demonstrating that neural networks perform worse the further from the training distribution you get. Because our method has no reliance on a training distribution, it shows consistency across all datasets. Both our method and PointSDF consistently outperform the voxel baseline on most datasets and metrics. The only exception is Chamfer distance on Objaverse scenes, where the voxel baseline outperforms PointSDF. The performance of our method relative to our baselines indicate that our method results in accurate reconstructions

Our second experiment on simulated scenes ablates our negative sampling method. We observe the effect of removing sampling under the table plane and removing the stratified sampling along the ray. In order to remove the stratified sampling, we replace it with taking discrete, fixed steps along each ray instead. We also compare against the original BHM sampling method explained in [2], where there negative samples are drawn randomly along the whole ray instead of near objects. This is labeled as **BHM Sampling**. The IoU and

Hyperparameters (Learning)	Value	Hyperparameters (Sampling)	Value (cm)
kernel type	Gaussian	grid length	5.0
kernel γ	1000	sampling r_{obj}	25.0
surface hinge pts.	32	subsample res. (objects)	1.0
iterations	3	subsample res. (empty)	1.5

Table 3: Hyperparameters for experiments on procedurally generated scenes.

Chamfer distance are reported in Table 2. Our negative sampling method outperforms the others on each dataset and metric. This implies that our proposed sampling method does improve reconstruction quality when compared to alternatives.

The hyperparameters used for the simulated experiments are shown in Table 3. These were kept constant across all procedurally generated datasets and corresponding experiments.

Real World Scenes: We evaluate our method by qualitatively comparing reconstructions on real world scenes. We use a Intel RealSense D415C camera to obtain point clouds of tabletop scenes. In order to get accurate segmentations of the scene, we use the Segment Anything Model (SAM) [17]. We compute reconstructions on five scenes consisting of multiple objects. Each map of these scenes took between 2 and 5 seconds to compute. We compare our method to PointSDF. The qualitative comparison can be seen in fig. 6. Because these scenes are significantly more noisy than simulated scenes, PointSDF struggles to coherently reconstruct the scene. In contrast, our method is capable of producing quality reconstructions even with very noisy input point clouds. This suggests that our method is capable of bridging the sim to real gap and is robust to unknown, noisy scenes.

Uncertainty: To show how our model captures uncertainty about the scene, we need a way to quantify uncertainty. We use the entropy of our map at each point in space as a measurement of uncertainty:

$$H_m(\mathbf{x}) = -\sum_{k=1}^{c} \hat{P}(y = c|\mathbf{x}) \ln \hat{P}(y = c|\mathbf{x}).$$

