Compact Generative Models of Point Cloud Data for 3D Perception

Benjamin Eckart

CMU-RI-TR-17-68

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

October 2017

Thesis Committee:
Alonzo Kelly, Chair
Martial Hebert
Srinivasa Narasimhan
Jan Kautz, NVIDIA

Dedicated to my parents.

Abstract

One of the most fundamental tasks for any robotics application is the ability to adequately assimilate and respond to incoming sensor data. In the case of 3D range sensing, modern-day sensors generate massive quantities of point cloud data that strain available computational resources. Dealing with large quantities of unevenly sampled 3D point data is a great challenge for many fields, including autonomous driving, 3D manipulation, augmented reality, and medical imaging. This thesis explores how carefully designed statistical models for point cloud data can facilitate, accelerate, and unify many common tasks in the area of range-based 3D perception. We first establish a novel family of compact generative models for 3D point cloud data, offering them as an efficient and robust statistical alternative to traditional point-based or voxel-based data structures. We then show how these statistical models can be utilized toward the creation of a unified data processing architecture for tasks such as segmentation, registration, visualization, and mapping.

In complex robotics systems, it is common for various concurrent perceptual processes to have separate low-level data processing pipelines. Besides introducing redundancy, these processes may perform their own data processing in conflicting or *ad hoc* ways. To avoid this, tractable data structures and models need to be established that share common perceptual processing elements. Additionally, given that many robotics applications involving point cloud processing are size, weight, and power-constrained, these models and their associated algorithms should be deployable in low-power embedded systems while retaining acceptable performance. Given a properly flexible and robust point processor, therefore, many low-level tasks could be unified under a common architectural paradigm and greatly simplify the overall perceptual system.

In this thesis, a family of compact generative models is introduced for point cloud data based on hierarchical Gaussian Mixture Models. Using recursive, data-parallel variants of the Expectation Maximization algorithm, we construct high fidelity statistical and hierarchical point cloud models that compactly represent the data as a 3D generative probability distribution. In contrast to raw points or voxel-based decompositions, our proposed statistical model provides a better theoretical footing for robustly dealing with noise, constructing maximum likelihood methods, reasoning probabilistically about free space, utilizing spatial sampling techniques, and performing gradient-based optimizations. Further, the construction of the model as a spatial hierarchy allows for Octree-like logarithmic time access. One challenge compared to previous methods, however, is that our model-based approach incurs a potentially high creation cost. To mitigate this problem, we leverage data parallelism in order to design models well-suited for GPU acceleration, allowing them to run at real-time rates for many time-critical applications. We show how our models can facilitate various 3D perception tasks, demonstrating state-of-the-art performance in geometric segmentation, registration, dynamic occupancy map creation, and 3D visualization.

Contents

1 Introduction **1**
- 1.1 Approach . . . 2
- 1.2 Thesis Contributions . . . 4
- 1.3 Thesis Outline . . . 4

2 Background – Range Data Representations **7**
- 2.1 Voxel-Based Representations . . . 8
  - 2.1.1 Dense Voxel Grids . . . 8
  - 2.1.2 Recursive Subdivisioning and Sparse Techniques . . . 9
- 2.2 Parametric Point Cloud Models . . . 10
  - 2.2.1 Planar Models . . . 11
  - 2.2.2 Generative Models . . . 11
  - 2.2.3 Normal Distance Transforms . . . 12
- 2.3 A Comparison of our Proposed Method . . . 13

3 Data-Parallel Gaussian Mixtures **15**
- 3.1 Model Construction . . . 15
- 3.2 Gaussian Mixtures . . . 17
- 3.3 EM Parallelization for GMMs . . . 17

4 Registration **21**
- 4.1 Previous Approaches . . . 22
  - 4.1.1 Iterative Closest Point . . . 22
  - 4.1.2 Statistical Algorithms . . . 23
- 4.2 Gaussian Mixture-Based Approaches . . . 25
  - 4.2.1 Traditional Approaches . . . 25
  - 4.2.2 Our Approach: Mixture Decoupling . . . 27
- 4.3 REM-Seg and Gradient Descent . . . 28
  - 4.3.1 Extensions . . . 30
  - 4.3.2 Implementation and Algorithm . . . 31
  - 4.3.3 Experimental Design and Metrics Used . . . 32
  - 4.3.4 Experiments . . . 33
- 4.4 MLMD and Expectation Maximization . . . 36
  - 4.4.1 Approach . . . 37

4.4.2 Overview . . . 37
4.4.3 E and M_{Θ} Steps . . . 38
4.4.4 M_T Step . . . 39
4.4.5 Closed Form Approximations . . . 41
4.4.6 Results . . . 42
4.4.7 Scalability . . . 44
4.5 Mixture Decoupled Mahalanobis Estimation . . . 46
4.5.1 Mahalanobis Estimation Least Squares Derivation . . . 49
4.6 Summary . . . 49

5 Hierarchical Mixture Construction **51**
5.1 Expectation Sparsification . . . 53
5.2 Introduction . . . 53
5.3 Related Work . . . 55
5.4 Method Overview . . . 56
5.4.1 Model Definition . . . 57
5.4.2 Expectation Sparsity . . . 58
5.4.3 A Top-Down Hierarchy of Mixtures . . . 58
5.4.4 Sparsification: Hard and Soft Partitioning . . . 59
5.4.5 Parallel Construction . . . 60
5.5 Implementation Details . . . 60
5.5.1 Soft Partitioning . . . 63
5.6 Model Evaluation . . . 64
5.6.1 Reconstruction Fidelity . . . 64
5.6.2 Computational Speed and Scalability . . . 67
5.7 Summary . . . 67

6 Inference and Visualization **73**
6.1 Sampling and Integration . . . 73
6.1.1 Point Cloud Reconstruction . . . 73
6.1.2 Occupancy Grid Generation . . . 73
6.2 Isosurface Extraction . . . 75
6.2.1 Marching Cubes . . . 75
6.2.2 Sparse Augmentation of Marching Cubes . . . 78
6.2.3 Results on Datasets . . . 79
6.2.4 Adaptive Coarse-to-Fine Scene Sampling . . . 79
6.2.5 Real-Time Operation on Mobile Device . . . 80
6.3 Summary . . . 82

7 Context-Aware Registration, Fusion, and Mapping **85**
7.1 Introduction . . . 85
7.2 Related Work . . . 87
7.3 Approach and Methodology . . . 89
7.3.1 Ω-Estimation . . . 89

7.3.2 Expected Overlap Estimation 90
7.4 Evaluation 92
7.4.1 Robustness to Partially Overlapping Views 92
7.4.2 Kinect Data 93
7.4.3 Velodyne LIDAR Data 94
7.5 GMM Fusion and Mapping: A Unified Pipeline 96
7.6 Summary 99

8 Conclusion **105**

Bibliography **107**

List of Figures

4.1 **Mixture Decoupling** *Left:* Given two similarly sized point clouds of size N, point-based GMMs produce N^2 potential matches. *Right:* Decoupling points from mixtures into a smaller, more descriptive set of latent $\mathbf{\Theta}$ produces two separate procedures of size $O(JN)$. If $J \ll N$, this effectively linearizes the registration process with respect to N. 28

4.2 Pseudocode for REM-Seg. 32

4.3 Example surface patches with each point colored by its most likely cluster. Note that since the GMM formulation is additive with respect to each cluster, over-segmenting serves to more accurately construct a PDF of scene. 33

4.4 Goodness of fit using the Bunny dataset. These convergence results are from 100 random rigid transformations. REM-Seg consistently converges to near-zero error while the other algorithms tend to get stuck at local minima. The bottom right subfigure shows the two best algorithms with a zoomed x-axis. 35

4.5 Example Registration Results 35

4.6 Timing results as the number of points in the cloud to be matched to the scene increases for the Kinect dataset. The top plot shows the time to convergence on a log scale, while the bottom plot shows the same data on a linear y-axis among ICP, EM-ICP-GPU, and REM-Seg. 36

4.7 **Algorithm Flow** The algorithm first iterates over E and $M_{\mathbf{\Theta}}$ until convergence, a condition denoted by the binary variable $C_{\mathbf{\Theta}}$. Once $M_{\mathbf{\Theta}}$ converges under $\mathcal{Z}_1$, the algorithm then switches to the second point cloud for input (shown through the one-bit multiplexer). In order to produce an optimization criterion of size J, the $M_{\mathbf{\Theta}}$ Step result then feeds into the M_T Step along with the last $\hat{\mathbf{\Theta}}$ from $\mathcal{Z}_1$ as $\mathbf{\Theta}_{final}$. The algorithm then iterates accordingly, finding new transformation updates $\hat{T}$, which are applied to $\mathbf{\Theta}_{final}$. 38

4.8 **Datasets** *Left:* Bunny [143], *Right:* Lounge [156] 42

4.9 **Registration quality for varying Frobenius norm errors** Three sample results from the random transformation test detailed in Section 4.4.6 and Table 4.2. The results depicted are the final converged results between both point clouds (red and blue points). (a) shows an error level that falls within both chosen bounds as given by Table 4.2. (b) shows a level of registration error that is within the 0.025 error threshold but not 0.01. (c) shows a high level of Frobenius norm error that is outside of both the chosen bounds. 43

4.10 **Robustness to Large Transformations** Pitch axis transformations from -180 to 180 degrees were performed over randomly sampled Stanford Bunnies (N=2000). ICP has the smallest convergence radius, handling transformations of -45 to 45 degrees. EM-ICP and GMMReg approximately double the convergence region. Due to sampling nonuniformity, SoftAssign often produced poor registration accuracy though its worst-case solution was the best of all algorithms besides our proposed algorithm. 44

4.11 **Robustness to Noise** We injected both outliers and point-level Gaussian noise into two subsampled Bunny point clouds (N=2000) separated by 15 degrees on all roll, pitch, and yaw axes. Each iteration included both more outliers (shown as a percentage on the x-axis) and more point noise (shown as a Signal-to-Noise dB value). We did not include ICP or SoftAssign since any amount of noise and outliers tested corrupted the result significantly. 45

4.12 **Computation Times** Scaling the number of points subsampled from each point cloud for registration. Note that the scale is logarithmic on both x and y axes. . . 45

5.1 **Probabilistic Modeling using a Tree of Gaussian Mixtures:** Four Stanford bunny models reconstructed from different levels of detail in our GMM hierarchy. Each color denotes the area of support of a single Gaussian, and the ellipsoids indicate their 1 sigma extent. For better visualization, the ellipsoids are not drawn on the two rightmost figures to avoid cluttering due to the high level of detail. 52

5.2 **Processing PCD with a Hierarchy of Gaussian Mixtures:** (a) Raw PCD from Stanford Bunny (*35k vertices*), (b) and (c) Two levels of detail extracted from the proposed model. Each color denotes the area of support of a single Gaussian and the ellipsoids indicate their one σ extent. Finer grained color patches therefore indicate higher statistical fidelity but larger model size, (d) a log-scale heat-map of a PDF from a high fidelity model. (e) stochastically re-sampled PCD from the model (*5k points*), (f) occupancy grid map also derived directly from the model. See Chapter 6 for technical details regarding (e) and (f). 54

5.3 **Graphical model of hierarchical GMM** An example of a three-level hierarchy, where a series of causally linked latent variables are used to identify salient geometric regions of influence for each observed point $\mathbf{z}_i$. 60

5.4 **Data set used for our evaluation:** (Top left): Bunny - 35,947 points, (Top right): Burghers - 3.47M points, (Bottom left): Cactus - 1.9M points, (Bottom right): Lounge - 1.62M points 63

5.5 **Levels of Fidelity:** The colors in each heatmap shows the accumulated PDF values projected onto the screen space. PSNR, model size, and *the reduced storage size* are shown for each level. We vary the level of detail (L2 to L4) to show the trade-off between storage size and fidelity. The original PCD is 421 kB. . . . 64

5.6 A comparison of data structure size vs fidelity over several standard point cloud datasets. The blue dashed line indicates the original point cloud size. Note the x-axis is on log scale. The star markers indicate different levels in the GMM hierarchy. At similar size models, the hierarchical GMM has much better PSNR (reconstruction performance) with respect to the original data when compared against the 3D-NDT, 3D-NDT-Octree, and a simple subsampling strategy for point cloud reduction. 65
5.7 **Burghers of Calais:** Comparison of reconstruction fidelity for same-sized models. 69
5.8 **Lounge:** Comparison of reconstruction fidelity for same-sized models. 70
5.9 **Cactus Garden:** Comparison of reconstruction fidelity for same-sized models. . 71
5.10 **Stanford Bunny:** Comparison of reconstruction fidelity for same-sized models. . 72

6.1 **Occupancy estimates:** Top: Raw PCD. Bottom: an example high resolution occupancy map obtained by sampling a hierarchical GMM (max depth 5) produced from the points. 76
6.2 **Overview of our approach.** We first build a hierarchical GMM by recursively applying EM (Chapter 5). Secondly, we run stochastic importance sampling on the highest detail PDF to build a list of active voxels (Sec. 6.2) that we feed to marching cubes for the final isosurface reconstruction (Sec. 6.2.2). 77
6.3 **Sparsity of point cloud data** Sparsity is a common scenario for many modern range sensors. For a sample point cloud data captured from a depth sensor, the number of occupied voxels in a sample grows close to $O(V^2)$, where the total number of voxels grows as $O(V^3)$. 77
6.4 **Qualitative comparison between our reconstruction and original meshes:** Images in 1st row show isosurfaces generated from our L5 models (cropped to show detail) using stochastic marching cubes. The images in the 2nd row show the original mesh [156]. 78
6.5 **Isosurface reconstruction under noise:** (1st column): Stanford bunny with randomly added noise (Normal distribution), (2nd column): *our results*, (3rd column): Poisson reconstruction [65], (4th column): Ball-pivoting [2]. Note that even with more aggressive noise (2nd row), our approach can provide better overall model quality. 79
6.6 **Adaptive scene sampling:** Left : entire scene reconstruction, notice the coarseness of the triangulation (originally 3.47M points), Right: a zoomed-in region (denoted as red rectangles in Right) is finely triangulated. In both cases the sampling volume is fixed at 256x256x256. 80
6.7 **Modeling on the Android platform** We run our approach on a mobile platform at interactive rates. 81
6.8 **PDFs and reconstructed structures:** (1st column): Heat maps represent the areas of high data probability (PDFs) for each model, (2nd column): The reconstructed scenes with our hierarchical GMM and stochastic marching cubes in low voxel resolution (256^3). The red rectangle denotes the region of selection where we reconstruct with higher resolution. (3rd column): Our reconstruction with higher quality. Note that the original meshes are from [156] 83

7.1 **Point Cloud Registration with Limited Overlap** Leftmost: Two range sensors at different poses capturing different slices of geometry. Note that the geometry is illustrated in 2D contours for simpler visualization: Blue and red contours for the geometry captured from each view, and magenta for overlapped region. During the registration process, the non-overlapping pieces of geometry are problematic as they may be mistakenly matched. For generative model-based methods, this situation manifests itself as a violation of the paradigm's most fundamental assumption that both point sets are *iid* samples of a single model. Rightmost: If overlap can be directly estimated as part of the generative model, then non-overlapping regions could be pruned, leading to a more well-formed registration problem both practically and theoretically. 86

7.2 **Expected Overlap Estimation.** A range sensor from two different viewpoints (A and B) has two different fields-of-view (Red and Blue). Building the sensor view into the registration process allows robust estimation of overlapping regions. 90

7.3 **Partially Overlapping Bunnies** Five partially overlapping depthmaps of the Stanford Bunny were registered together in sequence and then placed into a global coordinate system. Each color refers to points sets from different views. The top row consists of algorithms run normally, and the bottom row shows the performance of those same algorithms augmented with EOE. Though MDME+EOE (bottom-left) has the lowest error, the addition of EOE generally improves all algorithms. 93

7.4 Input depth stream used for bunny overlap experiments. 93

7.5 **Lounge Error Distributions** Histograms of frame-to-frame error as measured by average Euler angular deviation from ground truth, for the Lounge dataset. The top row show the results of algorithms without EOE and the bottom row includes EOE. EOE improves the overall accuracy of all algorithms tested. 94

7.6 Lounge data for additional experiments. **Top**: Sample depth map stream. **Bottom**: Ground truth pose used for accuracy calculations. 95

7.7 **Velodyne View Model**: Though the Velodyne VLP-16 LIDAR has a 360 degree horizontal field-of-view, it suffers from a relatively narrow vertical field-of-view of 30 degrees. Similarly, the angular resolution of the pitch and yaw axes vary dramatically. These properties lead to the formation of dense rings of sampled geometry around the sensor, which often will degrade the accuracy of point-to-point matching algorithms. Expected Overlap Estimation (EOE) utilizes the knowledge of this narrow vertical field-of-view by automatically culling points belonging to dense rings that can no longer be seen (shown in green). 96

7.8 **Sample Problematic Registration**: Here we show one problematic example of 2 frames of data (red and blue) being registered together. This is the final registered output of both Trimmed ICP and our proposed method, MDME+EOE. The dense concentric rings of points serve to degrade the robust ICP-based Trimmed ICP algorithm (top). Given that these rings of points are returns from flat ground, their appearance will not change very dramatically with any pose change on the x, y, or yaw axes. Thus, a point-to-point registration algorithm is likely to produce a no-movement result as it will try to maximally overlap the rings of points. The result on the bottom shows a correctly registered result: note that the rings are not overlapped. 97
7.9 **Top Left**: Velodyne view model of 360 degrees horizontal and 30 degrees vertical field-of-view. **Top Right**: A single registration result showing how TrICP tries to match the circular sampling patterns, while MDME+EOE correctly matches real overlapping geometry. **Bottom**: The LIDAR data in global frame. The smeared wall pattern on the right shows TrICP's inability to register this type of LIDAR data. 98
7.10 **FICP, ICP, MLMD**: This is a top-down view of twelve frames of LIDAR data that were registered frame-to-frame and then plotted in a global coordinate system. The ground points have been removed for clarity. Look to the northern corridor for the clearest signs of registration error. ICP+EOE and FICP+EOE produce the best results. 100
7.11 **IRLS-ICP, Trimmed ICP, MDME (our proposed method)**: This is a top-down view of twelve frames of LIDAR data that were registered frame-to-frame and then plotted in a global coordinate system. The ground points have been removed for clarity. Look to the northern corridor for the clearest signs of registration error. Note the severe inadequacy of Trimmed ICP, which produces too little movement. Our proposed method without EOE does fairly well, but performs best with the addition of EOE. 101
7.12 Frame-to-frame mapping over many frames in an outdoor urban environment. Units are in meters. The calculated path is the multicolor trajectory. The red dotted path is ground truth. We compared the best performing algorithm in the *libpointmatcher* library without EOE (a point-to-plane ICP with robust outlier rejection using a trimmed distance outlier filter) against MDME+EOE (the best performing algorithm with EOE). Note the increased drift of point-to-plane ICP when compared to our method, especially when turning around corners. 102
7.13 **Left** Pose graph without yet performing optimization. Edges found during our matching process are represented by red lines, and poses by blue dots. We have circled two areas of accumulated drift with red ellipses. **Right** Graph optimization has provided pose corrections to remove the small amount of drift that accumulated from driving around a city block. 103

List of Tables

4.1 **A Comparison of Probabilistic Registration Methods.** *Multiply Linked*: many-to-one or many-to-many correspondences (robustness under non-uniform sampling) *Probabilistic Correspondences*: fully probabilistic correspondences or not (as opposed to kernel-based or nearest neighbor approaches). Allows for application of EM, GEM, or ECM. *Covariance Estimation*: Improves convergence as opposed to statically set covariances or simulated annealing methods. *Anisotropic*: non-spherically shaped mixture covariance. Most earlier methods restrict covariances to be uniformly isotropic across all mixtures. This improves robustness through better local shape alignment. *Compactness*: Given PCD of size N, compact models are those that have a number of mixtures $J \ll N$. This addresses scalability problems with point-based GMM representations. 26

4.2 **Accuracy of random rigid transformations** 100 random 6DOF transformations were applied to a downsampled Stanford bunny corrupted with 5% outliers. The random rotations were bounded by 90 degrees in absolute sum over each axis angle, and the random translations were bounded by the extent of the dataset on each axis. Given two different error levels, the percentage of "successful" registration results is shown, where the error is defined by the Frobenius norm distance of the rotation matrix. To see a qualitative representation of Frobenius norm error, refer to Figure 4.9. 43

4.3 Frame-to-frame rotation error over a stream of Kinect frames from the Stanford Scene dataset (Lounge) 46

5.1 **A Comparison of 3D Point Cloud Data Structures** *Hierarchical*: Hierarchical methods compress free space and are therefore more compact than dense grids. *Generative*: Generative models add parametric structure to PCD, facilitating statistical inference, maximum likelihood, or continuous optimization methods. *Voxel Free*: The lack of voxelization present in the model avoids discretization errors, allowing higher fidelity at smaller model sizes. *Construction complexity*: N is the number of points in the PCD, and J the number of mixtures in a GMM, with $J \ll N$ for most typical applications. 55

5.2 **Hierarchy construction time.** L1 to L4 refers to a level of the hierarchy (i.e., L3 denotes the the process including L1 to L3). D refers to a desktop computer (i5-3500/GTX660) used for the computation, and M denotes a Mobile device (NVIDIA Shield tablet). Raw depth refers the point cloud directly captured from Softkinetic DS325 depth camera. 67

5.3 **Construction speed-up relative to a flat GMM model.** The table compares the E Step execution time of the hierarchical GMM compared with a flat GMM having the same number of mixtures on the full Burghers model (∼4.5M pts). At higher detail levels, the proposed hierarchical GMM is significantly faster to build than the flat GMM. 67

7.1 Comparison to other methods. 94

List of Symbols

Symbol	Description
$\mathcal{Z}$	A collection of 3D points (3D point cloud)
$\mathbf{z}_i$	A 3D point $\{x, y, z\}$ with index i in a point cloud $\mathcal{Z}$
N	The number of points in a point cloud $\mathcal{Z}$
$\boldsymbol{\Theta}$	The collection of parameters that define a Gaussian Mixture
J	The number of components in a Gaussian Mixture comprising parameters $\boldsymbol{\Theta}$
μ	Gaussian distribution mean
$\boldsymbol{\Sigma}$	Gaussian distribution covariance
π	Weighting vector of Gaussian Mixture
$\boldsymbol{\Theta}_j$	The parameters $\{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$ that define the jth weighted Gaussian distribution of a Gaussian Mixture
$\mathcal{C}$	A NxJ set of binary correspondence variables, indicating for each point $\mathbf{z}_i$ the Gaussian distribution $\boldsymbol{\Theta}_j$ that generated it
c_{ij}	A particular binary correspondence variable indicating the relationship between the ith point $\mathbf{z}_i$ and the jth mixture comprising $\boldsymbol{\Theta}_j$
γ_{ij}	The expected value of c_{ij} under a given Gaussian mixture with fixed parameters
$\mathcal{N}(\cdot\|\cdot)$	Conditional Gaussian distribution
$\sum_{ij}$	A double sum over all N points and J mixture components
$\tilde{\mathcal{Z}}$, $\tilde{\mathbf{z}}_i$, $\tilde{\boldsymbol{\mu}}_j$	Denotes variables that have undergone rigid spatial transformations
$\hat{\boldsymbol{\Sigma}}_j$, $\hat{\boldsymbol{\mu}}_j$	Denotes the Maximum Likelihood Estimate (MLE) of parameter values inside an Expectation Maximization (EM) framework
R	3x3 rotation matrix
$\mathbf{t}$	3x1 translation vector
$\mathbf{q}$	associated quaternion for R

Chapter 1

Introduction

In any given large-scale robotics application, there might be dozens of perceptual tasks running concurrently, often using the same underlying sensor data, but each maintaining completely separate data processing pipelines. This situation is not ideal.

In terms of data processing, most common range-based 3D perception algorithms (e.g. segmentation, object detection, registration, path planning) do not operate directly on the raw point data that comes from the sensor. There are several reasons for this: often, the data is too dense, contains too many outliers or too much noise, or the points simply can't encode the required primitive elements needed for the perceptual task, like geometric or connectivity information.

To facilitate whatever perceptual operation needs to be performed, most algorithms that operate over point clouds therefore typically subsample heavily, discretize, or create some intermediate representation. For example, if the operation is plane detection, PCA is often used to find good planar fits around neighborhoods of points [118]; if it is object detection, pose-invariant descriptors are calculated around interest points [61]; and for registration, points may be decimated and locally approximated by Gaussian clusters [43]. In a complex robotics system, all of these processes may be performed concurrently, leading to redundant and/or potentially inferior techniques to filter, describe, and operate over the raw point data. Furthermore, when each perceptual process has to produce its own application-dependent data processing pipeline, this results in difficulties with information or context sharing between processes as well as an overall loss in programming productivity in terms of systems integration.

It therefore seems useful and necessary to explore the possibility of a common data structure that is both flexible and general enough to enable and unify as many common 3D perception tasks as possible. Such a low-level point processing structure could handle the problems of noise reduction, outlier handling, point sparsity, and solve basic questions about point geometry and free space. These operations could be formed in principled way, such that each individual perception algorithm doesn't have to independently implement a separate version of noise and outlier handling, geometric inference, model parametrization, etc. With a common model taking care of these basic point processing tasks, basic context between perception algorithms could be established under a unified architecture, offering a more cohesive, efficient, and overall better engineered paradigm for 3D perception based on range data.

1.1 Approach

Approaching the problem of 3D perception as an exploration of the representation of sensor data is a sensible strategy since many perceptual tasks share many common computational primitives. The problem is then to establish these perceptual primitives and devise a common model that can both leverage their similarities while accelerating their computation. Towards this end, we propose that a sufficient spatial model should have the following qualities:

1. **Statistical**: Statistics offer a natural ability to handle noise and represent uncertainties in data acquisition. In contrast, deterministic representations tend to be more expensive in terms of representational complexity or size.
2. **Generative**: In the context of 3D perception, generative models capture the full joint probability of 3D space, leading the way to efficient sampling methods and Bayesian inference techniques allowing one to incorporate prior knowledge into the model.
3. **Hierarchical/Multi-Scale**: The model's ability to describe the world needs to be dynamic and data-driven in order to handle scenes of arbitrary complexity and provide finer or coarser grained access to information depending on the demands of the application. Imposing a hierarchy can exponentially accelerate spatial reasoning systems by compressing empty space, provide view-dependent levels of visual detail to an operator, and enable anytime optimality.
4. **Data-Parallel**: Serial processing of sensor data is a common bottleneck. Sensors generate large amounts of data that must be operated over in parallel in order to obtain good scaling and real-time performance. Data structures for 3D perception then must be designed to have limited spatial interdependence in order to scale well on highly parallel architectures such as FPGAs and GPUs.
5. **Parametric/Differentiable**: Parametric models facilitate fast spatial transformations since only the parameters that describe the model need to be operated over, as opposed to spatial transformations over large arrays of voxels or 3D points. Parametric forms that are differentiable also facilitate well-defined and computationally tractable optimization techniques and allow the use of maximum likelihood methods when paired with statistical models.
6. **Compact**: Highly structured parametric models allow more compact representations and lead to smaller memory footprints, a key component when transferring data to other devices or through a network or when performing calculations on low-power or embedded hardware where memory is limited.

Much work in perception, learning, and control has been primarily focused on new algorithmic, theoretical, or mathematical developments, relegating the data processing element in these methods to a secondary role. Many commonly used 3D perception algorithms tend to deal with point cloud data in *ad hoc* ways (e.g. by subsampling, binning, decimating, or voxelization), use heuristic-based optimization approaches (e.g. ICP for 3D registration [3]), and do not scale well to massive amounts of data (e.g. most previous generative model-based approaches are far from real-time [43] [40]). Hierarchical spatial decompositions such as Octrees [153] or K-d trees [105] work well to facilitate efficient range searches, but can still require expensive preprocessing, are hard to parallelize, and, like raw point clouds, do not facilitate easy parametrization, reasoning

about uncertainty, or numerical optimization. Statistical constructions have been more successful in these latter areas, such as the Normal Distance Transform (NDT) [134], but these models too are affected by voxelization artifacts, scaling issues, and tradeoffs in representational fidelity (see Chapter 5.6.1 for an in-depth study of these issues).

It is for these reasons that we've chosen to explore the *Gaussian Mixture Model* as the fundamental statistical primitive to describe and parameterize raw point cloud data. The Gaussian Mixture is simply a convex sum of Gaussian distributions, but being generative and extensible to a full Bayesian formulation, it offers a natural way to denoise and prune outliers from potentially noisy clouds, and encoded in the Gaussian structure is inherent connectivity/continuity and geometric information. The Gaussian Mixture trivially includes and therefore generalizes the Gaussian distribution (a single component mixture), but also can describe any arbitrarily complex function or probability distribution as the number of mixtures is increased. Furthermore, if the Gaussian Mixture is used in conjunction with a partitioned or hierarchical scheme, many of the best benefits of various popular recursive subdivisioning techniques (e.g. Octrees, Kd-Trees) can still be obtained (Chapter 5).

The main trade-off between a mixture-based structure and something like an Octree/Kd-Tree is that construction isn't as trivial as repeated numerical comparison of 3D coordinates, but instead typically involves solving a maximum likelihood optimization problem. However, as we show in Chapter 3, this construction cost can be mitigated by a GPU or FPGA-based implementation that properly exploits the inherent data parallelism of point clouds.

Other generative Gaussian Mixture-based approaches have been used to accelerate various point-based perception tasks [43, 59]. The proposed approach in this thesis generalizes them into a single unified family of models, and shows how this generalization leads to various novel strategies for organizing mixture-based structures into compact yet statistically descriptive forms. Compactness, as we will show, is an especially important property for generative point models as it provides computational tractability and added robustness, and it is this property that most differentiates our proposed structures over other models discussed in the literature.

This thesis includes an in-depth discussion about the creation of this family of compact, hierarchical generative models for point cloud data, including their theoretical noise handling properties, model identifiability, convergence properties, fidelity and model size trade-offs, computational properties, limitations, and fully Bayesian formulation and use of priors. We discuss how these models relate to other generative models in the literature, including basic models used to facilitate registration and segmentation and other popular models such as the NDT.

Having established these basic properties, we then show how many common perceptual tasks can be unified under this single model, positioning it as an essential statistical structure used to provide common perceptual primitives to each. We demonstrate in this thesis how the following fundamental applications may be implemented efficiently under this common model: geometric segmentation (Chapters 3 and 5), registration (Chapter 4), inference and visualization (Chapter 6), and model fusion and mapping (Chapter 7).

1.2 Thesis Contributions

This thesis offers a broad exploration of a family of compact statistical models for 3D point cloud data that facilitate and unify several common low level 3D perceptual tasks. Given the model's ability to bundle common computational primitives into one coherent, compact statistical world model, we foresee this work having widespread applicability in domains that utilize 3D range data but have limited computational resources, which includes domains like autonomous driving, augmented reality, computational photography, multi-robot mapping, or teleoperative interfaces. Specifically, this thesis offers the following research contributions:

1. **Hierarchical Generative Modeling** A set of data-parallel algorithms for creating compact, hierarchical, parametric, and generative point cloud models that can be used to facilitate, accelerate, and unify various low to mid-level perceptual tasks in novel ways.
2. **3D Registration** A set of novel 6-DoF rigid registration algorithms based on these established models, showing robustness to large transformations and noise, and that beat current state-of-the-art.
3. **Context-Aware Registration** A novel augmentation to the traditional model-based registration pipeline is made to encodes sensor intrinsics into the registration process for further robustness over state-of-the-art.
4. **Model Fusion** A new algorithm for fusing together different generative representations of point cloud data over time for building arbitrarily complex models.
5. **Sampling and Visualization** New stochastic sampling methods for fast inference and several methods for 3D visualization.
6. **Embedded System and Real-World Performance Testing** An embedded, low-power NVIDIA Tegra based system running the aforementioned algorithms in real-time and tested on various challenging real-world LIDAR data captures, from structured urban environments to unstructured outdoor.

1.3 Thesis Outline

The next chapter (Chapter 2) will introduce previous approaches for point cloud data processing as it pertains to 3D perception and explain how our proposed architecture differs. Chapters 3-7 will cover the in-depth technical details of our model and our contributions in accelerating and unifying various 3D perception applications.

The full content chapter-by-chapter breaks down as follows:

- **Chapter 2 – Background**: We cover various historical methods for organizing point cloud data, including voxel grids, Octrees, and NDTs. We show how our approach of compact hierarchical and generative modeling differs in its intended use, abilities, and scope.
- **Chapter 3 – Data-Parallel Gaussian Mixtures**: We introduce the our basic representation used for point cloud data and an algorithm for efficient data-parallel construction. The

proposed model can be seen as a lossy compression scheme, allowing dynamic run-time trade-offs between compression rates and geometric accuracy.

- **Chapter 4 – Registration**: We show how our proposed representation can aid registration and how this can be used as a building block for 3D perception algorithms.
- **Chapter 5 – Hierarchical Mixture Construction**: We introduce a novel method for producing generative hierarchies of 3D point cloud data. We show how to recursively form probabilistic subdivisions of the data through local mixture modeling, and how these subdivisions can provide a maximum likelihood segmentations of the data at multiple levels of detail.
- **Chapter 6 – Inference and Visualization**: We discuss how to perform parallel sampling for stochastic inference about occupancy, how to produce voxelized maps, and also how one may visualize these models as fine-grained isosurface meshes for real-time display on embedded devices.
- **Chapter 7 – Context-Aware Modeling, Fusion and Mapping: A Unified Perception System**: We augment our point cloud models to be able to incorporate known range sensor properties in order to more robustly register together points to models when the amount of spatial overlap may be incomplete. We then discuss how to integrate many of the applications discussed in Chapters 3-6 into a single cohesive perception system.

Chapter 2

Background – Range Data Representations

The production of a structured data representation from raw sensor information is one of the cornerstones of robotics, dating as far back as the late 1960's where Stanford's Shakey Robot used a type of Quadtree-like data structure to organize its own internal map [103]. New data representations have the capacity to enable algorithms and techniques that might have been intractable otherwise, fundamentally reshaping and augmenting a robot's ability to perceive and effectively navigate through the world. In order to contend with the noisy and often spurious data produced by sonar sensors, for example, Moravec, Elfes, and Matthies in the late 1980's developed the probabilistic occupancy grid, and used it to enable many new applications and advances in mapping, navigation, and perception [25, 27, 28, 87, 96].

The choice of data representation is especially important for systems that utilize modern 3D range sensors. Modern LIDAR, structured light, and time-of-flight systems generate massive amounts of data on the order of millions of points per second. The high data throughput of these range sensors makes it extremely difficult if not impossible to perform any kind of complex computation directly on the raw data while preserving real-time or online operation, yet real-time computation is often a critical requirement for mobile robotics applications. Many algorithms therefore simply discard or sub-sample data for computational tractability, even if throwing away data may potentially waste useful sensor information.

For real-time perception applications, an intermediate data representation only helps if it can accelerate further data processing enough to offset its own construction cost. However, this acceleration is often at odds with descriptive fidelity of the original data. Thus, when discussing models for point cloud data, there is an innate trade-off between the descriptive power of the model and its space and time complexity.

In the rest of this chapter, we will review various representations that have been used to model point cloud data for 3D perception and discuss them in terms of their fidelity and complexity trade-offs. We discuss voxel-based structures, parametric models, and hybrid techniques combining elements of both voxelization and parametric modeling. Finally, we will discuss how our approach of compact generative modeling fundamentally differs from previous approaches in its methodology, applicability, and scope.

2.1 Voxel-Based Representations

A fundamental area of world model research concerns the structure of the basic perceptual primitive. Early work tended to favor an object-based approach, using sparse sets of polyhedra, lines, and planes as the fundamental building blocks behind world representation (e.g. early topological maps), but historically these types of approaches have trouble scaling with dense measurements and unstructured terrain. Voxel grid techniques and other types of discretized metric mapping approaches, on the other hand, have proven to be an efficient way to rapidly synthesize large amount of range data, the perceptual primitive in this case being the cell (2D) or voxel (3D).

2.1.1 Dense Voxel Grids

In 1985, Moravec introduced a way to glean more data from range sensors than before by not only recording the depth return itself but also the empty space leading up to it [93]. In this way, free space could be seen as being "carved out" by range sensors. By combining a rasterized grid with a simple Bayesian sensor model for probabilistic occupancy estimation, one could easily calculate degree-of-belief values for cells indicating whether they were empty or occupied and update them over time given new sensor data. Given that these methods are easy to implement and can well handle noisy sensor measurements, Moravec's probabilistic occupancy grids have since been used extensively, enabling many advances in range-based navigation, path planning, and mapping [25, 27, 28, 87, 96].

The transition from 2D cell maps to 3D voxel grids started also around the same time as the development of probabilistic occupancy maps, but had limited applicability due to the computational and memory limitations at the time [132]. For dense grids, increasing the dimensionality from 2D to 3D causes an exponential increase in the number of elements needed to be tracked. Even now, memory can be a limiting bottleneck for modern algorithms that rely on fine-grained dense 3D voxel grids [100].

To circumvent the computational limitations of maintaining a fully 3D dense voxel grid, some applications utilize a degenerate "2.5D" dense representation of range data in the form of elevation maps or multi-level surface maps [71, 111, 112, 141], or multi-volume occupancy grids [20, 97]. In all of these methods, some form of height information is stored in each cell of a 2D rasterized grid. In situations where the world is mostly planar but not necessarily flat (like sloping ground), or in the case of aerial robots, it becomes very advantageous to be able to model ground height. In cases where there exist multiple surfaces that might constitute valid ground (e.g. on and below a bridge), multi-level and multi-volume techniques are able to model these ambiguities when simple elevation maps cannot. Multi-level surface maps have been augmented to incorporate sensor uncertainty [115] and also used to facilitate plane detection [114].

Another problematic aspect of dense voxel grids is that the grid extent and voxel size typically must be statically set before any data acquisition occurs. Given finite memory, there is an inherent tradeoff therefore between how fine-grained the grid can be as well as how large its extent can be. In some applications where the environment may be relatively unknown *a priori*, this introduces an engineering challenge of needing a certain amount of granularity for the perceptual task at hand but also needing to set the size of the grid so that the robotic agent cannot leave its bounds. Toward this end, attempts have been made to make voxel size and grid extent dynamic. On

the topic of voxel size, some have tried to remove the static voxel size constraint by expanding voxels dynamically during data collection [29][47]. On the topic of extent, one popular way to create dynamic extent is by using a dynamically scrolling map, centered around the mobile agent, where data is placed onto the world frame using modulo arithmetic and timestamps to differentiate old from new [73][66]. The scrolling voxel grid still has finite extent in this case, but since the map effectively "moves" with the robot, the robot will never be able to run off the edge of the map. In both of these cases, however, memory efficiency can still be problematic for fully dense 3D grid techniques.

2.1.2 Recursive Subdivisioning and Sparse Techniques

Tackling the problem of memory inefficiency for dense grids most often means tackling the problem of data sparsity. Whether it be through structured light, LIDAR, or sonar, all range sensor data is collected as point samples from the solid outer layers of solid objects. In this way, point clouds can only contain samples of a set of sub-dimensional manifolds and thus are naturally sparse (i.e. range sensors cannot see "inside" volumes, only their surfaces). Even with 2D data, the vast majority of cells in a grid map will be empty, and this situation is made exponentially worse with the transition to 3D. Recursive subdivisioning and sparse representations save memory by not explicitly representing empty space.

Quadtrees

Quadtrees were introduced as a way to effectively compress empty cells in 2D grid maps. Producing a Quadtree is a recursive process: every cell is repeatedly split into four smaller child cells representing the quadrants of its parent. In this way, given a stream of range data points, a tree can be constructed where the nodes are cells each having four children that represent its four quadrants and leaves store the fine-grained occupancy information. This representation obviates the need for shifting or scrolling as new terrain is added because one can simply add more nodes to an existing tree. Space is conserved by not having to represent large empty spaces as many densely packed yet empty voxels, but instead empty space is implicitly defined by the absence of children at a particular node.

Though the early robot "Shakey" used a primitive form of the Quadtree for its internal map representation in the 1960's [103], Finkel and Bentley in 1974 were the first to formally define the modern Quadtree structure [34]. Samet further elaborated on these structures a few years later [124] and would go on to publish many other works in the area of spatial data structures, including an early survey as well as several textbooks [125, 126, 127, 128]. Quadtrees have been used for collision-free path planning [64], compact probabilistic occupancy grids [68], in conjunction with potential fields techniques [109], and augmented to accelerate range queries [31].

Octrees and K-d Trees

Octrees are the 3D analog of quadtrees. Given three dimensions, Octrees split in eight ways (octants) at each node. Octrees have a rich history in both computer graphics and robotics since they were introduced. In 1980, Moravec proposed Octrees for 3D computer graphics in order to

compute Fast Fourier Transforms for illumination calculations [94, 95]. Its uses for geometric modeling [88] and the important robotics-related problem of generating models from successive range scans [16] were explored throughout the 1980's. Applications of Octrees can be seen in path planning [67], collision detection [62], and 3D navigation [110, 148]. More recently, researchers have focused on using statistical techniques in conjunction with Octrees, including coding [54], optimization to searches [9], and approximations for efficient rendering [1]. Octrees have also proven to be highly compressible [8, 46, 91, 129]. A modern open-source implementation of an Octree for probabilistic occupancy mapping can be found in [153].

K-d trees are similar to Quadtrees and Octrees in that they use recursive subdivisions in order to compress empty space, but instead of using a regular splitting criteria, the splits occur as branches on a binary tree and are calculated relative to a space splitting hyperplane derived from the parent along a chosen dimension. Using this process recursively, one can construct a tree that successively splits space into voxels of different sizes. Compared to Octrees, K-d trees have received less attention in the robotics community due to their slower insertions, but they can be highly optimized for range search and ray tracing [55] and have been utilized for SLAM [105] and terrain modeling [147].

Sparse Voxel Techniques

Due to the extreme sparsity of occupied voxels in 3D environments, it sometimes becomes more efficient to simply store only the occupied voxels as a hashed voxel list. Ryde and Hu propose this construction, keeping around any voxel with a more than 0.5 probability of being occupied [120]. In comparison to traditional occupancy grids, ray tracing is not done until the path planning phase, where rays are reconstructed from stored poses. Sparse voxel techniques also appear in the context of 3D terrain classification [146] and more recently for online 3D reconstruction [102]. We use these techniques as well as a way to implement a fast Marching Cubes method for 3D visualization in Chapter 6.

2.2 Parametric Point Cloud Models

Unfortunately, discretizing the world in the form of a voxel-based representation comes at a cost. High spatial frequencies in a scene can be hidden through aliasing and smoothly sloping geometry can appear stair-like. The choice of base voxel size also prevents any type of geometric-based object recognition for small objects approaching the size of a single voxel. Decreasing the voxel size, however, increases computational and memory demands. Discrete representations also offer challenges when using numerical optimization methods. It is often the case that one might want to utilize continuous spatial gradients, for example, in order to perform 3D registration, but discrete representations of spatial data can render continuous optimization techniques impractical or impossible.

Many researchers therefore have been attracted to the idea of using a continuous parametric model to describe the original point cloud data instead of using voxels. If geometric information can be encoded into a continuous and differentiable parametric model, efficient transformations and gradient calculations may be performed over 3D space. Significant space savings can also be

attained by representing large amounts of point data as a compact set of descriptive parameters. However, it is not clear how to build an explicit or implicit parametrization of point cloud into a compact form that provides robust, continuous point estimates.

2.2.1 Planar Models

Many environments, especially those indoors, are composed of mostly rectilinear or planar structures. In these scenarios, it can be advantageous to use a parametric planar object as the basic geometric primitive for perception. Research on this topic first started appearing in the late 1990's [37, 151]. Under this paradigm, the task of point cloud registration, for example, changes to the task of first decomposing the environment into a set of planes and then finding the best transformation of a new set of points onto the set of planes [77, 86, 137]. Various methods have been developed to decompose range data into a set of planes. Plane fitting under the Octree paradigm is discussed in [149]. Multiscale approaches are given in [107, 108]. Tseng *et al.* discuss the problem of growing planes from regions [155].

Of particular interest are techniques using the Expectation Maximization (EM) algorithm [18] to generate low complexity planar models of indoor environments [48, 76, 137, 139]. These EM-derived models reduce the variance of the data, thereby also reducing the effects of noisy measurements, a characteristic that is shared by our proposed models. Planar modeling can be improved using multiple rounds of EM, and can be used to robustly find planes and the main normal directions for rectilinear environments [140]. Though well-suited for some types of indoor environments, planar-based world models may prove too restrictive in cases where clutter, occlusion, lack of structure, or non-uniform sampling predominate the scene; a richer representation is often required for more unstructured types of environments.

Though we also use EM as our main method for model construction, our proposed model uses the full 3D anisotropic Gaussian as its main basis function instead of planes. This formulation still retains the ability to geometrically describe highly planar surfaces if one allows the Gaussian to become degenerate in one dimension during model creation. The fully anisotropic 3D Gaussian, however, can well-describe non-planar pieces of scene geometry as well, providing a richer set of representational types.

2.2.2 Generative Models

Another class of models view the data as a set of independently and identically distributed (*iid*) samples from some unknown latent distribution [56]. These statistical algorithms use the principle of maximum data likelihood to optimize a set of latent parameters that describe the original data. By jointly describing the probability of the entire 3D space, these generative models can make use of a fully Bayesian paradigm for data collection and updating state beliefs. These models often suffer from high construction costs, however, and may require a lot of *a priori* knowledge about the model's structure and the nature of the surrounding physical environment.

For modeling 3D point cloud data, the most common generative model used in the literature is the Gaussian Mixture Model (GMM). Most notably, GMMs have been used successfully to facilitate robust point cloud registration techniques [5, 21, 23, 32, 60], which we will cover in detail in Chapter 4. The work of Jian and Vemuri [60], for example, convert a point cloud into a

GMM by placing a covariance around each point. Though this minimizes the model construction cost, inference then becomes very slow as the GMM is more complex than the original raw points. In contrast to these methods, our proposed hierarchical technique is exponentially faster with respect to the size of the model. A "flat" version must iterate linearly $O(J)$ through all J mixtures, whereas our method is $O(\log J)$. Furthermore, we do not need to specify the number of mixtures *a priori*. Our coarse-to-fine construction allows us to both filter out low-frequency details quickly (floors and walls) and drill down to deeper levels for high-frequency details.

Other work has experimented with hierarchical forms of GMMs for applications like 2D image segmentation [36]. Typically, these methods operate bottom-up, repeatedly grouping together like clusters of points using divergence measures to split and merge the data. For example, Goldberger *et al* [41] construct an iterative EM-like algorithm using KL-Divergence in order to repeatedly merge candidate clusters. In contrast, we adopt a top-down hierarchical approach, motivated by the need to keep the calculation of point-mixture correspondences sparse for 3D point clustering. As such, our approach is more amenable to parallel hardware and is much more computationally efficient.

2.2.3 Normal Distance Transforms

Normal Distance Transform (NDT) methods first discretize the point cloud into voxels and then compute Gaussian distribution parameters for each voxel by simply recording the mean and covariance of all points that fall into it [5, 123]. As such, these family of methods can be seen as a "hybrid approach" to both voxel-based methods and parametric methods. Given that each voxel now contains a probabilistic interpretation of a point falling into it, the likelihood has a well-defined gradient and the optimization over the rigid transformation can be more robust than methods that operate directly over raw points. In fact, a GMM can then be constructed as a weighted sum of the voxel's respective Gaussian parameters. As opposed to point-based GMM constructions, the NDT is a voxel-based GMM construction and thus can provide large time savings, which is especially important for many time-critical applications. Imposing a voxel grid presents the same technical issues and computational limitations as before, however, since the grid boundaries may lie across objects, and small grid sizes produce large amounts of parameters.

Building off the work of [4, 6] for Normal Distance Transforms (NDT), a 3D-NDT method for registration was introduced, and later further refined in [133, 134]. The 3D-NDT shares similarities to our proposed work in that it first establishes a compact model from the data that is then operated over for the purpose of registration. In contrast to our work on registration, however, they do not employ EM, but instead use the voxel occupation of the projected scene or Euclidean distance to voxels to establish correspondence. However, the NDT methods use only an approximation to the true likelihood (sum instead of product) so that the derivatives are easy to calculate, but in contrast we use the EM algorithm and latent correspondences to factorize the joint likelihood completely so that simple closed form solutions may be used.

2.3 A Comparison of our Proposed Method

Our proposed model is novel in its scope and applicability. Many generative models discussed in the literature for point cloud data are "one-off" implementations, where the data model is inextricably linked to a specific algorithm and thus not generalizable across a wide range of perception operations. In contrast, we seek to provide a set of compact generative models that are flexible enough to be suitable for many common operations in 3D perception. Indeed, the hierarchical variant of our model generalizes the NDT under certain assumptions. We will give many examples of this generalizability in subsequent chapters.

Similar to an Octree, one can view our proposed model also as a type of 3D hierarchical spatial decomposition. However, instead of voxels and recursive regular subdivisioning, as in the case of an Octree, we use a hierarchy of soft probabilistic boundaries defined by a series of maximum data likelihood optimization processes. Thus, like an Octree, we produce a structure that has logarithmic spatial access time but does not have the problem of discretization artifacts. Furthermore, the compact and parametric structure of our model provides potentially large space and time savings when trying to operate over the data for common perception applications.

Our proposed method does have several trade-offs over other established structures for 3D point cloud data. In order to obtain the benefits of hierarchical mixture modeling, such as higher fidelity and compact size, we must solve many small EM-based optimization problems. Our proposed model's primary trade-off therefore is in its potentially burdensome construction time. We attempt to mitigate this cost, however, by exploiting point cloud data parallelism as much as possible and designing these models almost exclusively for highly parallel hardware such as FPGAs or GPUs. Toward this end, we borrow from ideas from data parallel adaptations of EM over high dimensional data used in the Machine Learning community [70, 83], but modify these algorithms to be more efficient and well-suited for the problem of 3D point cloud parametrization and segmentation, where our data contains vastly many more points than dimensions and where the sensor capture properties are well-understood and can be used to augment our methods further (see Chapter 7).

Chapter 3

Data-Parallel Gaussian Mixtures

In this chapter, we will introduce the basic underlying point cloud model that we will use and further augment throughout the rest of this thesis.

3.1 Model Construction

Our goal for model construction is to transform the raw 3D points coming from the sensor into a parametric form that conveys higher level statistical geometric entities. A suitable model not only reduces the amount of data needed to describe the world, but it also provides a more tractable statistical representation for perception tasks.

Given that we have an unknown world model, $\mathbf{\Theta}$, and a set of N point samples of surfaces, $\mathcal{Z} = \{\mathbf{z}_i = (x_i, y_i, z_i)^T\}$, we would like to find the most likely $\mathbf{\Theta}$ that "explains" $\mathcal{Z}$. That is, we wish to find a collection of parameters $\hat{\mathbf{\Theta}}$ such that

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, p(\mathcal{Z}|\mathbf{\Theta}) \tag{3.1}$$

We choose to parametrize $\mathbf{\Theta}$ by a convex combination of J probability distribution functions, or mixtures. This basic formulation will form the common element for all subsequent models. Using the mixture model, each individual point in the point cloud, $\mathbf{z}_i$, has a probability defined by

$$p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_{j=1}^{J} \pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j) \tag{3.2}$$

where π_j represents our mixing weights and $\sum_{j=1}^{J} \pi_j = 1$.

The total probability of a point cloud is then the product of the individual point probabilities, given that each point is an *iid* sample of the world.

$$p(\mathcal{Z}|\mathbf{\Theta}) = \prod_{i=1}^{N} p(\mathbf{z}_i|\mathbf{\Theta}) = \prod_{i=1}^{N} \sum_{j=1}^{J} \pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j) \tag{3.3}$$

Using this model, we can re-interpret the point cloud data (PCD) as a sampling of N points generated from J distinct probability distributions, with prior sampling probabilities given by π_j. If we parametrize each function by a small set of parameters and constrain J to be small ($J \ll N$), we can compress the possibly millions of raw 3D points into a collection of easily storable functions that describe the spatial characteristics of the original cloud.