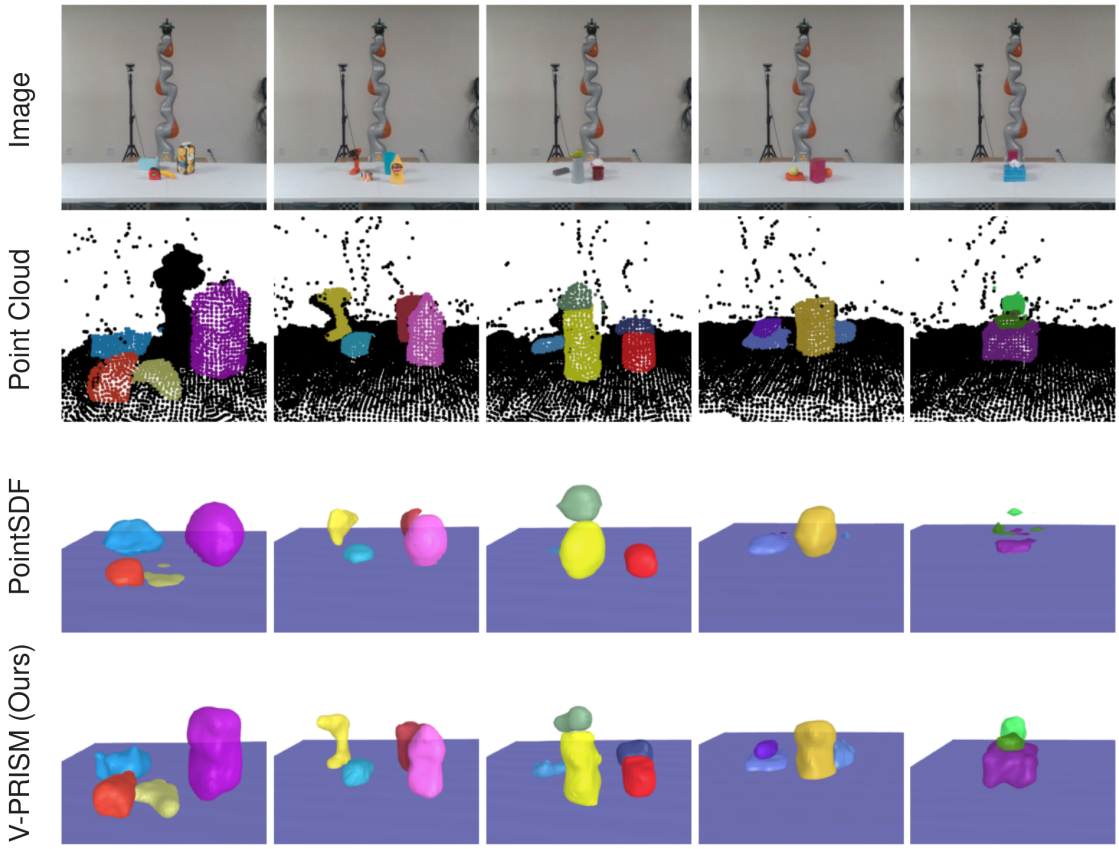

Figure 6: Qualitative comparisons with PointSDF reconstructions. **First row:** RGB images. **Second row:** the segmented point cloud used as input. **Third row:** PointSDF reconstructions. **Last row:** V-PRISM's (our method) reconstructions. V-PRISM results in quality reconstructions on noisy scenes.

This is maximized when the model predicts a uniform distribution over classes and minimized when the model predicts a single class with a probability of 1.

We compare our method with an alternate non-Bayesian version of our method, where we train a single weight vector with stochastic gradient descent (SGD) instead of the EM algorithm, to minimize the negative log-likelihood of our augmented data.

To visualize this uncertainty, we calculate this uncertainty over a 2D slice from each of our 5 real world scenes. The heat maps for each slice can be seen in fig. 7. Qualitatively, we can see that our method obtains high uncertainty values in occluded sections of the scene. This contrasts to the non-probabilistic model that does not accurately capture uncertainty