Unfortunately, calculating the optimal world model directly from the data is intractable. If we wish to maximize data likelihood, the common strategy is to first transform the total data likelihood product into a sum by taking its logarithm. However, if we look at the log likelihood,

$$\ln p(\mathcal{Z}|\mathbf{\Theta}) = \sum_{i=1}^{N} \ln \sum_{j=1}^{J} \pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j) \tag{3.4}$$

we see a sum inside the logarithm due to the mixture formulation, which in general we cannot solve analytically for each component $\mathbf{\Theta}_j$.

To produce a tractable likelihood function, we need to introduce a NxJ set of binary correspondence variables $\mathcal{C} = \{c_{ij}\}$ for each point-mixture component combination $\{\mathbf{z}_i, \mathbf{\Theta}_j\}$ that serve as labels for the mixture component to which the point belongs. In other words, each point $\mathbf{z}_i$ will have J correspondence variables with values 0 or 1, where 1 means "belongs to this mixture" and 0 means "does not belong to this mixture." Then we can represent the inner sum as a product of terms with binary exponentials and thus completely factorize the log likelihood,

$$\ln p(\mathcal{Z}, \mathcal{C}|\mathbf{\Theta}) = \sum_{i=1}^{N} \sum_{j=1}^{J} c_{ij} \left\{\ln \pi_j + \ln p(\mathbf{z}_i|\mathbf{\Theta}_j)\right\} \tag{3.5}$$

We still can't solve this factored form because we don't know $\mathcal{C}$. However, if we somehow did know the correspondence variable c_{ij} for each $\mathbf{z}_i$, we could easily maximize this joint likelihood by setting $\mathbf{\Theta}_j$ to the sample mean and sample covariance of all points for which $c_{ij} = 1$. Alternatively, if we knew $\mathbf{\Theta}$, we could easily find the probability of a point's correspondence with a particular mixture component using Bayes' rule and calculating,

$$p(c_{ij}|\mathbf{z}_i, \mathbf{\Theta}) = \frac{p(\mathbf{z}_i|c_{ij}, \mathbf{\Theta}_j)p(c_{ij}|\mathbf{\Theta}_j)}{p(\mathbf{z}_i|\mathbf{\Theta})} \tag{3.6}$$

$$= \frac{\pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j)}{\sum_{j'}^{J} \pi_{j'} p(\mathbf{z}_i|\mathbf{\Theta}_{j'})} \tag{3.7}$$

To summarize the above equation, the probability that a certain point z_i belongs to the jth mixture component is the data probability of that mixture component having generated the point normalized by the total point probability given all other mixture components. Also note that in the expression above, $p(\mathbf{z}_i|\mathbf{\Theta})$ can be seen as a marginalization over mixture components of the joint distribution $p(\mathbf{z}_i, \mathbf{c_i}|\mathbf{\Theta})$ where $p(c_{ij}|\mathbf{\Theta}_j) = \pi_j$ and thus $p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_j p(c_{ij}|\mathbf{\Theta}_j)p(\mathbf{z}_i|c_{ij}, \mathbf{\Theta}_j)$.

Since we do not know the correspondence labels $\mathcal{C}$, nor do we know the model parameters $\mathbf{\Theta}$, we can use the Expectation Maximization (EM) algorithm [18] to iteratively estimate both the labels and cluster parameters alternately with the following two steps:

E Step: calculate the expected values of $\mathcal{C}$ given $\boldsymbol{\Theta}^{old}$:

$$\mathbb{E}[c_{ij}] = \frac{\pi_j^{old} p(\mathbf{z}_i|\boldsymbol{\Theta}_j^{old})}{\sum_{j'}^{J} \pi_{j'}^{old} p(\mathbf{z}_i|\boldsymbol{\Theta}_{j'}^{old})} \tag{3.8}$$

M Step: maximize the expected log-likelihood with respect to $\boldsymbol{\Theta}$ using fixed point-to-component associations by holding $\mathbb{E}[c_{ij}] \stackrel{\text{def}}{=} \gamma_{ij}$ as constants:

$$\boldsymbol{\Theta}^{new} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_{ij} \gamma_{ij} \{\ln \pi_j + \ln p(\mathbf{z}_i|\boldsymbol{\Theta}_j)\} \tag{3.9}$$

We iteratively hold $\mathbb{E}[\mathcal{C}]$ as a constant and optimize over $\boldsymbol{\Theta}$ and then hold $\boldsymbol{\Theta}$ as a constant and optimize over $\mathbb{E}[\mathcal{C}]$. This procedure is guaranteed to converge and will lead to a local maximum data log likelihood in the parameter space over $\boldsymbol{\Theta}$.

3.2 Gaussian Mixtures

In our world model, we assume that the PCD has been generated by a discrete set of 3D Gaussians, producing a Gaussian Mixture Model (GMM). Thus, given a set of J 3D Gaussians with $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_j = (\pi_j, \mu_j, \boldsymbol{\Sigma}_j)\}$, our function describing the likelihood of a point is a linear combination of Gaussians,

$$p(\mathbf{z}_i|\boldsymbol{\Theta}) = \sum_j \pi_j \mathcal{N}(\mathbf{z}_i|\boldsymbol{\Theta}_j) \tag{3.10}$$

where $\sum_j \pi_j = 1$ and $\mathcal{N}(\cdot)$ is the multivariate (3D) Gaussian probability distribution.

Each Gaussian has nine free parameters representing its mean (three) and covariance (six, due to symmetry). If we wish to produce a compact model, therefore, we must therefore restrict our parameters such that $J \ll \frac{1}{9}N$. As we will show later in this proposal, due to the descriptive nature of the Gaussian and the non-randomness of point cloud data, this restriction is easily obtained while allowing very high model fidelity.

3.3 EM Parallelization for GMMs

Framing the model construction as a Gaussian mixture segmentation problem over point cloud data offers several benefits. The EM algorithm is nicely parallelizable due to the relative lack of interactions among points, low-dimensional parameter space, and mismatch in size of data points compared to model parameters. Additionally, whereas many segmentation algorithms must rely on nearest neighbor calculations at some point (e.g. region growing) or operations over local neighborhoods, each point in the point cloud for EM segmentation needs to compare itself to one of only a few cluster parameters. Thus, we do not need to impose any special spatial data structure like Octrees to be able to quickly calculate nearest neighbors. The only interaction among the points comes during the calculation of the normalization terms of the M step, which is the sum of the expectations of the correspondence variable.

In the M step on the k^{th} iteration, we calculate

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_i \gamma_{ij} \mathbf{z}_i}{\sum_i \gamma_{ij}} \tag{3.11}$$

$$\Sigma_j^{new} = \frac{\sum_i \gamma_{ij} (\mathbf{z}_i - \boldsymbol{\mu}_j^{new})(\mathbf{z}_i - \boldsymbol{\mu}_j^{new})^T}{\sum_i \gamma_{ij}} \tag{3.12}$$

$$\pi_j^{new} = \sum_i \frac{\gamma_{ij}}{N} \tag{3.13}$$

Note that we can remove the dependence on μ_j^{new} when calculating the bulk of $\boldsymbol{\Sigma}_j^{new}$ by rewriting as

$$\Sigma_j^{new} = \frac{\sum_i \gamma_{ij} \mathbf{z}_i \mathbf{z}_i^T}{\sum_i \gamma_{ij}} - \frac{\boldsymbol{\mu}_j^{new} \boldsymbol{\mu}_j^{newT}}{\sum_i \gamma_{ij}} \tag{3.14}$$

This shows us that the following quantities constitute the sufficient statistics for our segmentation:

1. $\sum_i \gamma_{ij} \mathbf{z}_i \mathbf{z}_i^T$, for all $j = 1..J$ (6J numbers)
2. $\sum_i \gamma_{ij} \mathbf{z}_i$, for all $j = 1..J$ (3J numbers)
3. $\sum_i \gamma_{ij}$, for all $j = 1..J$ (J numbers)

The calculation of the above three quantities requires:

1. Calculation of each γ_{ij}.
2. A weighted sum over all first and second moments.

The former requires information about all J mixtures but no shared information among points. The latter requires sharing data over all N points but no data is needed from the J surfaces once γ_{ij} are calculated. Thus, we can use point level parallelism in calculating γ_{ij} and mixture level parallelism to calculate the weighted moment sums.

Past work from the machine learning community has looked at data parallel algorithms for EM to find GMM parameters, but were tailored toward specific high dimensional machine learning applications [70] [83]. Our application of point cloud data segmentation differs in scope in that we deal strictly with 3D geometric point data. Our application therefore operates under a different set of design considerations:

1. We can afford a certain representational richness that would otherwise be computationally intractable in high dimensional scenarios.
2. We can devote our parallelization efforts toward point-level parallelism as this is where the bulk of the work lies.

Given the former consideration, unlike other work, we do not restrict our covariances to be spherical or axis-aligned elliptical, allowing non-zeros on the non-diagonal. In our particular application we need the set of Gaussians to correspond with real 3D objects, and these objects will often not align nicely along the principle axes. Thus, we calculate the full 3D covariance matrix

for our application, where in previous work it was assumed that inverting a high dimensional matrix for the expectation calculation would be prohibitively expensive if the matrix is non-sparse. For the latter consideration, we can tailor our algorithms solely toward point-level parallelism given that the feature space is so small. Any optimization toward removing point-level data interactions will thus have large consequences in terms of speed, and so this is something we focus on when implementing these algorithms on the GPU.

Chapter 4

Registration

The first application we will show can be accelerated by our proposed model is point cloud registration. Registration of point cloud data, or estimation of the relative pose between two point clouds, forms the basis of many algorithms in 3D vision applications such as object matching, SLAM, body/head pose estimation, medical imaging, and environment reconstruction.

In this Chapter, we establish several techniques to quickly and robustly register point clouds together using the mixture model formulated in Chapter 3. The mixture representation sidesteps the costly problem of matching points to points since incoming data only need to be compared with the mixtures. The chosen parametrization of the mixtures (as Gaussians) leads to a smooth data likelihood function with a well-defined gradient as well as a closed form MLE approximation under latent correspondences. This representation thus forms the basis for three novel registration algorithms:

1. REM-Seg (Robust EM Segmentation): Parallelized gradient descent combined with an annealing method to find the 6-DOF rigid transformation between the incoming point cloud and an augmented Gaussian Mixture Model. Of the three, most robust to large displacements, but relatively slow.
2. MLMD (Maximum Likelihood Mixture Decoupling): Uses the same augmented mixture model as REM-Seg but instead solves the registration problem as an approximate Expectation Maximization algorithm. Fast and robust to large displacements.
3. MDME (Mixture Decoupled Mahalanobis Estimation): Uses the same approximate EM algorithm of MLMD, but instead uses covariance eigendecomposition to reframe its optimization as an efficient weighted point-to-plane minimization. Fastest and most accurate, but only if initial displacement is not too large.

All three variants are novel statistical algorithms enabled by generative point cloud modeling that provide fast, compact, and accurate solutions for 3D PCD registration.

We build on the idea of using Gaussian Mixtures to represent point clouds and provide several methods to find our registration solution. However, in contrast to previous work connecting Gaussian Mixtures to point cloud registration, we position the model construction as an additional optimization step before the actual registration occurs to decouple the mixtures from the points ("*mixture decoupling*"). This reduces the computational and representational cost of our methods.

At the same time, we estimate non-uniform mixture weights and anisotropic covariances, according to the principle of maximum data likelihood. This is also in contrast to previous techniques that utilize per-point Gaussians, uniformly weight each mixture, and restrict covariances to be the same isotropic value per point, which we show to be ill-posed in the case of range data exhibiting high degrees of anisotropic uncertainty.

We show how producing compact mixture model facilitates both faster and more robust registration by first optimizing over the mixture parameters (decoupling the mixture weights, means, and covariances from the points) before optimizing over the $6DOF$ registration parameters. We test these algorithms against other GPU-accelerated registration algorithms on simulated and real data and show that the described methods scale well to large numbers of points, have a wide range of convergence, and are suitably accurate for 3D registration.

4.1 Previous Approaches

Two broad categories of point cloud registration algorithms exist: those that do direct point-to-point or point-to-local-surface matching, and those that utilize some type of statistical model. The former category includes the widely used Iterative Closest Point (ICP) algorithm [3, 11], and the latter includes registration algorithms based on Gaussian Mixture Models (GMM) [59] or Normal Distance Transforms (NDT) [134].

4.1.1 Iterative Closest Point

Iterative Closest Point is an iterative procedure that cycles between trying to find correspondences from one cloud (or mesh) to the other, and minimizing the summed distance between all previously established correspondences under a set of transformation parameters. Since the correspondences may not be correct, the algorithm iterates between matching and minimizing, with the hope that successive iterations will provide more and more accurate correspondences and therefore the minimization of the summed distance will provide the correct pose displacement.

The basic steps of ICP are as follows, which are repeated until convergence:

1. Sample points from each cloud (or use entire cloud)
2. For each point in a given cloud, find a corresponding point (e.g. via minimum Euclidean distance)
3. Assign weights to the correspondences
4. Reject outlier pairs
5. Apply an error metric to the current transform (e.g. sum of squared error)
6. Use an optimization technique to minimize the error metric (e.g. SVD, gradient descent, Newton's method, etc.)

ICP is the *de facto* standard for many registration applications, but can easily succumb to misalignment when point matching between clouds is inappropriate, for example, if two surfaces are sampled very differently or there is a large transformation between clouds. ICP is somewhat ill-posed given the fact that each point cloud consists of non-uniform point samples and thus

a given point is very unlikely to have an exact corresponding point in the other point cloud. Furthermore, if outliers are not properly caught and discarded, then trying to minimize outliers using the squared distance will not be robust and can cause the algorithm to diverge. Thus, the original ICP algorithms tend to converge poorly when subjected to severe noise and large pose displacements without a good initial estimate.

Since the advent of ICP in 1992, there have been hundreds of papers published on the topic of point cloud registration. To make the ICP algorithm robust to noise, outliers, and sampling differences, many variants of the classic ICP algorithm have been introduced over the last decade to boost both its robustness and computational characteristics [116] [92]. For computational improvement, most modifications focus on circumventing the expensive nearest neighbor step during the matching step. Some effective advancements toward this goal include the use of Octrees for approximate nearest neighbor calculations or fast projections [116].

For improvements in robustness, many approaches have tried to eschew point-to-point squared-distance metrics matching in favor of methods that offer robust estimators or perform projective or point-to-plane matching. Some notable works include point to quadratic surface approximations [92] and robust M-Estimators for Euclidean distance solved by Levenberg-Marquadt optimization [35].

All these enhancements can add to the complexity of the original method, and often offer severe tradeoffs between robustness and speed. Whereas the original point-to-point formulation is guaranteed to converge, correspondence weighting and projective or point-to-plane matching can cause divergence, especially in cases of large pose displacement. Extensions to the minimization step as an M-Estimator are often very expensive to calculate, as opposed to the least squares Euclidean distance, which can be solved in closed form for rigid transformations using Horn's method [52]. Furthermore, there are many cases where range images are not available (unstructured LIDAR output), making techniques like projective matching and neighborhood-based normal calculations very expensive.

4.1.2 Statistical Algorithms

The Softassign algorithm [40] was the first in a new class of methods that framed ICP into a statistical framework. Developed as an effort to replace the *ad hoc* procedure of correspondence matching and distance minimization with a theoretical footing in statistics, algorithms in this class are derived from general assumptions about the nature of point clouds, their underlying noise properties, and how the minimization step relates to some type of energy function or probabilistic estimate. Thus, these methods facilitate robust outlier rejection, proofs of convergence, and various continuous optimization procedures with differentiable objective functions.

Under a statistical framework, the iteration of correspondences and minimization of summed distances can be seen as a type of Expectation Maximization (EM) procedure [18]. By establishing the connection between ICP's point matching step with calculating expectations over latent correspondence variables (E Step), and ICP's distance minimization step with the maximization of some bound on the data likelihood (M Step), many of the EM-based formulations actually generalize back to ICP under various basic statistical assumptions about the input data [43, 130]. Additionally, if each point is given a Gaussian noise parameter, the total point cloud can be interpreted as a type of Gaussian Mixture Model (GMM). Thus, most robust registration techniques

explicitly utilize a GMM representation for point cloud data (PCD) to derive claims and proofs about robustness and convergence [14, 33, 43, 51, 98]. Unfortunately, these algorithms tend to be much more complex than ICP, and their robustness often comes at a high computational and representational cost.

Softassign used a simulated annealing scheme borrowed from statistical physics to add additional robustness. Shortly afterwards Mixture Points Matching (MPM) [14] and EM-ICP [43] derived multiple links from Gaussian noise assumptions. Both authors noted how point sets with Gaussian noise were equivalent to a Gaussian Mixtures if each point was interpreted as being generated by some Gaussian with a set isotropic covariance. Thus, the probability measure of another point cloud having been generated by the same Gaussian Mixture set is well-defined and its rigid transformation can be solved in closed form for isotropic covariances. Also, using annealing for robustness, these methods established the process of interpreting PCD as a Gaussian Mixture with each point representing a mixture mean with an associated (isotropic) covariance. It should be noted that, similar to our work, EM-ICP also recognized the inherent scalability problems of point-level GMMs and thus devised a very primitive form of mixture decoupling using a sphere decimation technique.

Tsin and Kanade derived a correlation-based approach called Kernel Correlation (KC) [142], followed by Jian and Vemuri with GMMReg [57, 59]. The latter method minimizes GMM-to-GMM L_2 distance, a metric that is heavily related to Tsin and Kanade's correlation-based approach. Instead of utilizing an EM-based framework, however, these methods directly optimize over a cost function containing all N^2 point pairs. Since there is no explicit correspondence step (as in EM), these methods have difficulty scaling. Tsin and Kanade attempted to get around the scalability problems by numerical differentiation over grids, while GMMReg is restricted to small or subsampled point sets. Maintaining all point-point correspondences, however, results in a very robust algorithm since Gaussian point-pair "distances" act as robust loss functions.

Later, the Coherent Point Drift (CPD) [98, 99] algorithm was introduced as a method similar to EM-ICP, having both explicit E and M Steps, but with the addition of isotropic covariance estimation into the M Step in lieu of the annealing technique used by SoftAssign, MPM, EM-ICP, and GMMReg. ECMPR [51] would later extend CPD's covariance estimation to fully anisotropic covariances, solving the optimization via Expectation Conditional Maximization or ECM [90], an EM-like framework that performs several M-Steps in succession after each E Step, holding all parameters but the variable to be optimized constant.

Like CPD and ECMPR, Generalized ICP (G-ICP) [130] extended ICP to include covariance estimation. Functioning as a bridge between methods like ICP and EM-ICP, G-ICP uses the nearest neighbor criterion of ICP along with a generalized M Step that, depending on the choice of the type of covariance estimation, behaves like point-to-point ICP, point-to-plane ICP, or a plane-to-plane ICP. However, instead of incorporating covariance estimation into their M Step, they estimated it separately by looking at local (20-nn) neighborhoods around each point.

Building off the work of [4, 6] for Normal Distance Transforms (NDT), a 3D-NDT method for registration was introduced, and later further refined in [133, 134]. Similar to G-ICP's data-driven covariance estimation apart from the registration optimization itself, the NDT is a way to produce a smaller set of means and covariances through voxelization. Instead of neighborhoods around each point producing a covariance estimate, as in G-ICP, NDT methods simply voxelize and then record the mean and covariance of points that fall within that voxel. As opposed to point-

based GMM constructions, the NDT is a voxel-based GMM construction and thus can provide large time savings, which is especially important for many time-critical applications. The 3D-NDT shares similarities to our proposed work in that it first establishes a compact model from the data that is then operated over for the purpose of registration. In contrast to our work, however, they do not employ EM, but instead use the voxel occupation of the projected scene or Euclidean distance to voxels to establish correspondence (most versions therefore are singly linked, though [134] uses the 4-nearest neighbors), and then the implicit M-Step is solved through Newton's method over a modified GMM-to-GMM L2 distance. Thus, one can then interpret the 3D-NDT methods in terms of a GMMReg variant that first performs a type of mixture decoupling through voxelization.

More recently, JRMPC [33] decouples the means from the points and does so without resorting to voxelization like the NDT methods. However, it uses a much larger number of isotropic mixtures (J=0.6N). JRMPC finds the GMM jointly from all available data in batch, interleaving the optimization of the model with the calculation of the transformation parameters under an ECM framework. In contrast, we perform the optimization of the model to completion first before recovering the pose transformation. We view our construction to be more robust since, before the geometry has been fully defined, the transformation calculation could be ill-posed. Furthermore, JRMPC's lack of anisotropy precludes compactness, a necessary component for scalability.

Refer to Table 4.1 for a short breakdown of the differences between these related works and the registration algorithms detailed in this Chapter (REM-Seg, MLMD, MDME).

4.2 Gaussian Mixture-Based Approaches

Our approach distinctly differs from previous approaches in that we first build a rich and compact point cloud model upon which we perform our optimization. To explain this difference in detail, we will first explain how previous work has approached the modeling problem, and then introduce our contrasting approach ("Mixture Decoupling").

4.2.1 Traditional Approaches

For most traditional GMM-based methods, given a point cloud with N points $\mathbf{z}_i \in \mathbb{R}^3$, the probability of an arbitrary point in space $x \in \mathbb{R}^3$ is defined to be,

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(x|\mathbf{z}_i, \sigma^2 \mathbf{I}) \tag{4.1}$$

where $\mathcal{N}(x|\mathbf{z}_i, \sigma^2\mathbf{I})$ is the multivariate Gaussian with mean and covariance $\{\mathbf{z}_i, \sigma^2\mathbf{I}\}$. Thus, the means are the point locations of the modeled point cloud, and all covariances are controlled by a single isotropic bandwidth parameter, σ^2. Implicit in this construction is that the mixture vector, π_i is uniformly constant for every point ($\pi_i = \frac{1}{N}$). In other words, if each point has Gaussian uncertainty, then the collection of these points as a PDF (the linear combination of their individual Gaussian PDF's) forms a GMM. The GMM, being a continuous and valid PDF,

Method	Year	Multiply Linked	Probabilistic Correspondences	Covariance Estimation	Anistropic	Compact
ICP [3]	1992					
SoftAssign [40]	1998	✓	✓			
MPM [14]	2000	✓	✓			
EM-ICP [43]	2002	✓	✓			
KC [142]	2004	✓				
GMMReg [57]	2005	✓				
CPD [99]	2006	✓	✓	✓		
3D-NDT [85]	2007			✓	✓	✓
G-ICP [130]	2009			✓	✓	
ECMPR [51]	2011	✓	✓	✓	✓	
NDT-D2D [133]	2012	✓		✓	✓	✓
JRMPC [33]	2014	✓	✓	✓		
REM-Seg [24]	-	✓		✓	✓	✓
MLMD [22]	-	✓	✓	✓	✓	✓
MDME	-	✓	✓	✓	✓	✓

Table 4.1: **A Comparison of Probabilistic Registration Methods.**
Multiply Linked: many-to-one or many-to-many correspondences (robustness under non-uniform sampling)
Probabilistic Correspondences: fully probabilistic correspondences or not (as opposed to kernel-based or nearest neighbor approaches). Allows for application of EM, GEM, or ECM.
Covariance Estimation: Improves convergence as opposed to statically set covariances or simulated annealing methods.
Anisotropic: non-spherically shaped mixture covariance. Most earlier methods restrict covariances to be uniformly isotropic across all mixtures. This improves robustness through better local shape alignment.
Compactness: Given PCD of size N, compact models are those that have a number of mixtures $J \ll N$. This addresses scalability problems with point-based GMM representations.

thus forms the foundation for robust probability estimates, noise handling, and EM-based energy minimization or likelihood maximization methods for registration.

Though registration algorithms designed around the GMM/EM concept have been established in the literature to be much more robust than ICP [43], ICP still remains the industry standard for 3D PCD alignment in vision algorithms due to its simplicity and speed. One problematic aspect of existing GMM/EM registration methods stems from the choice of representation. Though it is theoretically desirable to assign a covariance around each point, this representation is now much more complex than the original point cloud and now has a number of means equal to the number of points. Furthermore, point-based GMM approaches operate under the assumption that each point should have a corresponding point (match) in the other cloud, when, in fact, sampling differences make this unlikely. The fundamental assumption that a point is generated by a single Gaussian source doesn't actually correspond to the reality of non-uniformly scanned geometry, which is that points belong to surfaces, not other points.

Intuitively, it should be noted that point samples representing pieces of the same local geometry could be compressed into smaller clusters, with the local geometry encoded inside the covariance of that cluster. Thus, if the point cloud is of size N, it is possible to adequately describe it by a smaller set of J means and covariances.

4.2.2 Our Approach: Mixture Decoupling

The former discussion motivates the need to add a preprocessing step by first finding a compact but representative model before attempting to establish correspondences or minimize log-likelihood. This is because, in terms of computation, the complexity of EM is dependent on the size of the model. Furthermore, the accuracy and descriptive power of the model directly affects the convergence rate and robustness of the algorithm. Therefore, we propose to *optimize with respect to the representation first* for the express purpose of aiding registration.

We do this by modifying the normal EM procedure to first maximize the model data over the set of all possible GMMs of a given size J, denoted as $\mathbf{\Theta}$, without restriction to the structure of the covariance, the placement of the means, and the mixture weighting. Given point clouds $\mathcal{Z}_1, \mathcal{Z}_2$ and some unknown transformation $T(\cdot)$ representing the spatial relationship between them, this forms a two-step optimization problem for the form,

$$\textbf{Step 1: } \hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, p(\mathcal{Z}_1|\mathbf{\Theta}) \tag{4.2}$$

$$\textbf{Step 2: } \hat{T} = \underset{T}{\operatorname{argmax}}\, p(T(\mathcal{Z}_2)|\hat{\mathbf{\Theta}}) \tag{4.3}$$

After mixture decoupling (Eq. 4.2-Step 1), we end up with a more descriptive, compact representation of the original data. Though this preprocessing step does not come for free (unlike the traditional approaches described in Section 4.2), as long as the model is sufficiently compact and well-suited to the data, the entire process can actually be sped up considerably. Fig. 4.1 shows this situation pictorially. If each point cloud is of size N, a typical GMM/EM registration algorithm will contain N^2 potential connections and thus scale poorly. If we first compress our cloud to size J, where $J \ll N$, then we have two different optimization procedures of size NJ. Given that $J \ll N$, we will be able to produce a method that is much more efficient overall ($O(N^2)$ to $O(N)$).

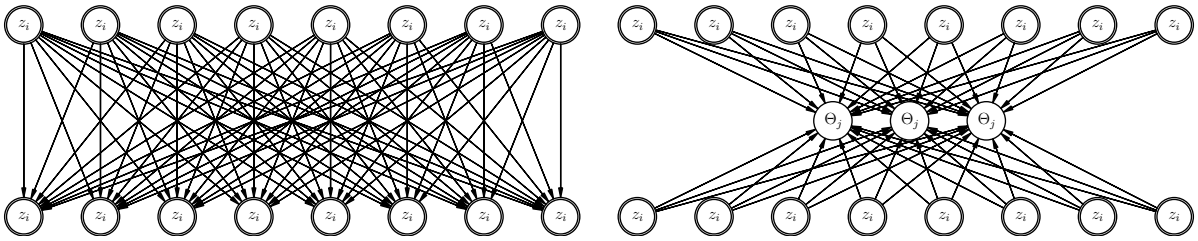


Figure 4.1: **Mixture Decoupling** *Left:* Given two similarly sized point clouds of size N, point-based GMMs produce N^2 potential matches. *Right:* Decoupling points from mixtures into a smaller, more descriptive set of latent $\mathbf{\Theta}$ produces two separate procedures of size $O(JN)$. If $J \ll N$, this effectively linearizes the registration process with respect to N.

Chapter 3 discussed in detail a highly efficient way to perform mixture decoupling by leveraging data parallelism. We can utilize these algorithms to quickly transform a point cloud into a set of parametric probabilistic functions so that the spatial representation of surfaces can be continuous and parametrically sparse.

Thus, registration as a two-step optimization problem (Eq. 4.2 and 4.3) then involves taking the representation described in Chapter 3 and using it to find the most likely rigid spatial transformation given a new set of range data representing the same scene.

In the rest of this chapter, we will detail three different methods that utilize the models of Chapter 3 to efficiently and robustly register together new sets of point cloud data. Each of the three methods uses the model described in Chapter 3 to solve Step 1 (Eq. 4.2), but vary widely in their approach for how Step 2 (Eq. 4.3) is solved.

We briefly discuss the differences in how Step 2 is solved below:

1. **REM-Seg** [24] relies on parallelized gradient descent, bounded expectations, line search, and simulated annealing. It is the most robust to large displacements, but is the hardest to scale to real-time speeds.
2. **MLMD** [22] unifies model creation and registration using an augmented two-step EM procedure. MLMD approximates isotropic covariances in its M-Step, giving an efficient closed form minimizer.
3. **MDME** also uses an EM unification procedure similar to MLMD, but approximates Mahalanobis distance minimization in its M-Step as a set of weighted point-to-plane distances using covariance eigendecompositions. Given a linear approximation of rotation along with the small-angle assumption, this formulation results in the most efficient and robust minimizer of the three methods so long as the relative angles between point clouds are small enough. MDME forms the base algorithm for the techniques introduced in Chapter 7.

4.3 REM-Seg and Gradient Descent

Our given GMM formulation serves as the basis for our scene. Given a new set of N 3D points, we can look at the probability of our scene having generated those points according to our gen-

erative model. For maximum likelihood registration, we wish to optimize over some spatial transformation in 3D space. Thus, we can minimize over the negative log-likelihood of the data with respect to the transformation parameters to find the most likely pose difference. One natural idea is to try gradient descent to find this minimum. Our registration algorithm utilizing gradient descent is called REM-Seg (Robust EM Segmentation) [24].

Because the particular problem of point cloud registration involves a vast amount of points, we can parallelize the computation of the gradient by computing as much of it as possible independently and in parallel (e.g. with CUDA threads on a GPU) and then doing a logarithmic number of reductions to obtain the final gradient step. Minimizing over the negative log-likelihood will provide the MLE spatial transformation of the data to the scene, and will allow us to register together successive 3D measurements and recover the motion of the sensor.

Given that we have a world model parameter set $\mathbf{\Theta}$, we can describe the registration problem as follows: Given a set of 3D points, we can look at the probability of a scene having generated those points according to some surface model. If the probability of a point is $p(\mathbf{z}_i|\mathbf{\Theta})$, where $\mathbf{\Theta}$ is the set of parameters describing the scene, then the total probability of an entire scan of data is $p(\mathcal{Z}|\mathbf{\Theta}) = \prod_i p(\mathbf{z_i}|\mathbf{\Theta})$. In our problem, we wish to optimize over some spatial transformation in 3D space. Parametrizing our rigid transformation by a 3D vector $\mathbf{t} = (t_x, t_y, t_z)^T$ and a unit quaternion $\mathbf{q} = (u, v, w, s)^T$, we can minimize over the negative log-likelihood of the data with respect to the transformation parameters,

$$\min_{\mathbf{q},\mathbf{t}} -\ln p(T(\mathbf{Z}|\mathbf{q},\mathbf{t})|\mathbf{\Theta}) = -\sum_i \ln p(T(\mathbf{z}_i|\mathbf{q},\mathbf{t})|\mathbf{\Theta}) \tag{4.4}$$

Using the EM algorithm, our surface model is just the collection of Gaussians and their respective mixing parameters, $\mathbf{\Theta} = \{\mathbf{\Theta}_j = (\pi_j, \mu_j, \mathbf{\Sigma}_j)\}$, so our data log-likelihood becomes

$$\ln p(T(\mathcal{Z}|\mathbf{q},\mathbf{t})|\mathbf{\Theta}) = \sum_i \ln\{\sum_j \pi_j \mathcal{N}(\tilde{\mathbf{z}}_i|\mathbf{\Theta}_j)\} \tag{4.5}$$

where $\tilde{\mathbf{z}}_i = R(\mathbf{q})\mathbf{z}_i + \mathbf{t}$ and $R(\mathbf{q})$ is the 3x3 rotation matrix derived from the unit quaternion $\mathbf{q}$. Note the similarity to the likelihood function given by Equation 3.4 maximized during model creation. In this step, however, we have a fixed $\mathbf{\Theta}$ and we optimize over $\mathbf{q}$ and $\mathbf{t}$. As before, we cannot find a closed form solution due to the sum inside the logarithm, but we can easily take the gradient of the negative log likelihood with respect to each transformation parameter,

$$\nabla_{\mathbf{t}} = \frac{1}{N}\sum_i \frac{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)(\tilde{\mathbf{z}}_i - \mu_j)^T \mathbf{\Sigma}_j^{-1}}{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)} \tag{4.6}$$

$$\nabla_{\mathbf{q}} = \frac{1}{N}\sum_i \frac{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)(\tilde{\mathbf{z}}_i - \mu_j)^T \mathbf{\Sigma}_j^{-1} \mathbf{\Xi}_{\tilde{\mathbf{z}}_i}}{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)} \tag{4.7}$$

where $\mathbf{\Xi}_{\tilde{\mathbf{z}}_i}$ is the skew symmetric matrix of $\tilde{\mathbf{z}}_i$, the transformed point,

$$\mathbf{\Xi}_{\tilde{\mathbf{z}}_i} = \begin{bmatrix} 0 & -\tilde{z}_i & \tilde{y}_i \\ \tilde{z}_i & 0 & -\tilde{x}_i \\ -\tilde{y}_i & \tilde{x}_i & 0 \end{bmatrix} \tag{4.8}$$

The gradient with respect to the quaternion only gives updates to (u, v, w), with s fixed to 1. For more details, see [150]. Finally, we note that, like the E step of the segmentation algorithm, the $\hat{\gamma}_j$ values can be calculated in parallel and then summed as a reduction.

4.3.1 Extensions

As the algorithm has been presented thus far, we have a few free parameters such as the number of clusters, the step size to take along the gradient, and possible singularities when inverting the covariance matrix of a cluster. For noisy data, outliers should also be handled. We give our extensions to solve these problems in the next few sections.

Oversegmentation

One important parameter to consider when segmenting is J, the amount of clusters to use. To obviate the need for model selection or some other type of parameter search, we simply oversegment and then drop clusters without a lack of adequate support. This process also prevents singularities when doing EM for the GMM parameters, since clusters with low support produce low-rank covariance matrices. Since the final generative model is the linear superposition of Gaussians, it is perfectly acceptable in terms of the gradient calculation, for example, to have overlapping Gaussians describe the same piece of real geometry.

Robust Boundaries

Unfortunately, range data is often corrupted by several sources of noise, from spurious readings or discretization errors in the sensor. Similarly, operations on point cloud data are often burdened by occlusion and sampling sparsity when the objects are far away or at bad grazing angles to the sensor. These aspects make the task of finding and filtering outliers very important, as they may adversely affect the performance of the point cloud segmentation and registration algorithm.

To reduce computational costs and enforce hard boundaries, some past work has used non-overlapping labels when doing EM for purely planar decomposition [86]. To achieve this, during the EM algorithm, all correspondence variables are set zero if they do not have the maximal expectation over all clusters. This method closely resembles the elliptical k-means algorithm, where the Mahalanobis distance from the center of the cluster is used instead of the Euclidean distance.

We adopt a similar approach, but make some important modifications to filter out outliers. Instead of a hard boundary, or a purely soft boundary as the normal EM approach makes, we employ *robust* thresholding, where a correspondence is zero if its expectation is less than or equal to ξ times the maximum expected correspondence, where $\xi \in [0, 1]$. Our approach is a generalization of hard and soft boundaries. When $\xi = 0$, this algorithm reduces to soft boundaries, and when $\xi = 1$, this corresponds to the hard boundaries of [86].

Annealing and Line Search

The EM algorithm, although it is guaranteed to converge, will often converge to a local optimum instead of the global optimum. One problem with running the gradient descent algorithm over

a GMM without annealing is that extremely planar surfaces will be described by nearly 2D Gaussians. In other words, the minimum eigenvalue of the covariance matrix will approach zero. In terms of surface reconstruction, this effect is desirable, since the surface will be very well-described by a degenerate Gaussian. In terms of gradient descent, however, a nearly 2D Gaussian will produce very steep peaks on the likelihood surface that do not give very useful gradient information when the current iterated solution is far away. As a solution gets close to a peak, the gradient will be nearly zero up until when the solution is near the mean, where the gradient will be an extremely high value and possibly cause overshooting or instability.

We adopt an annealing function to smooth out local minima and steep peaks formed by planar Gaussians. In our case, we add a decaying exponential function that adds a multiple of the identity matrix to the covariance matrix for each cluster,

$$\mathbf{\Sigma}_j^* := \mathbf{\Sigma}_j + \lambda(T)\mathbf{I} \tag{4.9}$$

where, if T is our current iteration

$$\lambda(T) = \epsilon + k_1 e^{-T/k_2} \tag{4.10}$$

This provides a "fattening" or regularization of the mixtures so as to have shallower gradients in an attempt to not oscillate between local minima. In our experiments, we found a wide array of values for k_1, k_2 to be acceptable. In the following experiments, we use $k_1 = 0.01$ and $k_2 = 20$, with $\epsilon = 1e - 5$.

As an additional protection against taking steps that may cause oscillations or jumps to non-optimal solutions, we do a line search over our step size. Note that this search requires repeated computation of the log likelihood of a given rigid transformation parameter set, which can be done fully in parallel for each $(\mathbf{z}_i, \mathbf{\Theta}_j)$ tuple and then summed as a reduction.

4.3.2 Implementation and Algorithm

We refer to our gradient descent-based algorithm as REM-Seg, or Robust Expectation Maximization Segmentation.

At a high level, starting from a raw point cloud, the REM-Seg algorithm does the following:

E step: We measure the probability of each point having been generated by each cluster, given its current mean and covariance. We then assign the correspondence for each point as its expectation or as zero, according to the highest probability cluster and ξ. At the end of the E step, we have a new set of correspondence values for each point.

M step: We gather each point weighted by its expectation and calculate the most likely mean and covariance given all the points that were assigned to the cluster. At the end of the M step, we end up with a new set of parameters for the Gaussians.

Gradient step: We transform each point according to our current transformation guess and then calculate the annealed gradient with respect to the unit quaternion and translation vector representing the 6-DOF spatial transformation. We then line search to find the step size to increase the overall data log likelihood.

Data: Two sets of PCD to be registered: $\mathbf{Z}$, $\mathbf{Z_2}$
Result: $\mathbf{q}$, $\mathbf{t}$
Initialize $\mathbf{\Theta}_j = \{\pi_j, \mu_j, \mathbf{\Sigma}_j\} = \{1/J, \mathbf{r}, I\} \forall j \in \{1, ..., J\}$, where $\mathbf{r}$ is a random vector inside of the unit cube.;
while *not converged* **do**
    Calculate correspondences $\gamma_j(\mathbf{z}_i) \forall \mathbf{z}_i \in \mathbf{Z}$ (in parallel);
    Apply robust thresholding to $\gamma_j(\mathbf{z}_i)$ (in parallel);
    Update $\mathbf{\Theta}$ (parallel reduction);
    Drop unsupported $\mathbf{\Theta}_j$;
end
Initialize $\mathbf{q} = (0, 0, 0, 1)^T$ and $\mathbf{t} = (0, 0, 0)^T$;
while *not converged* **do**
    Apply annealing step to $\Sigma_j, \forall j = \{1, .., J\}$;
    Calculate ∇_q, ∇_t given $\mathbf{Z_2}$ and current $\mathbf{q}, \mathbf{t}$ (in parallel);
    Perform line search to find α (parallel log-likelihood calculations);
    $q := q + \alpha \nabla_q$ (parallel reduction);
    $t := t + \alpha \nabla_t$ (parallel reduction);
end

Figure 4.2: Pseudocode for REM-Seg.

The EM algorithm therefore achieves our goal of "mixture decoupling" and allows us to operate over a descriptive yet very compact model for the gradient step. See Figure 4.2 for pseudocode.

4.3.3 Experimental Design and Metrics Used

We will briefly discuss our experimental design for our registration experiments in this and subsequent chapters.

It can be difficult to properly convey an algorithm's registration accuracy through a single number. There are several reasons for this:

1. When algorithms diverge, they may produce high levels of error that are not well-captured by a single average. Thus, a highly accurate algorithm that occasionally diverges may look "better" through average error than an inaccurate method that never completely diverges.
2. Rotation causes non-linear point displacement as one deviates from the coordinate system origin. Thus, the shape of the point cloud can have non-linear effects on how error is distributed.
3. The scale of point clouds for different types of environments and sensors may be radically different. Thus, error metrics with different scales of datasets will often not be comparable.
4. Translation is linear while rotation is non-linear, and so there is no clear way to fairly combine these errors into one metric.

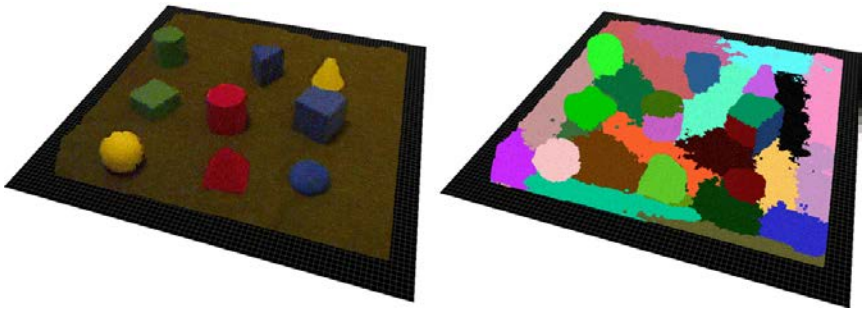


Figure 4.3: Example surface patches with each point colored by its most likely cluster. Note that since the GMM formulation is additive with respect to each cluster, over-segmenting serves to more accurately construct a PDF of scene.

5. Given different registration applications, we may care about different error metrics. For example, for a visualization or mapping application, we may be more interested in geometric error. For state estimation, we might be most interested in pose error.

To address these problems, we use several tactics:

1. If actual point-pairs exist (as in the case of some of our synthetic tests), we use the average squared Euclidean error between point pairs since this intuitively captures both rotation and translation error.
2. When running multiple tests, we try to show a distribution of errors when possible instead of a single average. If we calculate an average error, we also show the standard deviation.
3. We resize all our experimental data to fit within the unit cube to mitigate non-linear rotational effects.
4. We typically focus on rotation error over translation error, since this is typically where the most interesting differences are exhibited among competing registration algorithms.
5. We focus on pose error metrics (instead of geometric error) given that most of our applications revolve around state estimation rather than visualization/mapping. However, in Chapter 6.1, which is explicitly about visualization, we utilize a geometric error metric (PSNR) for our comparative results.

Even given the strategies listed above, we acknowlege that these choices consitute a set of trade-offs. For example, we choose nonintuitive error metrics (e.g. Frobenius norm error) that can only weight error uniformly across the scene, even though in some applications this may hide important sources of error. However, the quantitative metrics used in thesis conform well to our qualitative results, and so we feel that these trade-offs are warranted.

4.3.4 Experiments

For the following experiments, we use two datasets: a cross-section of the Stanford Bunny [17] and a point cloud generated from a Kinect of a table-top scene with various geometric shapes on top. A depiction of the table-top scene is given by the left image in Figure 4.3. Both datasets are scaled to fit within the unit cube for the following experiments.

We compare our algorithm, REM-Seg, with ICP, EM-ICP [44] [75], Softassign [39], and GMM-Reg [58]. Additionally, we test our method against GPU-accelerated versions of EM-ICP and Softassign [136]. The CPU versions of EM-ICP and ICP are accelerated with OpenMP. The algorithms to which we compared REM-Seg are all freely available to download. In particular, our implementation of ICP was taken from work described in [136]. All experiments were done on a quad-core CPU (Intel Q6600) and Nvidia GPU (GTX 680).

Segmentation

The Kinect table-top scene is shown in Figure 4.3. The colorized point cloud is shown on the left and the segmentation is shown on the right. The full scene contains over 50,000 points. We color the points according to the maximal expected cluster membership. Note that the GMM algorithm represents an "over-segmentation" in that some homogenous surfaces are broken up into separate clusters. For purposes of segmentation for perception tasks, one could trivially join together like clusters. For instance, clusters representing the ground plane will all have a very similar normal, and so a normal space clustering could be used in conjunction with Euclidean distance to join them. For purposes of registration, over-segmentation does not have any adverse effects since the GMM model adds individual cluster probabilities. In fact, in the limit of one cluster per point, the REM-Seg algorithm reduces to an algorithm somewhat similar to EM-ICP [43].

We can see that the sample segmentation of Figure 4.3 has some nice properties compared with similar planar decompositions. When a facet is large enough, it is described by a mostly flat Gaussian. Examples of this effect can be seen in the cube shape and the top of the larger cylinder. However, when the shape does not have as many supporting points, it is described by a single 3D Gaussian, as shown by the smaller shapes, such as the sphere. Describing the sphere by a 3D Gaussian provides a much better gradient than a planar segmentation of the same object.

Registration Quality

To test the convergence or "goodness-of-fit" properties of the various algorithms, using the Stanford Bunny, we randomly and uniformly sampled 100 rigid transformations from the space of transformations bounded by rotations of less than 15 degrees around any single axis, and translations bounded by 2 times the extent of the dataset in each dimension. Each algorithm was given the same randomly transformed and subsampled cloud and run until convergence. The squared error (SE) was then calculated. Figure 4.4 shows the distribution of the SE for each algorithm. The x-axis is the SE and the y-axis represents the number of trials that fell into the particular bin. We also repeated these experiments with the Kinect table-top scene with similar results, so for space reasons we have omitted them.

We found that our convergence was much more robust to random rigid transformations than ICP, Softassign, and EM-ICP. Even with potentially large transformations, our algorithm found a solution close to zero SE, whereas the other algorithms diverged. ICP, especially, succumbed to many local optima that represented globally non-optimal solutions. Our method most closely resembles GMM-Reg in terms of convergence, though 15 of the 100 GMM-Reg trials diverged to a wildly wrong registration result, while Rem-Seg fell into local minima that were closer to zero. Figure 3 shows example cases for various squared errors. Note that Figure 3a and 3b represent

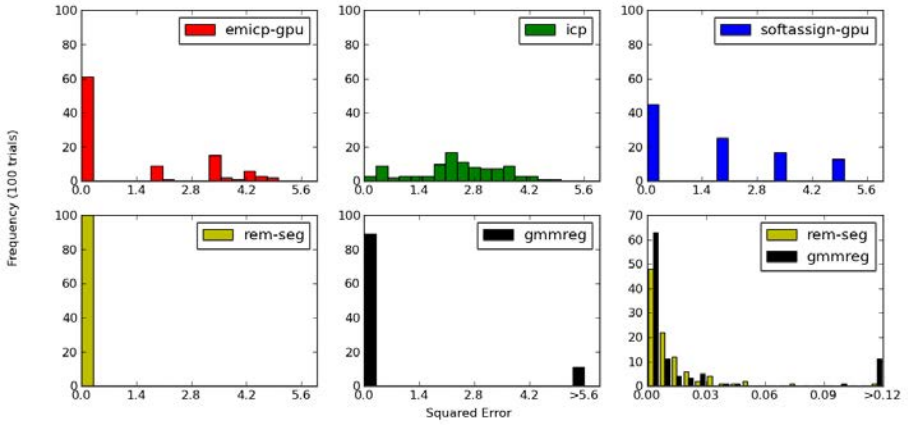
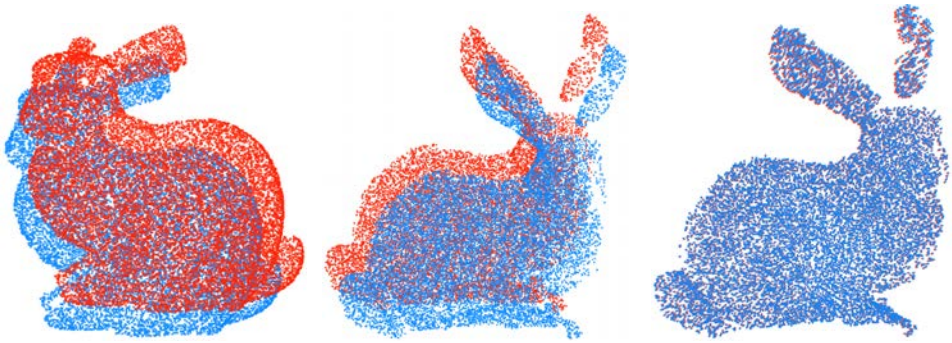


Figure 4.4: Goodness of fit using the Bunny dataset. These convergence results are from 100 random rigid transformations. REM-Seg consistently converges to near-zero error while the other algorithms tend to get stuck at local minima. The bottom right subfigure shows the two best algorithms with a zoomed x-axis.



(a) Sq. Error $\approx$ 0.16 (b) Sq. Error $\approx$ 0.13 (c) Sq. Error $\approx$ 7.5e-3

Figure 4.5: Example Registration Results

cases that are less than 0.20, compared to the local minima encountered by EM-ICP, ICP, and Softassign, which often were 3.0 or more. Figure 3c represents a correctly converged case.

Scaling and Speed

Using the Kinect tabletop dataset, we randomly subsampled from its points and ran our registration algorithm over varying amounts of points. For each data point, we averaged the time to convergence over 10 random rigid transformations of the scene.

The timing results can be seen in Figure 4.6. The figure on the top shows the y-axis as a log scale, giving the time to convergence. As the number of points increases, we can see that the scaling of all algorithms but EM-ICP and REM-Seg are poor. Note that regular ICP is the fastest of all algorithms when the number of points is very small, but the convergence is typically poorer. The bottom plot is the same data but on a linear y-axis to show more clearly the difference between the two best algorithms. We can clearly see that the scaling difference

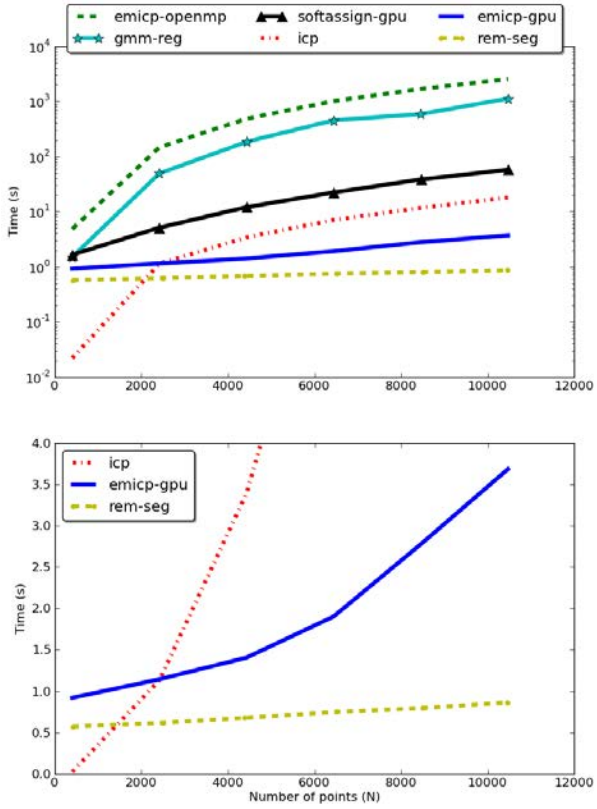


Figure 4.6: Timing results as the number of points in the cloud to be matched to the scene increases for the Kinect dataset. The top plot shows the time to convergence on a log scale, while the bottom plot shows the same data on a linear y-axis among ICP, EM-ICP-GPU, and REM-Seg.

grows superlinearly, favoring REM-Seg. Since the GPU-accelerated EM-ICP relies mainly on accelerated linear algebra routines, REM-Seg is faster since it presents a naturally data parallel solution for the GMM problem. Though not shown in the Figure, we can run REM-Seg over much larger point clouds, achieving sub-second registrations up to just under one million points. In general, the execution times scale roughly logarithmically until the parallelism limit is reached on the GPU, after which the scaling is mostly linear. With our current hardware, this limit is reached at around 100,000 points.

Though GMM-Reg was comparable to REM-Seg in terms of the goodness-of-fit of the final registration, the fact that it does not currently run on parallel hardware is a severe limitation when processing many points. Furthermore, the related methods rely on Gaussians around every point in the cloud, while REM-Seg efficiently compresses the clouds into a sparse amount of parameters.