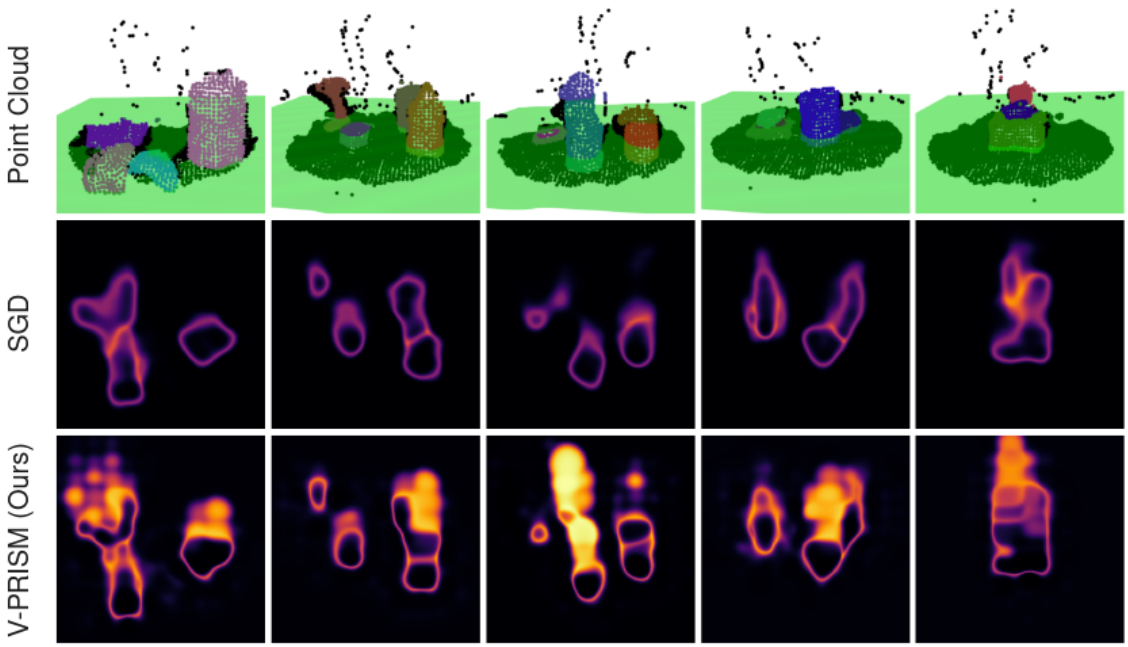

Figure 7: Qualitative comparison of uncertainty. **Top row:** the observed point cloud with a green plane corresponding to the 2D slice where the heat maps were calculated. We compare a non-probabilistic variant of V-PRISM trained with gradient descent (**middle row**) and our method (**bottom row**). In the heat maps, the bottom is closer to the camera and the top is farther from the camera. Lighter areas correspond to more uncertainty. Our method predicts high uncertainty in occluded areas of the scene.

about occluded regions. The heat maps showing occlusion-aware uncertainty suggest our model captures principled and accurate uncertainty measures.

REFERENCES

[1] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[2] R. Senanayake and F. Ramos, "Bayesian hilbert maps for dynamic continuous occupancy mapping," in *Conference on robot learning*, 2017, pp. 458–471.

[3] T. S. Jaakkola and M. I. Jordan, "A variational approach to bayesian logistic regression models and their extensions," in *Sixth international workshop on artificial intelligence and statistics*, 1997, pp. 283–294.

[4] R. Senanayake, A. Tompkins, and F. Ramos, "Automorphing kernels for nonstationarity in mapping unstructured environments." in *CoRL*, 2018, pp. 443–455.

[5] W. Zhi, L. Ott, R. Senanayake, and F. Ramos, "Continuous occupancy map fusion with fast bayesian hilbert maps," in *2019 international conference on robotics and automation (ICRA)*, 2019, pp. 4111–4117.

[6] R. Senanayake and F. Ramos, "Building continuous occupancy maps with moving robots," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.

[7] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[8] G. Bouchard, “Efficient bounds for the softmax function and applications to approximate inference in hybrid models,” in *NIPS 2007 workshop for approximate bayesian inference in continuous/hybrid systems*, 2007, vol. 6.

[9] J. Daunizeau, “Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables,” *arXiv preprint arXiv:1703.00091*, 2017.

[10] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in *Proceedings of the 14th annual conference on computer graphics and interactive techniques*, 1987, pp. 163–169.

[11] H. Thomas, “Learning new representations for 3D point cloud semantic segmentation,” PhD thesis, Université Paris sciences et lettres, 2019.

[12] M. Van der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans, “Learning continuous 3d reconstructions for geometrically aware grasping,” in *2020 IEEE international conference on robotics and automation (ICRA)*, 2020, pp. 11516–11522.

[13] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “ShapeNet: An Information-Rich 3D Model Repository,” Stanford University — Princeton University — Toyota Technological Institute at Chicago, arXiv:1512.03012 [cs.GR], 2015.

[14] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *2015 international conference on advanced robotics (ICAR)*, 2015, pp. 510–517.

[15] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 13142–13153.

[16] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4460–4470.

[17] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

Name of Candidate: Herbert Wright

Date of Submission: December 6, 2024