4.4 MLMD and Expectation Maximization

In this Section, we frame both the decoupling and registration process inside a unified, two-mode Expectation Maximization (EM) framework, for which we derive a Maximum Likelihood

Estimation (MLE) solution along with a parallel implementation on the GPU. We evaluate our MLE-based mixture decoupling registration method (MLMD) over both synthetic and real data, showing better convergence for a wider range of initial conditions and higher speeds than previous state of the art methods.

Finally, we show how the dual-mode optimizations (mixture decoupling and registration) can be unified under a common EM framework and how exploiting this relation can both improve robustness and drive a highly parallel implementation on the GPU.

Compared to MLMD, REM-Seg from the previous section follows a similar preprocessing structure, using EM to derive a small set of anisotropic mixtures, but unlike MLMD, does not utilize the EM framework for registration, instead minimizing the cost function directly using gradient descent (similar to NDT [134], GMMReg [59], and KC [142]). Though MLMD is much more efficient than REM-Seg, we include both algorithms in this thesis as a demonstration of the different ways in which these data parallel generative models may be utilized for a given perception task.

4.4.1 Approach

The GMM/EM formulation allows one to recast the correspondence problem of ICP into probabilistic terms, opening the door to soft correspondences based on relative likelihoods, as well as maximum likelihood optimization procedures or minimum energy functions. By explicitly introducing latent correspondences, the EM framework factorizes the joint likelihood so that the log likelihood has no exponentials in it at all, thus greatly simplifying the optimization procedure over methods that do not operate under the EM framework [24, 57, 133, 142].

Our method builds on the GMM/EM concept, and through mixture decoupling, offers large reductions in computational expense for both representation and registration. Furthermore, as we later show in Section 4.4.6, the added descriptive power of using general anisotropic covariances, mean-point decoupling, and floating mixture weights adds robustness to pose displacements, noise, and offers a more compact description of point cloud geometry.

In this Section, we discuss how mixture decoupling and registration can be unified under a dual-mode EM framework. Then, we derive several accurate MLE approximation techniques for models with general anisotropic covariance parameters since the exact MLE equation with anisotropic covariances has no closed form solution. These approximations along with our technique of mixture decoupling form a highly efficient Generalized EM (GEM) procedure [18] for maximum likelihood point cloud registration.

4.4.2 Overview

In general, directly optimizing over the GMM parameters, $\mathbf{\Theta}$, and the registration parameters, T is intractable. To solve this problem we introduce latent correspondence variables, enabling the EM algorithm as a means to iteratively produce estimates of these parameters. The E Step finds the maximum likelihood estimate of points-to-cluster correspondences. The correspondences are then used to drive a tractable optimization procedure over the both the model and registration parameters in succession. We denote the M Step over GMM parameters $\mathbf{\Theta}$ as $M_{\mathbf{\Theta}}$ and the M

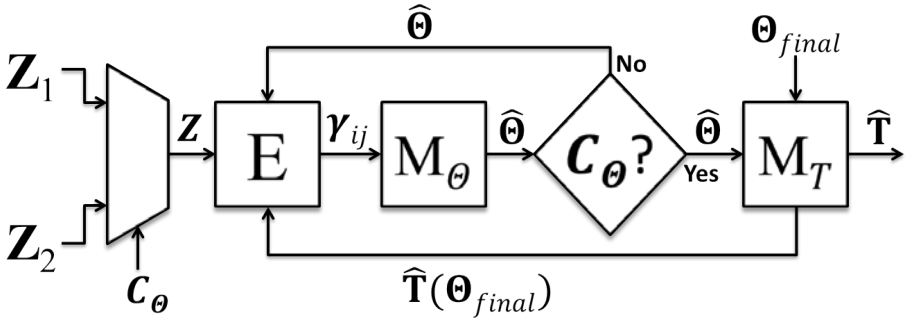


Figure 4.7: **Algorithm Flow** The algorithm first iterates over E and $M_{\mathbf{\Theta}}$ until convergence, a condition denoted by the binary variable $C_{\mathbf{\Theta}}$. Once $M_{\mathbf{\Theta}}$ converges under $\mathcal{Z}_1$, the algorithm then switches to the second point cloud for input (shown through the one-bit multiplexer). In order to produce an optimization criterion of size J, the $M_{\mathbf{\Theta}}$ Step result then feeds into the M_T Step along with the last $\hat{\mathbf{\Theta}}$ from $\mathcal{Z}_1$ as $\mathbf{\Theta}_{final}$. The algorithm then iterates accordingly, finding new transformation updates $\hat{T}$, which are applied to $\mathbf{\Theta}_{final}$.

Step over registration parameters T as M_T. Fig. 4.7 shows the diagrammatic depiction of the proposed algorithm.

The process of decoupling cluster means from the points is intimately related to the process of finding the pose transformation for registration. Fig. 4.7 shows our dual-step process: we perform the same E Step during both model optimization and registration, but utilize two different M Steps according to the state of convergence of the algorithm. We first iterate over $\hat{\mathbf{\Theta}}$ using $M_{\mathbf{\Theta}}$. Once this process has converged, we introduce the second point cloud, but now we iterate over jointly coupled M Step that solves an MLE approximation for the registration parameters. For 3D PCD, we have shown that $M_{\mathbf{\Theta}}$ can be solved efficiently in parallel and in closed form (Chapter 3). Since M_T is jointly coupled to $M_{\mathbf{\Theta}}$, as we will show, it too can be efficiently computed in parallel. Note that for additional computational efficiency, we transform $\mathbf{\Theta}$ and not $\mathcal{Z}$. Upon convergence, the concatenation of all $\hat{T}$ produce the correct relative transformation between the two point clouds, $\mathcal{Z}_1$ and $\mathcal{Z}_2$.

4.4.3 E and $M_{\mathbf{\Theta}}$ Steps

Our E and $M_{\mathbf{\Theta}}$ Steps for MLMD follow the formulation presented in Chapter 3, but with a special noise cluster for added robustness. For convenience, we will briefly review these steps below:

The first E and $M_{\mathbf{\Theta}}$ Steps iterate over a point cloud $\mathcal{Z} = \{\mathbf{z}_i\}, \forall i \in \{1..N\}$ in order to solve the maximum data likelihood problem over a set of Gaussian Mixture parameters, $\mathbf{\Theta} = \{\pi_j, \boldsymbol{\mu}_j, \mathbf{\Sigma}_j\}, \forall j \in \{1..J\}$. Since we cannot maximize this probability in closed form, we introduce latent binary correspondences $\mathcal{C} = \{c_{ij}\}$ for each point, cluster tuple $\{\mathbf{z}_i, \mathbf{\Theta}_j\}$.

In the *E Step*, we calculate in parallel the posterior for all $c_{ij} \in \mathcal{C}$ given $\mathbf{\Theta}$:

$$E[c_{ij}] = \frac{\pi_j \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j)}{\sum_{j'=1}^{J} \pi_{j'} p(\mathbf{z}_i|\mathbf{\Theta}_{j'}) + \frac{\pi_{J+1}}{\eta}} \tag{4.11}$$

where η and π_{J+1} control a special "noise cluster" to filter outliers.

In the $M_{\boldsymbol{\Theta}}$ *Step*, we maximize the expected log-likelihood with respect to $\boldsymbol{\Theta}$, using our current $E[c_{ij}] \stackrel{\text{def}}{=} \gamma_{ij}$:

$$\max_{\boldsymbol{\Theta}} \sum_{ij} \gamma_{ij} \{\ln \pi_j + \ln p(\mathbf{z}_i | \boldsymbol{\Theta}_j)\} \tag{4.12}$$

Given a fixed set of expectations, one can solve for the optimal parameters in parallel in closed form at iteration k:

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_i \gamma_{ij} \mathbf{z}_i}{\sum_i \gamma_{ij}} \tag{4.13}$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_i \gamma_{ij} \mathbf{z}_i \mathbf{z}_i^T}{\sum_i \gamma_{ij}} - \hat{\boldsymbol{\mu}}_j \hat{\boldsymbol{\mu}}_j^T \tag{4.14}$$

$$\hat{\pi}_j = \sum_i \frac{\gamma_{ij}}{N} \tag{4.15}$$

Thus, given a set number of mixtures, J, we can iterate these two steps until convergence, leaving us with a compact set of mixtures representing the original PCD, decoupled from the original points.

4.4.4 M_T Step

In this section we show how the relationship between $M_{\boldsymbol{\Theta}}$ and M_T can provide an efficient technique for recovering the registration parameters. As in the E-$M_{\boldsymbol{\Theta}}$ stage (Sec. 4.4.3), we introduce correspondences to allow the optimization problem to factorize. For shorthand, we denote our moving point cloud as $\tilde{\mathcal{Z}} \stackrel{\text{def}}{=} T(\mathcal{Z})$. The full joint probability is then,

$$\ln p(\tilde{\mathcal{Z}}, \mathcal{C} | \boldsymbol{\Theta}) = \sum_{i=1}^{N} \sum_{j=1}^{J} c_{ij} \{\ln \pi_j + \ln \mathcal{N}(\tilde{\mathbf{z}}_i | \boldsymbol{\Theta}_j)\} \tag{4.16}$$

We transform the problem into an iterative algorithm, alternating between finding probabilistic point-to-cluster correspondences of the new points to the old mixtures and then maximizing the expected likelihood,

$$\hat{T} = \operatorname*{argmax}_T E_{p(\mathcal{C}|\tilde{\mathcal{Z}}, \boldsymbol{\Theta})} [\ln p(\tilde{\mathcal{Z}}, \mathcal{C} | \boldsymbol{\Theta})] \tag{4.17}$$

$$= \operatorname*{argmax}_T \sum_{ij} \gamma_{ij} \{\ln \pi_j + \ln \mathcal{N}(\tilde{\mathbf{z}}_i | \boldsymbol{\Theta}_j)\} \tag{4.18}$$

$$= \arg\min_T \sum_{ij} \gamma_{ij} ||\tilde{\mathbf{z}}_i - \boldsymbol{\mu}_j||^2_{\boldsymbol{\Sigma}_j} \tag{4.19}$$

These equations show that the most likely estimate of T is the one that minimizes the weighted squared Mahalanobis distance between points and clusters, where the weights are determined by calculating the expectation of the correspondence with respect to the current guess $\hat{T}$.

Note that due to the double sum, this equation has NJ terms. However, we can reduce this problem to an optimization procedure of size J by deriving an explicit relation between $M_{\boldsymbol{\Theta}}$ and M_T.

To see how we can do this, we first make the simplification that $\Sigma_j = \mathrm{I}, \forall j \in \{1...J\}$. Now the Mahalanobis distance reduces to the L_2 norm,

$$\hat{T} = \arg\min_T \sum_{ij} \gamma_{ij} ||\tilde{\mathbf{z}}_i - \boldsymbol{\mu}_j||^2 \tag{4.20}$$

Note that in the $M_{\boldsymbol{\Theta}}$ step, the solution for the mixing parameters is $\pi_j^* = \frac{\sum_i \gamma_{ij}}{N}$, which we can rewrite as $\sum_i \gamma_{ij} = N\pi_j^*$. Similarly, in the $M_{\boldsymbol{\Theta}}$ step, the optimal mean is $\boldsymbol{\mu}_j^* = \frac{\sum_i \gamma_{ij}\mathbf{z}_i}{\sum_i \gamma_{ij}}$, which we can also rewrite as $\sum_i \gamma_{ij}\mathbf{z}_i = (\sum_i \gamma_{ij})\boldsymbol{\mu}_j^* = N\pi_j^*\boldsymbol{\mu}_j^*$. Finally, let $\tilde{\boldsymbol{\mu}}_j \stackrel{\text{def}}{=} T(\boldsymbol{\mu}_j^*)$. Given these equations, we can reduce the original quantity to be minimized by first expanding it and then completing the square for only those parts that are vary with T. We first expand,

$$\begin{aligned} &\sum_i \sum_j \gamma_{ij} ||\tilde{\mathbf{z}}_i - \boldsymbol{\mu}_j||^2 \\ &= \sum_j \{\sum_i \gamma_{ij} \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_i - 2\boldsymbol{\mu}_j^T \sum_i \gamma_{ij} \tilde{\mathbf{z}}_i + \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j \sum_i \gamma_{ij}\} \\ &= \sum_j \{\sum_i \gamma_{ij} \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_i - 2N\pi_j^* \boldsymbol{\mu}_j^T \tilde{\boldsymbol{\mu}}_j + N\pi_j^* \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j\} \end{aligned} \tag{4.21}$$

We can further break down the first term in order to complete the square,

$$\sum_i \gamma_{ij} \tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_i = \sum_i \gamma_{ij} ||\tilde{\mathbf{z}}_i - \tilde{\boldsymbol{\mu}}_j||^2 - N\pi_j^* \tilde{\boldsymbol{\mu}}_j^T \tilde{\boldsymbol{\mu}}_j \tag{4.22}$$

We then place this result back into Eq. 4.21:

$$\begin{aligned} &\sum_j \{\sum_i \gamma_{ij} (\tilde{\mathbf{z}}_i - \tilde{\boldsymbol{\mu}}_j)^T (\tilde{\mathbf{z}}_i - \tilde{\boldsymbol{\mu}}_j) + \\ &\qquad N\pi_j^* \tilde{\boldsymbol{\mu}}_j^T \tilde{\boldsymbol{\mu}}_j - 2N\pi_j^* \boldsymbol{\mu}_j^T \tilde{\boldsymbol{\mu}}_j + N\pi_j^* \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j\} \\ &= N \sum_j \{\frac{1}{N} \sum_i \gamma_{ij} (\tilde{\mathbf{z}}_i - \tilde{\boldsymbol{\mu}}_j)^T (\tilde{\mathbf{z}}_i - \tilde{\boldsymbol{\mu}}_j) + \\ &\qquad \pi_j^* ||\tilde{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j||^2\} \end{aligned} \tag{4.23}$$

The first term inside the summation over j represents the weighted average of the squared distance of each point to its (weighted) centroid, and is thus invariant under rigid transformations so it can be dropped. Dropping this term then transforms the problem into a weighted optimization criterion of size J.

$$\hat{T} = \arg\min_T \sum_j \pi_j^* ||\tilde{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j||^2 \tag{4.24}$$

In contrast with Eq. 4.20, we now have J virtualized point correspondences between the decoupled model means and the maximum likelihood means of the new points with respect to the model. Additionally, each pair is weighted by its expected relative contribution among all J mixtures. Finally, it should be stressed that the calculations of π_j^* and $\boldsymbol{\mu}_j^*$ are *exactly* the same as in

$M_{\boldsymbol{\Theta}}$, and this property allows us utilize the same optimized parallel computation for registration as for mixture decoupling.

Other GMM/EM methods have used similar transformations for reducing the problem size to a single sum over points to virtualized points [33, 43, 51]. These constructions can be viewed as transformations from a one-to-one point-based criterion into a many-to-one or point-to-model criterion. In contrast, our reduction can be viewed as a *many-to-many* or *model-to-model* optimization criterion.

4.4.5 Closed Form Approximations

If we restrict T to include only the set of $6DOF$ rigid transformations, parameterized by the set of all $R \in SO(3)$ and $t \in \mathbb{R}^3$, such that $\mathcal{Z}_1 = R\mathcal{Z}_2 + \mathbf{t}$, then the minimization of Eq. 4.24 can be solved by using a weighted version of Horn's method [52]:

$$a_j = \boldsymbol{\mu}_j - \frac{\sum_j \pi_j^* \boldsymbol{\mu}_j}{\sum_j \pi_j^*} \quad \text{, and} \quad b_j = \boldsymbol{\mu}_j^* - \frac{\sum_j \pi_j^* \boldsymbol{\mu}_j^*}{\sum_j \pi_j^*} \tag{4.25}$$

and

$$K = \sum_j \pi_j^* a_j b_j^T \tag{4.26}$$

where K is a 3x3 matrix containing all the elements required to construct a special 4x4 matrix, from which the SVD solution gives the optimal rotation as a unit quaternion. See [52] for details.

The optimal translation is then,

$$\hat{t} = \sum_j \pi_j^* (a_j - R b_j) \tag{4.27}$$

However, in the preceding derivation we omitted the covariances by setting $\Sigma_j = \mathrm{I}, \forall j \in \{1...J\}$. If we want to retain a closed form solution in M_T but do not want to force our GMM in E and $M_{\boldsymbol{\Theta}}$ to have isotropic covariances, one simple approximation strategy is to use the "closest" isotropic *inverse* covariance in a Frobenius sense for the M_T Step. Note that we are still letting the covariance be fully anisotropic for both the E Step and the $M_{\boldsymbol{\Theta}}$ Step. We can solve this trivial approximation for M_T by inspection,

$$\hat{\sigma}_j = \arg\min_{\sigma_j} ||\boldsymbol{\Sigma}_j^{-1} - \sigma_j^2 I||_F = \sqrt{\frac{\mathrm{Tr}(\boldsymbol{\Sigma}_j^{-1})}{d}} \tag{4.28}$$

Thus, the new M_T criterion is,

$$\sum_j \pi_j^* \hat{\sigma}_j^2 ||\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j||^2 \tag{4.29}$$

We denote the collection of $\hat{\sigma}_j$ as *shape weights*. To differentiate the shape weighting from isotropic weighting, we define M_T*-points* and M_T*-shape*. M_T*-points* is the M step that does not have shape weights, and the M_T*-shape* is the one with shape weights. Note that the shape weights correspond to the average inverse eigenvalue for each covariance, thus scaling point

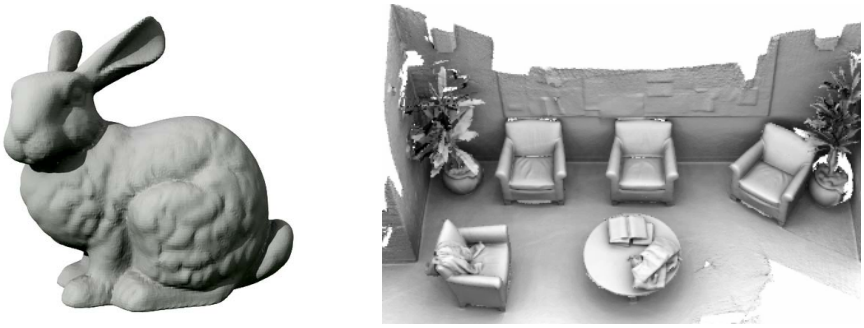


Figure 4.8: **Datasets** *Left:* Bunny [143], *Right:* Lounge [156]

distances inversely by the average "spread" of the cluster covariance. This formulation marks a significant departure to previous works that use just one sigma for all J clusters. We use full anisotropy in both the E steps for segmentation and registration, as well as for the $M_{\boldsymbol{\Theta}}$ step. The only step that does not use full anisotropy is the M_T step, where it is approximated.

4.4.6 Results

For our experimental results, we use both synthetic data (Stanford Bunny [143]) and real-world data from the Stanford Scene Datasets [156] (Fig. 5.4). The bunny contains around 40k points, while the lounge data is taken from a Kinect, and thus consists of 640x480 depth frames taken at 30 Hz. Our hardware testbed is a desktop computer with an i5-3500 CPU and GTX660 GPU.

We compare our method versus other popular publicly available open-source methods. For fairness, we use other parallelized registration methods when available, including optimized multicore implementations of ICP and EM-ICP (using OpenMP), and GPU-accelerated versions of Softassign and EM-ICP [136]. We also test using GMMReg [59], a popular open source package.

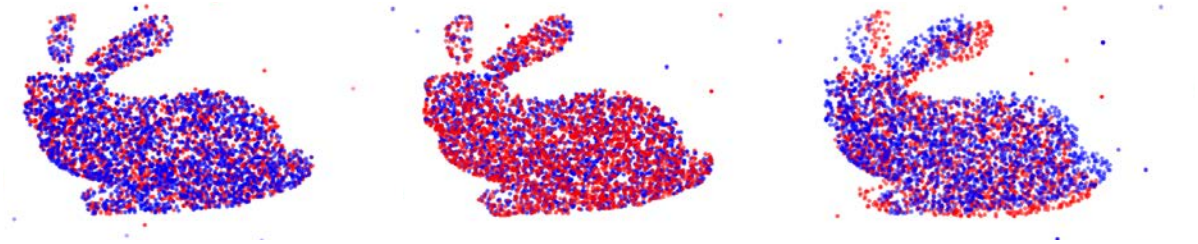
Robustness to Large/Random Rigid Transformations and Noise

To obtain a comparative measure of the convergence properties between our proposed method and the state of the art, we ran each registration algorithm 100 times while varying the amount of rotation (overlap). Using two differently subsampled cross-sections of the Stanford bunny (N=2000), we applied pitch axis transformations from -180 to 180 degrees. Additionally, we added noise in the form of 500 outliers sampled uniformly inside a region twice the extent of the original data. Fig. 4.10 shows the results. We used the Frobenius norm error between the recovered and real rotation matrices as our measure of rotational error. Both SoftAssign and our method were the only two algorithms that did not diverge over the entire range of pitch transformations, though SoftAssign's error is an order of magnitude higher than our method. As expected, ICP produced the worst convergence, recovering angles of < 45 degrees only. EM-ICP and GMMReg produced high quality results when the angle was less than 90 degrees (about double the range of ICP), but diverged to a local minimum for larger transformations.

To obtain a more general result of robustness to transformations on all axes, we randomly sampled 100 6DOF transformations and applied them to the Stanford Bunny. To sample from

Method	Recall (≤ 0.01)	Recall (≤ 0.025)	Avg. Time (s)	Std. Time (s)
ICP	0.00	0.00	0.196	0.050
SoftAssign	0.00	0.00	1.832	0.203
GMMReg	0.36	0.91	1.819	1.020
EM-ICP (CPU)	0.14	0.72	1.953	0.353
EM-ICP (GPU)	0.14	0.72	1.136	0.912
M_T-points	**0.53**	**0.92**	**0.107**	0.034
M_T-shape	**0.61**	**0.92**	**0.121**	0.037

Table 4.2: **Accuracy of random rigid transformations** 100 random 6DOF transformations were applied to a downsampled Stanford bunny corrupted with 5% outliers. The random rotations were bounded by 90 degrees in absolute sum over each axis angle, and the random translations were bounded by the extent of the dataset on each axis. Given two different error levels, the percentage of "successful" registration results is shown, where the error is defined by the Frobenius norm distance of the rotation matrix. To see a qualitative representation of Frobenius norm error, refer to Figure 4.9.



(a) Error = 0.005 (b) Error = 0.016 (c) Error = 0.268

Figure 4.9: **Registration quality for varying Frobenius norm errors** Three sample results from the random transformation test detailed in Section 4.4.6 and Table 4.2. The results depicted are the final converged results between both point clouds (red and blue points). (a) shows an error level that falls within both chosen bounds as given by Table 4.2. (b) shows a level of registration error that is within the 0.025 error threshold but not 0.01. (c) shows a high level of Frobenius norm error that is outside of both the chosen bounds.

the space of rotations in $SO(3)$ uniformly and without bias, we used the method outlined in [69]. The random rotations were bounded by 90 degrees in absolute sum over each axis angle, and the random translations were bounded by the extent of the dataset on each axis. As before, the bunny for both model and scene were randomly subsampled without replacement down to 2000 points and corrupted with 5% noise. In order to derive a simple accuracy measure, we looked at the percentage of solutions (% recall) for which the Frobenius error between the real and calculated rotation was acceptably small. To see a qualitative representation of what Frobenius norm error looks like at various levels under this test, refer to Figure 4.9. For our method, we set $J = 16$. The quantitative results are summarized in Table 4.2, along with average execution times.

Due to the random subsampling and noise, ICP and SoftAssign could not obtain accurate solutions over the random transformations tested. The GPU optimized EM-ICP was the fastest previous method tested, though ours is roughly an order of magnitude faster. GMMReg produced

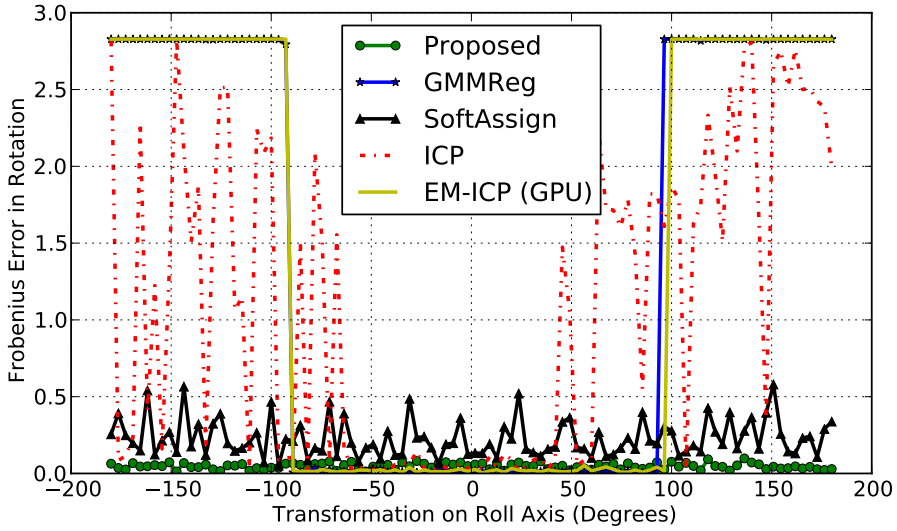


Figure 4.10: **Robustness to Large Transformations** Pitch axis transformations from -180 to 180 degrees were performed over randomly sampled Stanford Bunnies (N=2000). ICP has the smallest convergence radius, handling transformations of -45 to 45 degrees. EM-ICP and GMMReg approximately double the convergence region. Due to sampling nonuniformity, SoftAssign often produced poor registration accuracy though its worst-case solution was the best of all algorithms besides our proposed algorithm.

good results at an error threshold of 0.025, with a recall accuracy of 91%, but generated about a third of the solutions with error under 0.01. In contrast, our methods fared better, recovering over half of the transformations at the 0.01 error level, for both point-based and shape-based approximations. The M_T-shape method is about 10% slower in our current GPU implementation than M_T-points due to the need to calculate the shape parameters.

Fig. 4.11 shows how the proposed approach can handle the point cloud data with additional noises. Notice that our approach outperforms the other methods that can handles noisy inputs.

4.4.7 Scalability

Many modern sensors generate many thousands to millions of 3D points per second. In general, the registration methods reviewed in this paper greatly benefit from the addition of extra data in terms of accuracy and robustness. Unfortunately, the construction of point-based GMMs provide scaling challenges. When scaling to large numbers of points, even methods that have been optimized to be parallel [136] fail to approach real-time speeds for large point clouds. See Fig. 4.12 for details. We applied the same random transformations as in Section 4.4.6, but increased the amount of points sampled from the Stanford Bunny on every iteration. Due to our process of mixture decoupling inside a unified parallelized EM framework, Fig. 4.12 demonstrates orders of magnitude improvement on large point clouds.

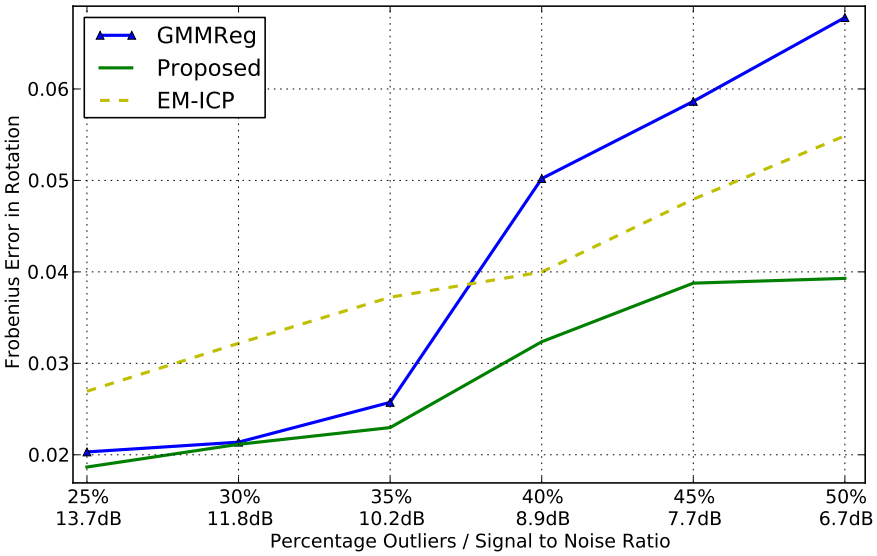


Figure 4.11: **Robustness to Noise** We injected both outliers and point-level Gaussian noise into two subsampled Bunny point clouds (N=2000) separated by 15 degrees on all roll, pitch, and yaw axes. Each iteration included both more outliers (shown as a percentage on the x-axis) and more point noise (shown as a Signal-to-Noise dB value). We did not include ICP or SoftAssign since any amount of noise and outliers tested corrupted the result significantly.

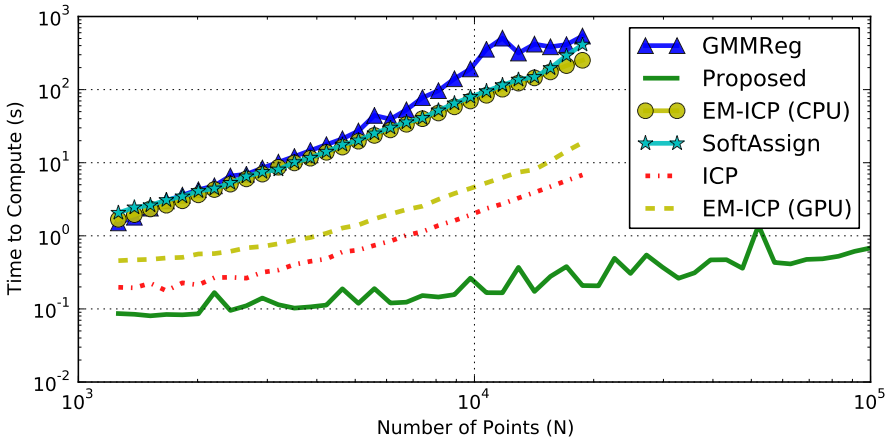


Figure 4.12: **Computation Times** Scaling the number of points subsampled from each point cloud for registration. Note that the scale is logarithmic on both x and y axes.

Real-World Data

To test our algorithm over real-world data, we chose to use the Lounge scene from the Stanford Scene dataset (Fig 5.4). For ground truth, we used the authors' pose calculations. Below is a table summarizing the frame-to-frame rotation error:

We subsampled down to 2000 points for each frame to produce tractable running times. ICP fared somewhat better on this real data since the rigid transformations remain small due to the

Method	Avg. Err	Std Dev
EM-ICP	0.890	1.109
SoftAssign	0.413	0.753
ICP	0.032	0.110
GMMReg	0.022	0.025
M_T-shape (N=2k)	**0.017**	**0.018**
M_T-shape (N=10k)	**0.010**	**0.005**

Table 4.3: Frame-to-frame rotation error over a stream of Kinect frames from the Stanford Scene dataset (Lounge)

fast framerate of the Kinect and the small amount of interframe motion. Suprisingly, EM-ICP often diverged, which we attribute to a failure of the multilevel registration process. We ran two versions of our algorithm, one using the same number of points as the others (2k), and the other using a much larger subsampling (10k). As shown in Section 4.4.7, the parallel nature of our algorithm along with the process of mixture decoupling allows us to incorporate much more data while still keeping running times better than the state of the art. Thus, the 10k subsampled PCD produces a better model and drives down the overall average frame to frame registration error significantly.

4.5 Mixture Decoupled Mahalanobis Estimation

In this section, we will derive another EM-based algorithm for finding the optimal transformation T between a point set $\mathcal{Z}_2$ and an arbitrary GMM $\hat{\Theta}_{\mathcal{Z}_1}$ that statistically describes the point set $\mathcal{Z}_1$. We produce $\hat{\Theta}_{\mathcal{Z}_1}$ from $\mathcal{Z}_1$ on the GPU using the same method described in Section 4.4.3.

As with MLMD (Section 4.4.1), given N points $\mathbf{z}_i$ and J clusters $\mathbf{\Theta}_j$, we introduce a NxJ set of point-cluster correspondences $\mathcal{C} = \{c_{ij}\}$, so that the full joint probability becomes

$$\ln p(T(\mathcal{Z}), \mathcal{C}|\mathbf{\Theta}) = \sum_{i=1}^{N}\sum_{j=1}^{J} c_{ij}\{\ln \pi_j + \ln \mathcal{N}(T(\mathbf{z}_i)|\mathbf{\Theta}_j)\} \tag{4.30}$$

We iterate between E and M Steps. On the E Step, we calculate $\gamma_{ij} = E[c_{ij}]$ under the current posterior. On the M Step, we maximize the expected data log likelihood with respect to T while keeping all γ_{ij} fixed,

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, E_{p(\mathcal{C}|T(\mathcal{Z}),\mathbf{\Theta})}[\ln p(T(\mathcal{Z}), \mathcal{C}|\mathbf{\Theta})] \tag{4.31}$$

$$= \underset{T}{\operatorname{argmax}} \sum_{ij} \gamma_{ij}\{\ln \pi_j + \ln \mathcal{N}(T(\mathbf{z}_i)|\mathbf{\Theta}_j)\} \tag{4.32}$$

$$= \arg\min_{T} \sum_{ij} \gamma_{ij}(T(\mathbf{z}_i) - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(T(\mathbf{z}_i) - \boldsymbol{\mu}_j) \tag{4.33}$$

$$= \arg\min_{T} \sum_{ij} \gamma_{ij}||T(\mathbf{z}_i) - \boldsymbol{\mu}_j||^2_{\boldsymbol{\Sigma}_j} \tag{4.34}$$

Thus, as before, the most likely transformation T between the point sets is the one that minimizes the weighted sum of squared Mahalanobis distances between points of $\mathcal{Z}_2$ and individual clusters of $\mathbf{\Theta}_{\mathcal{Z}_1}$, with weights determined by calculating expected correspondences given the current best guess for $\hat{T}$.

As we showed in Section 4.4.4, if we restrict T solely to the set of all rigid transformations ($T = \{R \in SO(3), t_{3x1}\}$) we can further reduce the double sum over both points and clusters into a single sum over clusters. This leaves us with a simplified MLE optimization criterion,

$$\hat{T} = \arg\min_{T} \sum_{j} \pi_j^* (T(\boldsymbol{\mu}_j^*) - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (T(\boldsymbol{\mu}_j^*) - \boldsymbol{\mu}_j) \tag{4.35}$$

where, as in $M_{\mathbf{\Theta}}$, $\pi_j^* = \frac{\sum_i \gamma_{ij}}{N}$ and $\boldsymbol{\mu}_j^* = \frac{\sum_i \gamma_{ij} \mathbf{z}_i}{\sum_i \gamma_{ij}}$

In the case of GMM-based registration, each affine transformation is determined by the covariance, or shape, of the cluster to which points are being registered. For example, clusters that are mostly planar in shape (two similar eigenvalues and one near zero) will tend to aggressively pull points toward it along its normal direction while permitting free movement in the plane. This observation should match one's intuition: given that we have chosen a probabilistic model that accurately estimates local geometry, a MLE framework will utilize this information to pull like geometry together as a type of probabilistic shape matching.

By using generalized anisotropic covariances, complex probabilistic point-to-geometry interactions can be well-handled. Intuitively, optimization of Eq. 4.35 should produce highly sensitive and robust registration performance. Unfortunately, however, previous algorithms in the literature have yet to fully leverage this general MLE construction. Simplifications are made either by 1) placing *a priori* restrictions on the complexity of the Gaussian covariance structure (e.g. isotropic only [33] or a single global bandwidth term [43]), or by 2) using approximations to the MLE criterion that remove or degrade this information [21]. The reasons behind both model simplification and MLE approximation are the same: Eq. 4.35 has no closed form solution. However, we will show how simply reinterpreting the Mahalanobis distance calculation can lead to a highly accurate, novel method for registration.

To arrive at our solution, we first need to re-interpret the Mahalanobis distance as a generalization of point-to-point distance where the coordinate system has undergone some affine transformation. To see this, one can use the Cholesky decomposition of the inverse covariance, for example, to reduce the Mahalanobis distance into a Euclidean distance where the points have undergone an affine transformation,

$$\begin{aligned} ||\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j||^2_{\boldsymbol{\Sigma}_j} &= (\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j) && (4.36) \\ &= (\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j)^T \mathbf{U}_j^T \mathbf{U}_j (\boldsymbol{\mu}_j - \tilde{\boldsymbol{\mu}}_j) && (4.37) \\ &= \|\mathbf{U}_j \boldsymbol{\mu}_j - \mathbf{U}_j \tilde{\boldsymbol{\mu}}_j\|^2 && (4.38) \end{aligned}$$

Unfortunately, we cannot use this reduction directly due to the non-commutative nature of matrix multiplication. We can, however, use this as insight into how the Mahalanobis distance may be approximated. Note that in most cases, range data is produced through rays intersecting with solid surfaces. Due to this phenomenon, range data can never "see inside" objects, and thus 3D point cloud data represents a sampling of a set of 2D manifolds in 3D space.

Since the manifolds are 2D, given a fine enough resolution, they will be locally planar. As noted in [130], we can see how this is true intuitively given that real-world surfaces are piece-wise differentiable. Thus, at a sufficiently fine resolution, the Mahalanobis distance will be dominated by the distance along the axis of least variance.

We first rewrite the inner Mahalanobis distance inside the MLE criterion of Eq. 4.35 by decomposing each covariance $\boldsymbol{\Sigma}_j$ into its associated eigenvalues λ and eigenvectors n using PCA, thereby producing the following equivalence,

$$||T(\boldsymbol{\mu}_j^*) - \boldsymbol{\mu}_j||^2_{\boldsymbol{\Sigma}_j} = \sum_{l=1}^{3} \frac{1}{\lambda_l} (\boldsymbol{n}_l^T (T(\boldsymbol{\mu}_j^*) - \boldsymbol{\mu}_j))^2 \tag{4.39}$$

Thus, we can reinterpret each cluster's Mahalanobis distance term inside the MLE criterion as a weighted sum of three separate point-to-plane distances. The weights are inversely determined by the eigenvalues, with their associated eigenvectors constituting each plane's normal vector. Going back to the example of a nearly planar Gaussian, its covariance will have two large eigenvalues and one near-zero eigenvalue, with the property that the eigenvectors associated with the larger eigenvalues will lie in the plane and the eigenvector associated with the smallest eigenvalue will point in the direction of its normal vector. Since the weights are inversely related to the eigenvalues, we can easily see that the MLE criterion will mostly disregard any point-to-$\boldsymbol{\mu}_j$ distance inside its plane (that is, along the two dominant PCA axes) and instead disproportionately focus on minimizing out-of-plane distances by pulling nearby points along the normal to the plane.

With a GMM of size J, we can see that by plugging in this equivalence back into Eq. 4.35, we arrive at the following MLE criterion,

$$\hat{T} = \arg\min_{T} \sum_{j=1}^{J} \sum_{l=1}^{3} \frac{\pi_j^*}{\lambda_{jl}} (\boldsymbol{n}_{jl}^T (T(\boldsymbol{\mu}_j^*) - \boldsymbol{\mu}_j))^2 \tag{4.40}$$

where the set of $\boldsymbol{n}_{jl}, l = 1..3$ represent the 3 eigenvectors for the jth Gaussian (anisotropic) covariance, and λ_{jl} the associated eigenvalues.

We have transformed the optimization from the minimization of a weighted sum of J squared Mahalanobis distances to an equivalent minimization of a weighted sum of $3J$ squared point-to-plane distances. In doing so, we arrive at a form that be leveraged by any number of minimization techniques previously developed for point-to-plane ICP [11]. Note that unlike traditional point-to-plane methods, which usually involve the computationally difficult task of finding planar approximations over local neighborhoods at every point and sometimes also for multiple scales [72, 145], the normals in Eq. 4.40 are found through a very small number of 3x3 eigendecompositions (typically $J \leq 1000$ for even complex geometric models) over the model covariances.

We propose an approximate solution to Equation 4.40 based on the point-to-plane ICP optimization technique described by Low [80], which we adapt into a weighted form in order to optimize in a linear least squares sense. The only approximation required is a linearization of R using the small-angle assumption. In practice, this is a fair assumption to use since GMM-based registration methods are local and thus tend to diverge for large pose displacements anyway.

4.5.1 Mahalanobis Estimation Least Squares Derivation

To set up the weighted linear least squares solution to Equation 4.40, we first define three vectors of weights,

$$\mathbf{w_1} = \begin{bmatrix} \sqrt{\frac{\pi_1^*}{\lambda_{1_1}}} \\ \sqrt{\frac{\pi_2^*}{\lambda_{2_1}}} \\ \sqrt{\frac{\pi_3^*}{\lambda_{3_1}}} \\ \vdots \\ \sqrt{\frac{\pi_J^*}{\lambda_{J_1}}} \end{bmatrix}, \mathbf{w_2} = \begin{bmatrix} \sqrt{\frac{\pi_1^*}{\lambda_{1_2}}} \\ \sqrt{\frac{\pi_2^*}{\lambda_{2_2}}} \\ \sqrt{\frac{\pi_3^*}{\lambda_{3_2}}} \\ \vdots \\ \sqrt{\frac{\pi_J^*}{\lambda_{J_2}}} \end{bmatrix}, \mathbf{w_3} = \begin{bmatrix} \sqrt{\frac{\pi_1^*}{\lambda_{1_3}}} \\ \sqrt{\frac{\pi_2^*}{\lambda_{2_3}}} \\ \sqrt{\frac{\pi_3^*}{\lambda_{3_3}}} \\ \vdots \\ \sqrt{\frac{\pi_J^*}{\lambda_{J_3}}} \end{bmatrix} \tag{4.41}$$

Next we define the following three vectors,

$$\mathbf{b_1} = \begin{bmatrix} \boldsymbol{n}_{1_1}^T(\boldsymbol{\mu}_1^* - \boldsymbol{\mu}_1) \\ \boldsymbol{n}_{2_1}^T(\boldsymbol{\mu}_2^* - \boldsymbol{\mu}_2) \\ \boldsymbol{n}_{3_1}^T(\boldsymbol{\mu}_3^* - \boldsymbol{\mu}_3) \\ \vdots \\ \boldsymbol{n}_{J_1}^T(\boldsymbol{\mu}_J^* - \boldsymbol{\mu}_J) \end{bmatrix}, \mathbf{b_2} = \begin{bmatrix} \boldsymbol{n}_{1_2}^T(\boldsymbol{\mu}_1^* - \boldsymbol{\mu}_1) \\ \boldsymbol{n}_{2_2}^T(\boldsymbol{\mu}_2^* - \boldsymbol{\mu}_2) \\ \boldsymbol{n}_{3_2}^T(\boldsymbol{\mu}_3^* - \boldsymbol{\mu}_3) \\ \vdots \\ \boldsymbol{n}_{J_2}^T(\boldsymbol{\mu}_J^* - \boldsymbol{\mu}_J) \end{bmatrix}, \mathbf{b_3} = \begin{bmatrix} \boldsymbol{n}_{1_3}^T(\boldsymbol{\mu}_1^* - \boldsymbol{\mu}_1) \\ \boldsymbol{n}_{2_3}^T(\boldsymbol{\mu}_2^* - \boldsymbol{\mu}_2) \\ \boldsymbol{n}_{3_3}^T(\boldsymbol{\mu}_3^* - \boldsymbol{\mu}_3) \\ \vdots \\ \boldsymbol{n}_{J_3}^T(\boldsymbol{\mu}_J^* - \boldsymbol{\mu}_J) \end{bmatrix} \tag{4.42}$$

We denote the set of J eigenvectors associated with each covariance's lth eigenvalue as $\boldsymbol{n}_{\lambda_l}$. Note that $\odot$ is the elementwise product. We weight the normals and vertically stack $\boldsymbol{\mu}_j^*$ as follows,

$$\mathbf{n} = \begin{bmatrix} \begin{bmatrix} \mathbf{w_1} & \mathbf{w_1} & \mathbf{w_1} \end{bmatrix} \odot \boldsymbol{n}_{\lambda_1} \\ \begin{bmatrix} \mathbf{w_2} & \mathbf{w_2} & \mathbf{w_2} \end{bmatrix} \odot \boldsymbol{n}_{\lambda_2} \\ \begin{bmatrix} \mathbf{w_3} & \mathbf{w_3} & \mathbf{w_3} \end{bmatrix} \odot \boldsymbol{n}_{\lambda_3} \end{bmatrix}, \hat{\boldsymbol{\mu}}^* = \begin{bmatrix} \boldsymbol{\mu}_j^* \\ \boldsymbol{\mu}_j^* \\ \boldsymbol{\mu}_j^* \end{bmatrix} \tag{4.43}$$

If we use the small angle assumption to linearize rotation, we are left with the following overconstrained linear system, where $\{\alpha, \beta, \gamma\}$ are the rotation angles and $\{t_x, t_y, t_z\}$ are the optimal translation parameters.

$$\begin{bmatrix} \hat{\boldsymbol{\mu}}^* \times \mathbf{n} & \mathbf{n} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} \mathbf{w_1} \odot \mathbf{b_1} \\ \mathbf{w_2} \odot \mathbf{b_2} \\ \mathbf{w_3} \odot \mathbf{b_3} \end{bmatrix} \tag{4.44}$$

Equation 4.44 can then be solved using any standard linear least squares algorithm.

4.6 Summary

In this Chapter, we presented three different methods for utilizing the models of Chapter 3 to create novel, robust registration methods. For each algorithm (REM-Seg, MLMD, and MDME)

we first represent the input 3D PCD with a GMM, and then decouple the optimization of the model parameters and registration (pose transformation) parameters. This two-step optimization provides more efficient and accurate registration performance even over large rigid transformations with poor initialization, as well as in conditions of high noise. Through experimentation, we showed that the proposed approaches can provide a more efficient and scalable solutions compared to other state-of-the-art registration algorithms.

In the next chapter, we will augment the base modeling scheme to be hierarchical, providing more efficiency and flexibility over the simple models described in Chapter 3, and we will use this as our platform for the more advanced techniques described in Chapters 6 and 7. Specifically as it pertains to the registration algorithms detailed in this chapter, in Chapter 7 we combine MDME with an augmentation that incorporates known sensor properties (Expected Overlap Estimation) in order to successfully register data in challenging real-world scenarios where data may not fully overlap.

Chapter 5

Hierarchical Mixture Construction

The generative models introduced in Chapters 3 and 4 assume the number of mixtures can be statically set *a priori*. The number of mixtures directly controls the fidelity of the model produced; the more mixtures, J, the greater the model's ability to reconstruct the original data. Statically setting J works well in many situations: where extremely fine detail is not needed, where the scene is bounded, or when we roughly understand the scene's geometric complexity *a priori*.

Ideally, however, we would like to be able to remove J as a hand-tuned parameter and let the scene itself decide the appropriate amount of mixtures to use. One difficulty of this approach is that, in the normal case, we want to constrain J to be much smaller than N in order to preserve the good computational properties of the model. Unfortunately, the algorithms presented thus far scale linearly with J, putting a hard constraint, regardless of the application's demands, on the number of mixtures that can be used effectively. Furthermore, the same application may call for several levels of fidelity (and therefore a dynamic number of mixtures) depending on its intended function. This situation suggests some type of multi-scale or hierarchical approach is needed.

A multi-scale or hierarchical approach provides an economical way to provide dynamic levels of geometric complexity of different parts of the same scene. When a single number of mixtures J drives the segmentation and model building process, roughly all parts of the scene are modeled with the same fidelity due to the MLE process. It is often the case, however, that different parts of the scene demand different levels of modeling fidelity. This is especially true in cases where we may want to visualize the data, where complex geometric surfaces (e.g. chandeliers) may need a finer level of resolution than simpler entities (e.g. walls).

To achieve a model with dynamic resolution, we have developed a hierarchical recursive version of the "flat" model introduced in Chapter 3 that subdivides clusters of high complexity in a top-down fashion. The subdivision process allows the model to be able to capture arbitrary amounts of detail, dynamically adjusting its complexity according to the complexity of the scene. The final output then is a tree of overlapping Gaussian mixtures that can be traversed much like an Octree, and where every node is itself a valid Gaussian Mixture describing some part of the original data at some particular level of detail.

We call the construction technique for such a model *Expectation Sparsification* due to its ability to exploit the limited spatial extent of surfaces, thereby allowing logarithmic scaling with respect to J (instead of linear scaling in the case of the model described in Chapter 3).

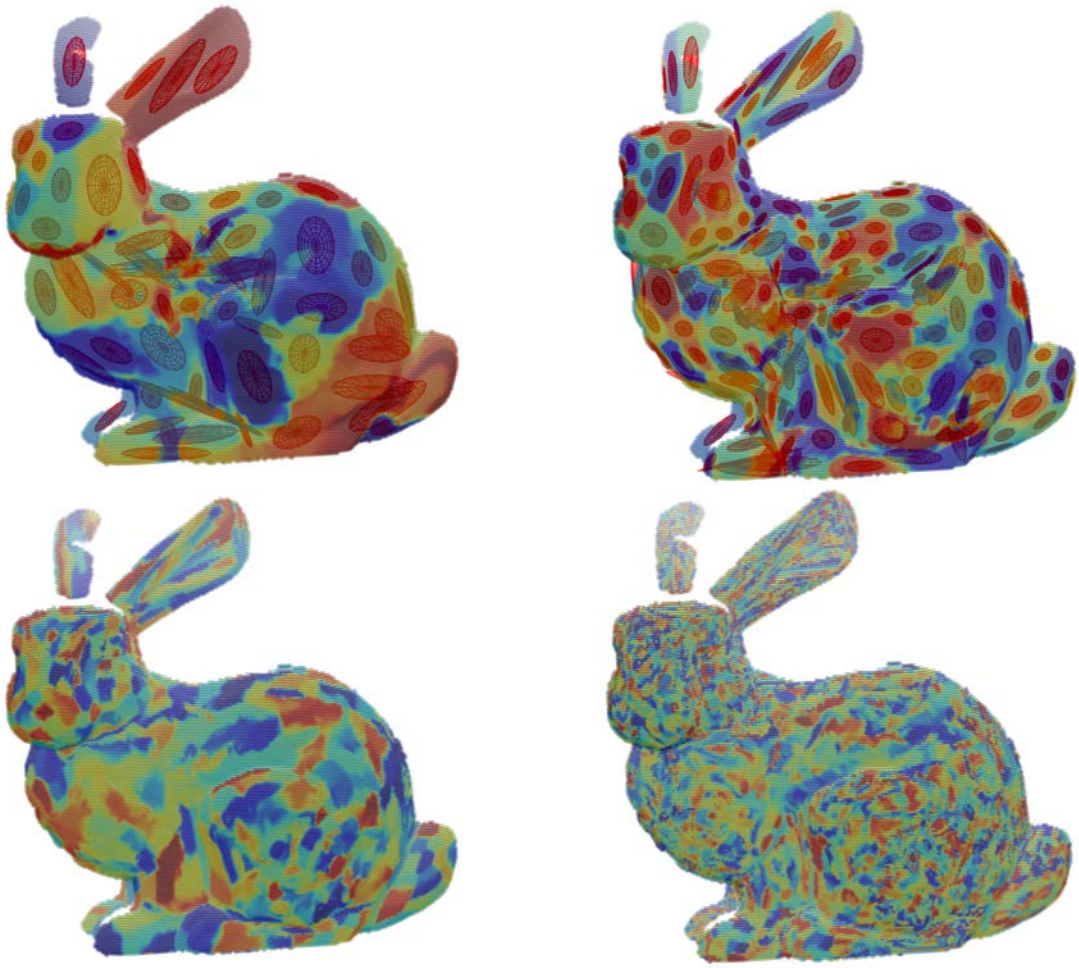


Figure 5.1: **Probabilistic Modeling using a Tree of Gaussian Mixtures:** Four Stanford bunny models reconstructed from different levels of detail in our GMM hierarchy. Each color denotes the area of support of a single Gaussian, and the ellipsoids indicate their 1 sigma extent. For better visualization, the ellipsoids are not drawn on the two rightmost figures to avoid cluttering due to the high level of detail.

5.1 Expectation Sparsification

Finding meaningful, structured representations of 3D point cloud data (PCD) has become a core task for spatial perception applications. In this chapter we introduce a method for constructing compact generative representations of PCD at multiple levels of detail. As opposed to deterministic structures such as voxel grids or octrees, we propose probabilistic subdivisions of the data through local mixture modeling, and show how these subdivisions can provide a maximum likelihood segmentation of the data. The final representation is hierarchical, compact, parametric, and statistically derived, and as we later show in Chapter 6, facilitates run-time occupancy calculations through stochastic sampling. Unlike traditional deterministic spatial subdivision methods, our technique enables dynamic creation of voxel grids according the application's best needs. In contrast to other generative models for PCD, we explicitly enforce sparsity among points and mixtures, a technique which we call expectation sparsification. This leads to a highly parallel hierarchical Expectation Maximization (EM) algorithm well-suited for parallel architectures and real-time execution. In this chapter, we explore the trade-offs between model fidelity and model size at various levels of detail, our tests showing favorable performance when compared to Octree and NDT-based methods.

5.2 Introduction

Given the recent commoditization of different types of active range sensors (*e.g.*, TOF, Lidar, structured light), spatial processing and visualization of large collections of 3D point clouds has become one of the most important stages in 3D imaging/vision pipelines [117]. 3D point cloud processing introduces several new challenging problems such as (1) uneven sampling density, (2) unstructured organization of the incoming data, (3) level-of-detail processing given varying speed and memory requirements, and (4) measurement uncertainty from sensor noise. Additionally, modern depth sensors generate millions of data points per second, making it difficult to utilize all incoming data effectively in real-time for devices with limited computational resources.

Many current techniques for processing large amounts of point cloud data (PCD) either simply subsample the data or apply some sort of discretization, either through dense, sparse [122] or hierarchical [30] voxelization techniques. Representing continuous geometry through voxels creates discretization artifacts and offers no clear way of handling noise or data uncertainty. Furthermore, the discrete nature of voxels and sub-sampled point clouds greatly complicate spatial processing procedures that require continuous derivatives or high quality normal estimates.

We address these challenges with a hierarchical and probabilistic representation of 3D point cloud data (PCD) in the form of a hierarchy of Gaussian Mixture Models (GMMs). As a representation of 3D space, a GMM model has several advantages. First, being a continuous probability density function (PDF), the GMM does not require the discretization of 3D space. Second, the uncertainties of data measurements are embedded in the covariance matrices of the GMM, which combined with a special cluster to handle outliers, provide an effective way of handling noisy measurements. Finally, the storage requirements for a GMM are much lower than for the original PCD.

Though GMMs have been used before for PCD representation [23, 32], we introduce a novel

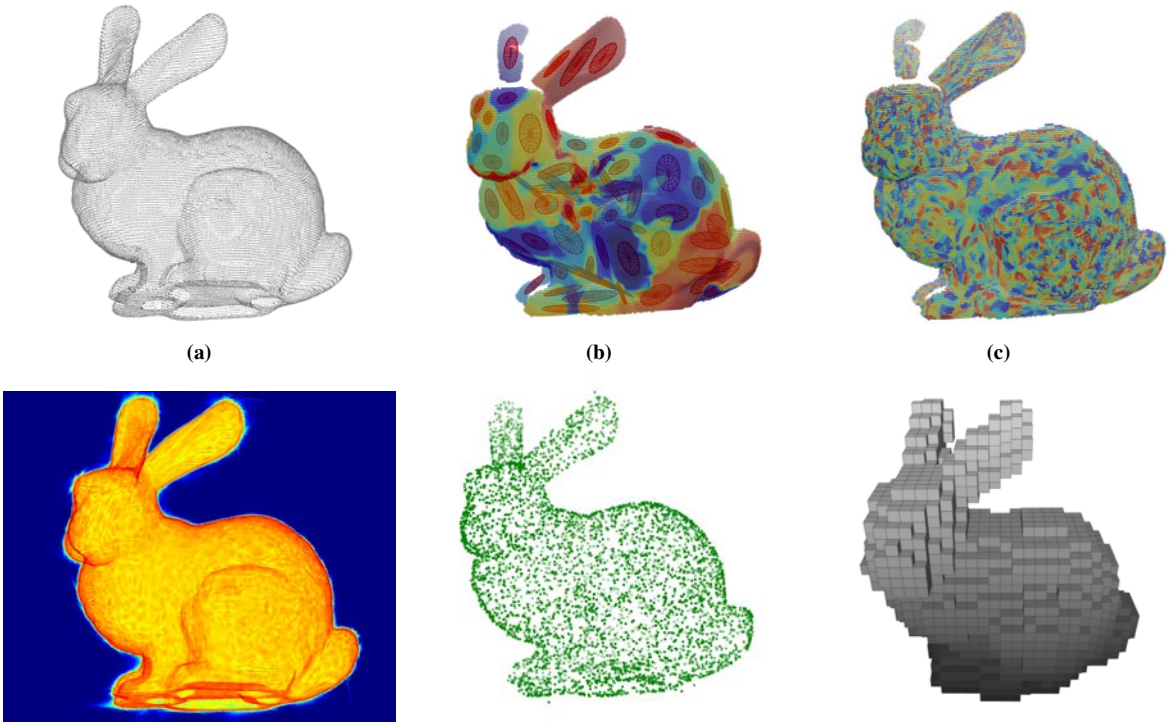


Figure 5.2: **Processing PCD with a Hierarchy of Gaussian Mixtures:** (a) Raw PCD from Stanford Bunny (*35k vertices*), (b) and (c) Two levels of detail extracted from the proposed model. Each color denotes the area of support of a single Gaussian and the ellipsoids indicate their one σ extent. Finer grained color patches therefore indicate higher statistical fidelity but larger model size, (d) a log-scale heat-map of a PDF from a high fidelity model. (e) stochastically re-sampled PCD from the model (*5k points*), (f) occupancy grid map also derived directly from the model. See Chapter 6 for technical details regarding (e) and (f).

Data Structure	Hierarchical	Generative	Voxel Free	Construction Complexity
Voxel Hash List [121]				N
Octree [53]	✓			$N \log N$
3D-NDT [5]		✓		N
3D-NDT-Octree [84]	✓	✓		$N \log N$
GMM [23]		✓	✓	NJ
Hierarchical GMM [41]	✓	✓	✓	N^2
Proposed Method	✓	✓	✓	$N \log J$

Table 5.1: **A Comparison of 3D Point Cloud Data Structures** *Hierarchical*: Hierarchical methods compress free space and are therefore more compact than dense grids. *Generative*: Generative models add parametric structure to PCD, facilitating statistical inference, maximum likelihood, or continuous optimization methods. *Voxel Free*: The lack of voxelization present in the model avoids discretization errors, allowing higher fidelity at smaller model sizes. *Construction complexity*: N is the number of points in the PCD, and J the number of mixtures in a GMM, with $J \ll N$ for most typical applications.

top-down hierarchical model which confers the following benefits: (1) dynamic allocation of the number of mixtures, with new clusters being added in areas of high-frequency detail, (2) efficient parallel coarse-to-fine construction by recursively partitioning the points in the PCD into their most influential mixture/s, and (3) multiple levels of detail for point cloud re-sampling and occupancy map creation.

In this chapter, we demonstrate a highly efficient and parallelizable method for hierarchical top-down GMM clustering that, as opposed to previous GMM-based techniques, applies sparse constraints on point to cluster assignments, thus enabling construction time logarithmic with respect to the overall model size.

5.3 Related Work

In most spatial processing applications that rely on point cloud data, using the raw points directly can be nearly intractable. Thus, most common operations one might want to perform: nearest neighbor queries, denoising, geometric or semantic inference, etc., stand to benefit from imposing some type of structure to the raw data. Table 5.1 summarizes typical data structures used for point cloud data.

Voxelization and occupancy grids [26, 138] have been established as a popular method to discretize raw PCD over a dense grid, but memory problems emerge when needing fine resolution or large grids. Especially in cases of 3D points, many voxels may be unoccupied, leading to inefficient memory usage. Octrees and kd-trees can be much more space efficient [53], as the construction of a regularly subdivided hierarchy effectively compresses empty space. These structures incur additional overhead compared to dense grids, however, requiring superlinear construction time with respect to the size of the PCD.

Whereas voxels and octrees rely on discretization to obtain structure from PCD, another class of algorithms instead model the data as a set of independent samples from some unknown distribution. These algorithms use the principle of maximum data likelihood to optimize a set of latent parameters that describe the original PCD. Known as generative models, these models can by construction provide robust probabilistic inference. Their trade-off, however, is the potentially high construction or inference cost and need for *a priori* knowledge.

For modeling 3D PCD, the most common generative model used in the literature is the Gaussian Mixture Model (GMM). Typically, GMMs are used to facilitate robust point cloud registration techniques [5, 21, 23, 32, 60]. The work of Jian and Vemuri [60], for example, convert a point cloud into a GMM by placing a covariance around each point. Though this minimizes the setup cost, inference then becomes very slow as the GMM is larger than the original raw points. In contrast to these methods, our proposed technique is exponentially faster with respect to the size of the model. A "flat" version must iterate linearly $O(J)$ through all J mixtures, whereas our method is $O(\log J)$. Furthermore, we do not need to specify the number of mixtures *a priori*. Our coarse-to-fine construction allows us to both filter out low-frequency details quickly (floors and walls) and drill down to deeper levels for high-frequency details.

The Normal Distributions Transform (NDT) [5, 123] is a widely used and elegant technique that attempts to merge the concepts of a voxel grid or octree with a GMM by simply recording the mean and covariance of all points that fall into each voxel. The GMM can then be constructed as a weighted sum of the voxel's respective Gaussian parameters. Though the construction of such a data structure is very efficient, the requirement to voxelize at the beginning can cause a loss of fidelity.

Other work has experimented with hierarchical forms of GMMs for applications like 2D image segmentation [36]. Typically, these methods operate bottom-up, repeatedly grouping together like clusters of points using divergence measures to split and merge the data. For example, Goldberger *et al* [41] construct an iterative EM-like algorithm using KL-Divergence in order to repeatedly merge candidate clusters. In contrast, we adopt a top-down hierarchical approach, motivated by the need to keep the calculation of point-mixture correspondences sparse for 3D point clustering. As such, our approach is more amenable to parallel hardware and is much more computationally efficient (see Table 5.1). Another similar top-down construction is that of Kalaiah *et al* [63], though this method is not generative.

5.4 Method Overview

Our model uses overlapping basis functions (anisotropic Gaussian mixtures) for representing 3D geometry. These functions are recursively applied in a top-down fashion to create a hierarchy of overlapping patches that approximate the original 3D PCD. The creation of this model is cast as the solution to a Maximum Likelihood Estimation (MLE) hierarchical Gaussian Mixture segmentation problem that can be solved by recursively employing the Expectation Maximization (EM) algorithm over increasingly smaller partitions of the point data.

5.4.1 Model Definition

As in Chapter 4, our world model is composed of J overlapping probabilistic mixtures $\mathbf{\Theta}_j$ plus a (J+1)$^{\text{th}}$ noise distribution. We choose our subsurface models to be weighted 3-dimensional multivariate Gaussians, $\mathbf{\Theta}_j = \{\pi_j, \boldsymbol{\mu}_j, \mathbf{\Sigma}_j\}$, and our noise distribution to be a uniform distribution over the bounding box of the point data. Together, these basis distributions produce a mixture model, which is itself a valid probability distribution.

Given a point cloud $\mathcal{Z}$ of size N, its probability of being generated by our model, given that each point is an *iid* sample of the world, is:

$$p(\mathcal{Z}|\mathbf{\Theta}) = \prod_{i=1}^{N} p(\mathbf{z}_i|\mathbf{\Theta}) = \prod_{i=1}^{N} \sum_{j=1}^{J+1} \pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j), \tag{5.1}$$

$$p(\mathbf{z}_i|\mathbf{\Theta}_j) = \begin{cases} \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j), & \text{for } 1 \leq j \leq J, \\ \frac{1}{\eta}, & \text{for } j = J+1, \end{cases} \tag{5.2}$$

where η is the size of the volume for which the noise cluster is active.

To find the basis functions to best fit the point cloud data we employ the EM algorithm [7], which has been established as a way to iteratively maximize data likelihood when there is no closed form solution to the maximizer, yet there is a way of finding a maximum of joint data likelihood of the data and a set of associated latent variables. We define a set C of latent variables c_{ij} that represents the binary associations between points $\mathbf{z}_i \in \mathcal{Z}$ and mixtures $\mathbf{\Theta}_j$. In the *E-Step*, we calculate the posterior for all $c_{ij} \in \mathcal{C}$ given $\mathbf{\Theta}$:

$$E[c_{ij}] = \frac{\pi_j p(\mathbf{z}_i|\mathbf{\Theta}_j)}{\sum_{j'=1}^{J+1} \pi_{j'} p(\mathbf{z}_i|\mathbf{\Theta}_{j'})} \tag{5.3}$$

In the *M-Step*, we maximize the expected log-likelihood with respect to $\mathbf{\Theta}$, using our current $E[c_{ij}] \stackrel{\text{def}}{=} \gamma_{ij}$:

$$\max_{\mathbf{\Theta}} \sum_{ij} \gamma_{ij} \{\ln \pi_j + \ln p(\mathbf{z}_i|\mathbf{\Theta}_j)\} \tag{5.4}$$

Given a fixed set of expectations, one can solve for the optimal parameters in closed form at iteration k:

$$\boldsymbol{\mu}_j^{k+1} = \frac{\sum_i \gamma_{ij} \mathbf{z}_i}{\sum_i \gamma_{ij}} \tag{5.5}$$

$$\mathbf{\Sigma}_j^{k+1} = \frac{\sum_i \gamma_{ij} \mathbf{z}_i \mathbf{z}_i^T}{\sum_i \gamma_{ij}} - \boldsymbol{\mu}_j^{k+1} \boldsymbol{\mu}_j^{k+1T} \tag{5.6}$$

$$\pi_j^{k+1} = \sum_i \frac{\gamma_{ij}}{N} \tag{5.7}$$

5.4.2 Expectation Sparsity

Given the above definitions and a sufficiently high number of mixtures J, the posterior over correspondences will be sparse due to the nature of 3D geometry. We can see this fact intuitively: Consider that in an indoor scene, for example, the geometric structure of a light fixture will not be statistically informative to a point sampled on a couch beneath it. Thus, given a point cloud of size N, if we naively try to calculate all NJ point-subsurface expectations (γ_{ij}), most will be zero or near-zero and not contribute meaningfully to the calculation. Therefore, one could save vast amounts of computation when trying to calculate γ_{ij} by restricting the summation to only those $\{\mathbf{z}_i, \mathbf{\Theta}_j\}$ tuples that are known to have sufficiently non-zero conditional probability. We show in the next section how to solve this problem by constructing a top-down hierarchy of GMMs.

5.4.3 A Top-Down Hierarchy of Mixtures

We can formally define our hierarchical Gaussian Mixture Model recursively by looking at the probabilistic form for a point $\mathbf{z}_i \in \mathbb{R}^3$. At the root of our tree, level 1 ($l = 1$), our model consists of a Gaussian Mixture of size $\hat{J}$, with a $\hat{J} + 1$th noise cluster:

$$p(\mathbf{z}_i|\mathbf{\Theta}^{l=1}) = \sum_{j=1}^{\hat{J}+1} \pi_j^{l=1} p(\mathbf{z}_i|\mathbf{\Theta}_j^{l=1}) \tag{5.8}$$

Each $p(\mathbf{z}_i|\mathbf{\Theta}_j^{l=1})$ can then be refined as another Gaussian Mixture:

$$p(\mathbf{z}_i|\mathbf{\Theta}_j^{l=1}) = \sum_{k=1}^{\hat{J}+1} \pi_k^{l=2|1} p(\mathbf{z}_i|\mathbf{\Theta}_k^{l=2|1}), \tag{5.9}$$

where the superscript indicates the selection of Gaussian parameters at level 2 given the parent node at level 1. The above is a proper Gaussian Mixture that satisfies $\sum_{k=1}^{\hat{J}+1} \pi_k^{l=2|1} = 1$. Our model is then fully defined by the set of all $\mathbf{\Theta}_k^l$ and π_k^l.

If we begin with a coarse decomposition into $\hat{J} \ll J$ mixtures, after convergence, the posterior over correspondences gives us a natural maximum likelihood partitioning of data into $\hat{J}$ coherent geometric regions. We can then use this posterior as a partitioning function over our data, and reduce our problem into $\hat{J}$ subproblems of roughly $1/\hat{J}$th the size. Recursing this process multiple times generates a tree of GMMs, requiring many small EM algorithms of size $\hat{J}$. The number of levels in the hierarchy would be $l = \log_{\hat{J}}(J)$, where each level produces $\hat{J}^{l-1}$ EM problems of size $\hat{J}$. Thus, we would need $O(\frac{J-1}{\hat{J}-1})$ EM algorithms of size $O(N_l\hat{J})$, where $N_l \approx \frac{N}{J^{*l-1}}$. The entire procedure will be logarithmic in the number of mixtures and linear in the number of points, $O(N \log_{\hat{J}}(J))$.

In order to maintain a valid global GMM, however, we need to share context between parents and children. Mathematically, we can derive this relation by assigning causal relationships to a set of l latent correspondence variables, $\mathcal{C}^l$, as depicted in Fig. 5.3. Using the model, we can

calculate the probability of our observed variable by marginalizing over the latent variables. In the two layer case,

$$\begin{aligned} p(\mathbf{z}_i|\mathbf{\Theta}^{l=2}) &= \sum_{\mathcal{C}^{l=1}} \sum_{\mathcal{C}^{l=2}} p(\mathbf{z}_i, c_i^{l=1}, c_i^{l=2}|\mathbf{\Theta}^{l=2}) \\ &= \sum_{\mathcal{C}^{l=1}} \sum_{\mathcal{C}^{l=2}} p(\mathbf{z}_i|\mathbf{\Theta}^{l=2}, c_i^{l=2}) p(c_i^{l=2}|c_i^{l=1}) p(c_i^{l=1}) \\ &= \sum_{j'=1}^{\hat{J}+1} \sum_{j=1}^{\hat{J}+1} \pi_{j'}^{l=1} \pi_j^{l=2|1} p(\mathbf{z}_i|\mathbf{\Theta}_j^{l=2|1}) \end{aligned} \tag{5.10}$$

We can clearly see that for multiple levels, the correct mixing value must be propagated down the tree to the leaf node, forming a multiplicative chain.

5.4.4 Sparsification: Hard and Soft Partitioning

When we recurse into a new tree level, we utilize the set of posteriors γ_{ij} of the parent level to obtain a partition of our PCD. We call this process *expectation sparsification*. One possible partitioning strategy is to simply assign a point to the mixture for which its parent expectation was the highest. We will refer to this as *hard partitioning*. However, though this method retains mixture overlap inside every group of $\hat{J}$ children, we will have no such overlap among groups of children from different parents.

We can use a *soft partitioning* scheme to constrain the amount of geometric context sharing among children of different parents while still maintaining logarithmic efficiency with respect to the number of mixtures. To do this, we introduce a parameter, λ_p, that relaxes the hard partitioning constraint but still keeps γ_{ij} sparse. Alg.1 describes the procedure in detail. To avoid double-counting observed points in the final GMM, we introduce a per-point weighting factor into the E-Step, called p_i. The total collection of all weights is denoted $\mathcal{P}$. ξ is a normalization constant such that $\frac{p_i}{\xi}$ over all active partitions sums to 1.0.

Algorithm 1 Expectation Sparsification Algorithm

```
procedure PARTITION(Z, P, Θ, Ĵ, λ_p)
    for z_i, p_i ∈ {Z, P} in parallel do
        calculate γ_ij, ∀j ∈ Ĵ
        for γ_ij ≥ λ_p do
            add z_i to partition j as {z_i, p_i γ_ij / ξ}
        end for
    end for
    return partitions_j, ∀j ∈ Ĵ
end procedure
```

In this formulation, a point can now contribute to multiple partitions, but an additional piece of information, p_i, needs to be recorded such that the observed point in a given partition contributes exactly $\sum_j \gamma_{ij} = p_i$. In this way, we can use λ_p to control the amount of context sharing

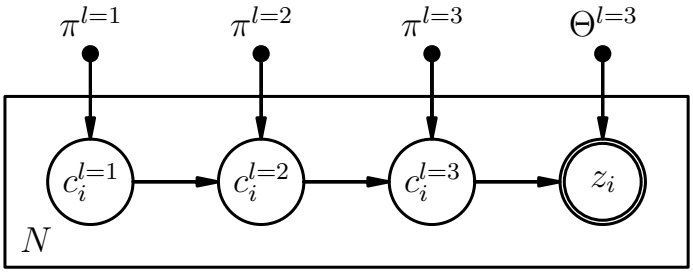


Figure 5.3: **Graphical model of hierarchical GMM** An example of a three-level hierarchy, where a series of causally linked latent variables are used to identify salient geometric regions of influence for each observed point $\mathbf{z}_i$.

among children of different parents. Given that λ_p is sufficiently small, the algorithm will remain both highly efficient and parallelizable as only a small amount of "border points" will need to be counted multiple times in different partitions.

Note that this efficiency depends on the value of lambda being small but nonzero. If we were to set $\lambda_p = 0$, we would then be required to visit every point, cluster tuple in the entire tree, thus eradicating our logarithmic time efficiency with respect to J.

5.4.5 Parallel Construction

Additionally, we can further accelerate the calculation of expectations by parallelization as described in Chapter 3. Inspecting Equations 5.5-5.7 reveals that one only needs to keep track of J zeroth, first and second moments, weighted by their expectations,

$$\{T_j^0, T_j^1, T_j^2\} \stackrel{\text{def}}{=} \{\sum_i \gamma_{ij}, \sum_i \gamma_{ij}\mathbf{z}_i, \sum_i \gamma_{ij}\mathbf{z}_i\mathbf{z}_i^T\} \tag{5.11}$$

These constitute sufficient statistics for the GMM. For the purposes of parallelization, the calculation of the above three quantities can be done in two steps: (1) Calculation of each γ_{ij} and (2) a weighted sum over all zeroth, first, and second moments. The former requires information about all J clusters but no information needs to be shared among points. The latter requires information about all N points but no information is needed from the J clusters once γ_{ij} are calculated. This allows for point-level parallelism in calculating γ_{ij} and an efficient point-level reduction sum when calculating the weighted moments.

5.5 Implementation Details

We first review the implementation of our hierarchical EM algorithm using hard partitions, and then in Sec. 5.5.1 discuss a generalization to soft partitioning.

Algorithm 2 shows the pseudocode for implementing the hard partitioned variant.

HIERARCHICAL_EM: The algorithm takes as an input a point cloud $\mathcal{Z}$, the maximum number of levels of recursion, L, and two convergence parameters λ_s, λ_d. The first convergence parameter controls the stopping condition for a given set of EM steps, and the second convergence

Algorithm 2 Hierarchical EM with Hard Partitions

procedure HIERARCHICAL_EM($\mathcal{Z}, L, \lambda_s, \lambda_d$)
  Init: parentIdx $\leftarrow \{-1\}_N$; $\mathbf{\Theta} \leftarrow \mathbf{\Theta}_{\text{init}}$
  for $l = 0 \ldots L-1$ **do**
    while !Converged(λ_s) **do**
      $\{T^0, T^1, T^2, \text{currIdx}\} \leftarrow$ E_step($\mathcal{Z}$, Θ, parentIdx)
      $\mathbf{\Theta} \leftarrow$ M_step($T^0, T^1, T^2, l, \lambda_d$)
    end while
    parentIdx $\leftarrow$ currIdx
  end for
end procedure
procedure E_STEP($\mathcal{Z}$, Θ, parentIdx)
  for $i \in$ size($\mathcal{Z}$) **in parallel do**
    for $j \in$ Children(parentIdx[i]) **do**
      $\gamma_{ij} \propto \pi_j \mathcal{N}(\mathbf{z}_i | \mathbf{\Theta}_j)$
      $\{T_j^0, T_j^1, T_j^2\} \leftarrow$Accumulate($T_j^0, T_j^1, T_j^2, \gamma_{ij}, \mathbf{z}_i$)
    end for
    currIdx[i] $\leftarrow j$ s.t. $\max(\gamma_i) = \gamma_{ij}$
  end for
  return $\{T^0, T^1, T^2, \text{currIdx}\}$
end procedure
procedure M_STEP(T^0, T^1, T^2, l)
  for $j \in$ Level(l) **in parallel do**
    $\mathbf{\Theta}_j \leftarrow$ ML_Estimator(T_j^0, T_j^1, T_j^2)
    if !Supported(T_j^0, λ_d) **then** $\pi_j \leftarrow 0$
  end for
  return $\mathbf{\Theta}$
end procedure

parameter controls the degree of geometric complexity of the final output by dropping clusters with insufficient support. To initialize Θ, we set our means to be the corners of the unit cube centered around zero. Note that during the execution of our algorithm we implicitly and recursively scale and offset the data to fit within the unit cube. The mixing weights are initially equal. Since these values are the same regardless of the recursion level for every new set of $\hat{J}$ subsurfaces, we only need to set these once at the very beginning of the algorithm. Likewise, we need to initialize an integer array *parentIdx* of size N to the value of -1, which will give us the correct child indices when $l = 0$ ($[0 \dots 7]$) to iterate over inside the first level's E step. After initialization is complete, we then iterate through L levels of the EM algorithm. After a given level has converged, we update our *parentIdx* array to point to the Maximum Likelihood estimates of subsurface expectation, recorded in *currIdx* during each iteration of the E step.

E_STEP: The E step calculates expectations over the child mixtures given the ML expectation of every point to the set of parent mixtures. The weighted moments $\{T^0, T^1, T^2\}$ (Eq. 5.11) can be calculated efficiently and in parallel using sum reductions or CUDA's *atomicAdd* functionality.

M_STEP: While the E step parallelizes over points, the M step parallelizes over subsurfaces (see Section 5.4.5). The *ML_Estimator* updates the model according to the standard MLE equations for GMM-based EM (cf. Eq. 5.5-5.7). Tikhonov regularization is done on the covariances to prevent numerical instability. Finally, clusters are dropped with insufficient support.

Note that if we implicitly encode the Gaussian Mixture tree in a large flat statically allocated array, the indexing functions *Children* and *Level* can be calculated in constant time: $\text{Children}(i) = [(i+1)\hat{J} \dots (i+2)\hat{J} - 1]$ and $\text{Level}(l) = [\frac{\hat{J}(\hat{J}^l - 1)}{\hat{J}-1} \dots \frac{\hat{J}(\hat{J}^{l+1} - 1)}{\hat{J}-1} - 1]$.

Algorithm 3 E Step with Soft Partitions

procedure E_STEP($\{\mathcal{Z}, \mathcal{P}\}_K, \Theta$)
  for $\mathbf{z}_i, p_{ik} \in \{\mathcal{Z}, \mathcal{P}\}_k, \forall k = 1 \dots K$ **in parallel do**
    for $j \in$ Children(k) **do**
      $\gamma_{ij} \propto \pi_j \mathcal{N}(\mathbf{z}_i | \mathbf{\Theta}_j)$
      $\{T_j^0, T_j^1, T_j^2\} \leftarrow$Accumulate$(T_j^0, T_j^1, T_j^2, p_{ik}\gamma_{ij}, \mathbf{z}_i)$
    end for
  end for
  return $\{T^0, T^1, T^2\}$
end procedure

By looking at the construction of the algorithm and noting that $L = \log_{\hat{J}}(J)$ and $J \ll N$, we can see that the algorithm will run in $O(k \log_{\hat{J}}(J)(\hat{J}N))$, where k is the number of EM iterations until convergence. The normal "flat" EM algorithm would execute in $O(kNJ)$. Thus, we have produced an algorithm that speeds up model creation exponentially with respect to J, the total number of mixtures in the model. Furthermore, we have liberated J as a parameter that must be set *a priori*, instead letting the convergence criterion λ_s and low support threshold λ_d determine when the point cloud has been sufficiently segmented.

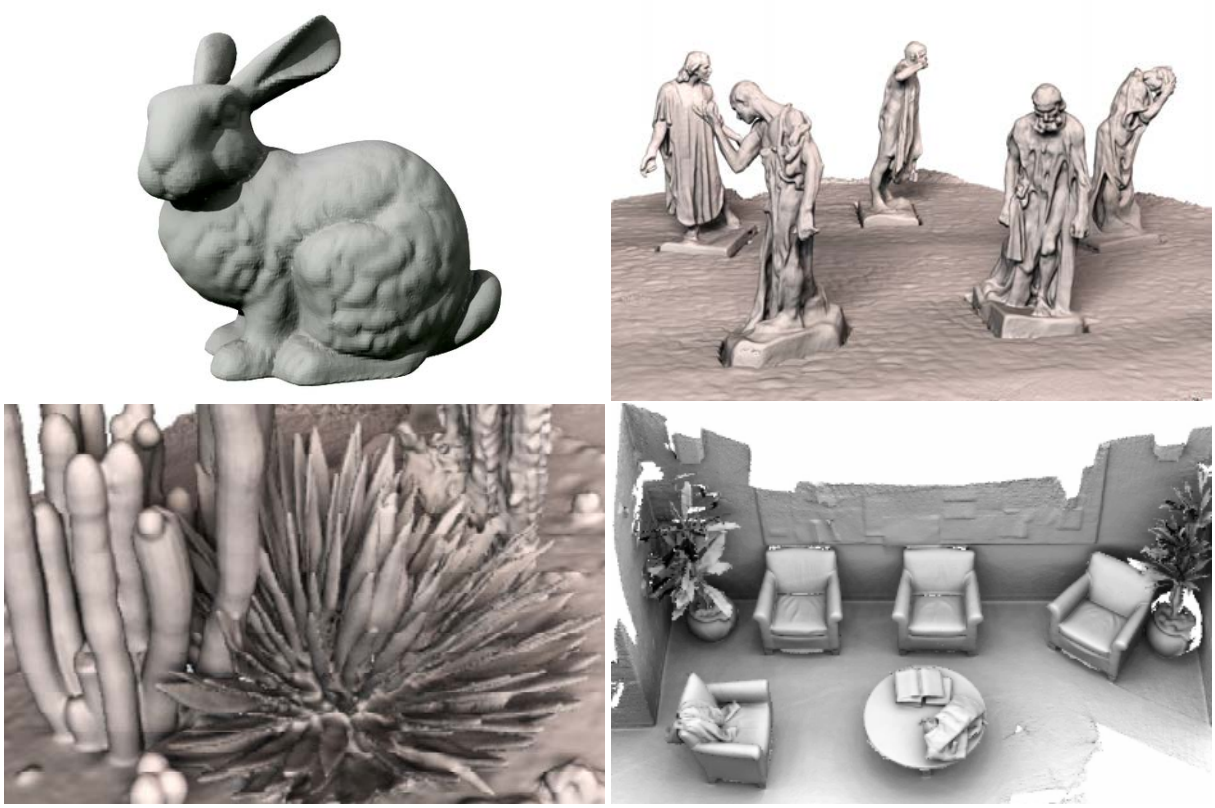
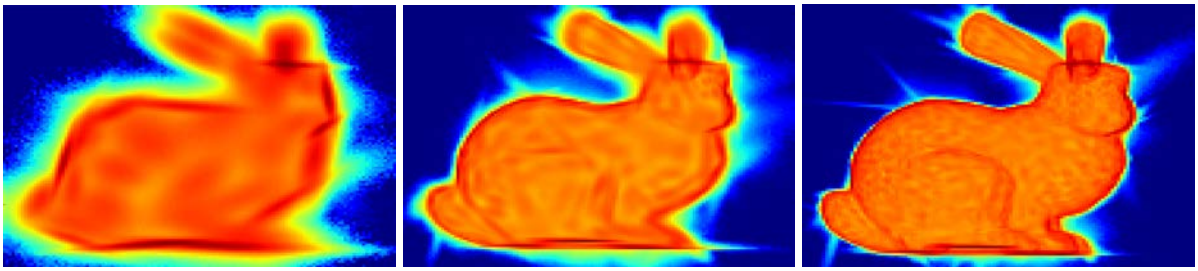


Figure 5.4: **Data set used for our evaluation:** (Top left): Bunny - 35,947 points, (Top right): Burghers - 3.47M points, (Bottom left): Cactus - 1.9M points, (Bottom right): Lounge - 1.62M points

5.5.1 Soft Partitioning

For hard partitions, updating the pointer *parentIdx* after EM convergence is all that is necessary for hierarchical construction since in the subsequent level we can then use the updated *parentIdx* in conjunction with the *Children* function as our index array into the tree.

To generalize the algorithm presented in Alg.2 to soft partitions, however, we need to record a few more pieces of data. Instead of a single *parentIdx* array, we need to record all expectations that fall above λ_p, as per the partitioning function outlined in Alg.1. Thus, we need to store both the index of partitioned points and their respective soft partitioning weights, $\frac{p_i \gamma_{ij}}{\xi}$. To do this, we modify line 8 of Alg.2 to instead call the *Partition* function from Alg. 1. The E step is then modified according to Alg.3. The only other change is that now inside *ML_Estimator* of the M Step: the new mix value must now be normalized using $\sum_i^N p_i$ and not the normal N (cf. Eq.5.5-5.7). With the modifications, a point in multiple subsurfaces will get distributed recursively throughout the hierarchy to all branches containing those subsurfaces in such a way that its expected contribution still sums to one among them. This important bookkeeping operation keeps consistency among the different paths down the tree.



L2: 45.7 PSNR, 2.5kB
170x less data

L3: 48.7 PSNR, 20kB
21x less data

L4: 50.7 PSNR, 159kB
2.6x less data

Figure 5.5: **Levels of Fidelity:** The colors in each heatmap shows the accumulated PDF values projected onto the screen space. PSNR, model size, and *the reduced storage size* are shown for each level. We vary the level of detail (L2 to L4) to show the trade-off between storage size and fidelity. The original PCD is 421 kB.

5.6 Model Evaluation

We evaluate our model with respect to reconstruction fidelity and construction execution time. We used for testing the Stanford Bunny ($\sim$36k points) and the Stanford 3D scene dataset, containing scenes with approximately $\sim$1-3 million points [144, 156]. See Figure 5.4 for more details. On all these experiments, we statically set $\hat{J} = 8$ and our sparsity constraint, $\lambda_p = 0.1$.

5.6.1 Reconstruction Fidelity

The reconstruction fidelity provides a measure of how well our model can re-create the original point cloud. For this purpose, we use a PSNR (Peak Signal to Noise Ratio) metric derived from the Hausdorff distance as suggested by [15]. Specifically, given a reference a point cloud of size N, we stochastically generate an equivalent N amount of points from our model (This method is later described in detail in Chapter 6). Then, for every point in the original point cloud, we find the nearest neighbor in the reconstructed cloud. The logarithm of the inverse root mean squared error for all points relative to the bounding box size around the point cloud gives our PSNR metric. Note that the PSNR is on a logarithmic scale. Thus, linear increases in PSNR correspond to exponential reductions in average point-to-point recreation error. By computing the PSNR at different levels of details of our model we can provide an insight on the trade-off between model size and fidelity and fairly compare these trade-offs against other generative techniques.

Figure 5.5 shows a visual representation of the PDF, PSNR, and model size for three different levels of the Bunny model. Though many different potential GMMs may be extracted from the hierarchy, we restrict our comparisons to GMMs consisting of leaf nodes from different maximum levels. For example, a "Level 3" GMM would be the GMM extracted from the hierarchy by taking all the leaves of the GMM tree at a max depth of 3. As seen in Figure 5.5, the PDF from Level 2 provides a fairly good approximation while at the same time using $\sim$170 less storage than the original PCD. By level 4, the fidelity is such that points generated from the model are virtually indistinguishable from the original PCD. Because model construction using the hierarchical GMM framework can be viewed as the sparse embedding of 3D points into a higher dimensional 10D space, we can save on storage space with respect to the original size of

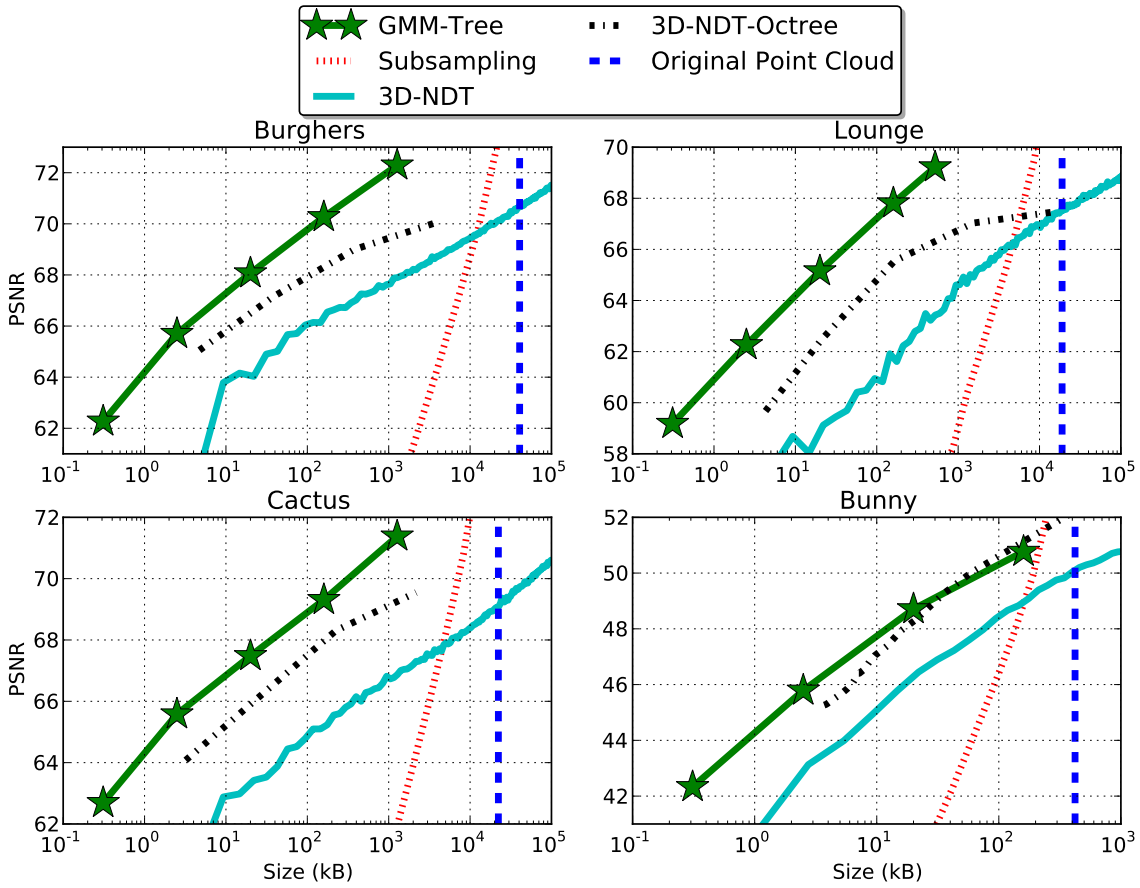


Figure 5.6: A comparison of data structure size vs fidelity over several standard point cloud datasets. The blue dashed line indicates the original point cloud size. Note the x-axis is on log scale. The star markers indicate different levels in the GMM hierarchy. At similar size models, the hierarchical GMM has much better PSNR (reconstruction performance) with respect to the original data when compared against the 3D-NDT, 3D-NDT-Octree, and a simple subsampling strategy for point cloud reduction.

the PCD, while still retaining a high level of geometric fidelity.

We compare our model against the 3D-NDT representation [5] and its Octree-based variant [84]. We chose the 3D-NDT as it is a widely used state-of-the-art generative model. To generate the trendlines for 3D-NDT, we calculate models at increasingly finer voxel resolution, and for the Octree variant, we use increasingly smaller splitting thresholds. As a baseline we compute a subsampled version of the original point-cloud. Randomly subsampling of the PCD can be seen as a basic way to reduce the data size and is a common preprocessing technique for many 3D point algorithms. Fig. 5.6 shows the fidelity reconstruction vs storage size results. In every case but the bunny, we find that our model performs favorably to both NDT variants. This is because our optimization procedure yields a more compact representation without the need for voxelization. In the case of the bunny, however, the original point cloud is small enough that at higher model sizes we see diminishing returns over the Octree-based NDT. In terms of statisti-

cal downsampling, both the hierarchical GMM and 3D-NDT are clearly much better choices to downsize the data while still retaining high geometric fidelity: for example, the level 5 GMM for Burghers achieves 72 PSNR with respect to the original point cloud, whereas a subsampled point cloud of equivalent size only yields 60 PSNR. Similarly, the smallest GMM more than doubles the PSNR over downsampling. To give another interpretation of this data: the level 3 GMM for Burghers contains as much reconstruction fidelity as if the point cloud were subsampled to about a fifth of its size, however, the level 3 GMM uses about 450x less memory.

In order to gain further intuition regarding the differences in fidelity shown in Figure 5.6, Figures 5.7-5.10 show a highlighted point on each of the series of trendlines in Figure 5.6 for which we show the actual point cloud resampling used to calculate the fidelity of the model (via PSNR with respect to the reference PCD). It is perhaps easier to understand differences in fidelity by visual inspection than just by looking at a numerical PSNR metric. As one can see from these figures, our model is able to recreate the original data much more faithfully than competing methods.

	Burghers (D) 307k pts	Bunny (D) 44k pts	Raw depth (M) 60k pts
L1	32.6ms (31Hz)	2.6ms (385Hz)	10.6ms (95Hz)
L2	57.8ms (17Hz)	4.3ms (233Hz)	15.3ms (65Hz)
L3	72.6ms (14Hz)	7.7ms (130Hz)	22.0ms (45Hz)
L4	92.2ms (11Hz)	11.7ms (85Hz)	31.5ms (32Hz)

Table 5.2: **Hierarchy construction time.** L1 to L4 refers to a level of the hierarchy (i.e., L3 denotes the the process including L1 to L3). D refers to a desktop computer (i5-3500/GTX660) used for the computation, and M denotes a Mobile device (NVIDIA Shield tablet). Raw depth refers the point cloud directly captured from Softkinetic DS325 depth camera.

#J	Hierarchy	Flat	Speed-up
8	79.2 ms	61.6 ms	0.78×
64	145.6 ms	166.5 ms	1.14×
512	184.2 ms	1145.4 ms	6.22×
4096	213.8 ms	8750.8 ms	40.93×
32768	251.0 ms	71061.7 ms	283.11×

Table 5.3: **Construction speed-up relative to a flat GMM model.** The table compares the E Step execution time of the hierarchical GMM compared with a flat GMM having the same number of mixtures on the full Burghers model (∼4.5M pts). At higher detail levels, the proposed hierarchical GMM is significantly faster to build than the flat GMM.

5.6.2 Computational Speed and Scalability

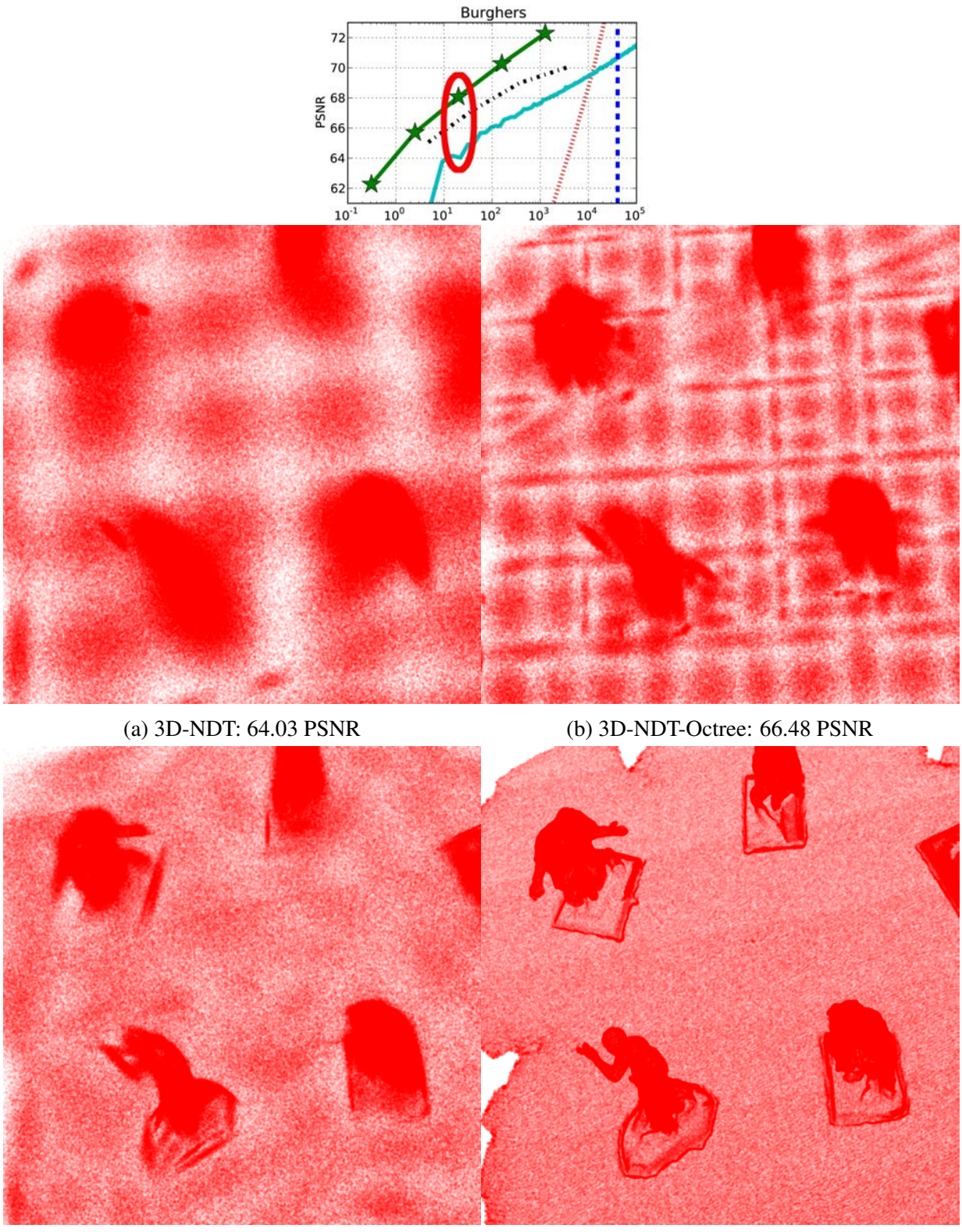
We now take a look at the execution time of the hierarchical model construction. We implemented our algorithm in C++/CUDA and are able to run it on desktop and mobile platforms. Table 5.2 shows the construction times for each level in the hierarchy over different data sets. Note that, even on the mobile platform, the PCD from a depth camera can be processed at rates higher than 60FPS for a level 2 decomposition. In addition, we compare the construction time of our hierarchical model against the flat GMM model. We show in Table 5.3 that our hierarchical method becomes increasingly faster relative to a flat version with larger numbers of clusters, making our method more suitable in applications where high fidelity is desired yet construction times need to remain low.

5.7 Summary

In this chapter, we introduced a hierarchical data structure based on Gaussian mixture models that comprises a compact and generative representation for 3D point cloud data. The creation of the model is accelerated by producing bounds on spatial interactions between the points and model. We have shown that our hierarchical model construction is orders of magnitude faster than the equivalent flat model, which follows as a result of replacing a large EM problem into multiple smaller ones. The model is also shown as a continuous PDF that effectively models the

original data at different levels of detail, while requiring significantly less storage. Compared to discrete methods or hybrid approaches such as the NDT, our model yields higher fidelity results at smaller sizes, with modest construction times even on mobile hardware.

We introduced a novel compressed representation for 3D PCD using a hierarchy of GMMs. We developed a technique of *expectation sparsification* to bound spatial interactions and thus allow for more independent computation and faster convergence, accelerating model construction exponentially with respect to the number of mixtures. The representation is a continuous PDF that compactly represents the input data, effectively serving both as a lossy compression mechanism and as a method for obtaining a probabilistic spatial hierarchy.



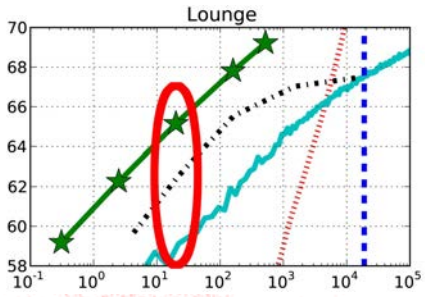
(a) 3D-NDT: 64.03 PSNR

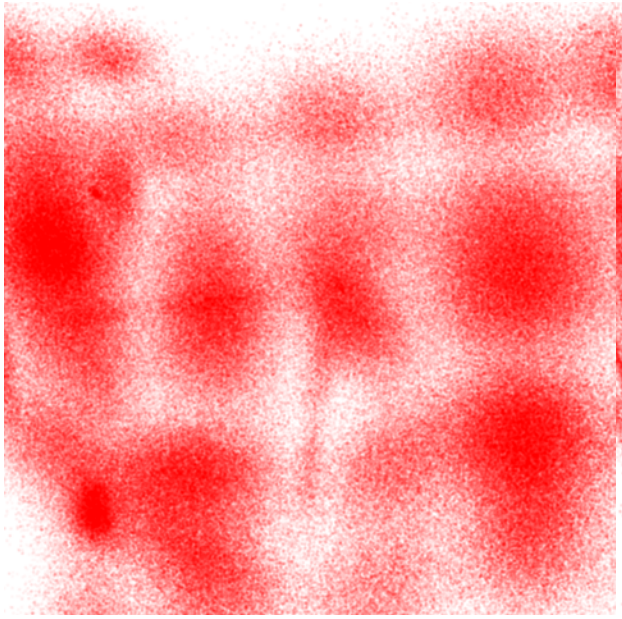
(b) 3D-NDT-Octree: 66.48 PSNR

(c) Proposed Model: 68.07 PSNR

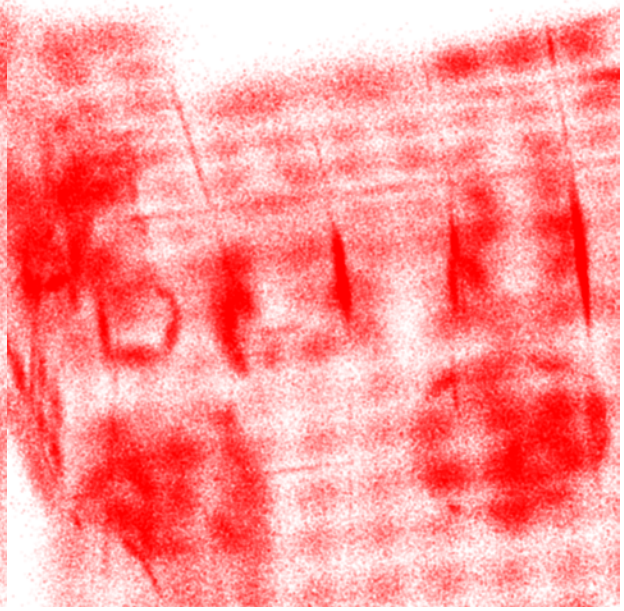
(d) Reference PCD

Figure 5.7: **Burghers of Calais:** Comparison of reconstruction fidelity for same-sized models.

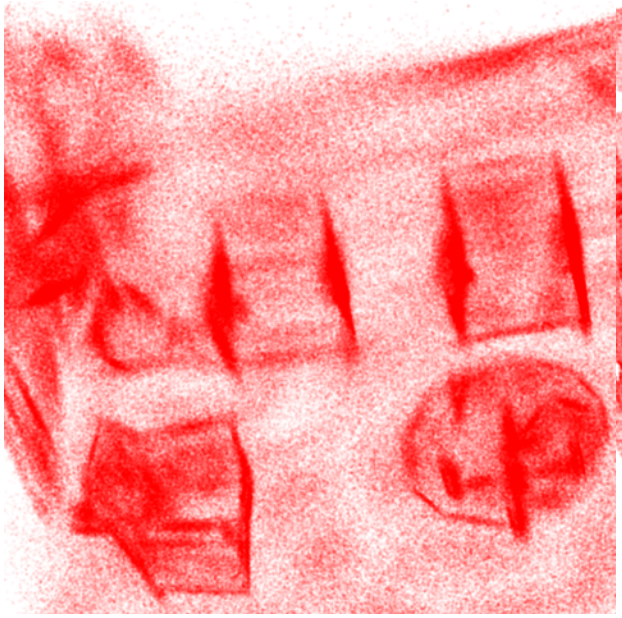




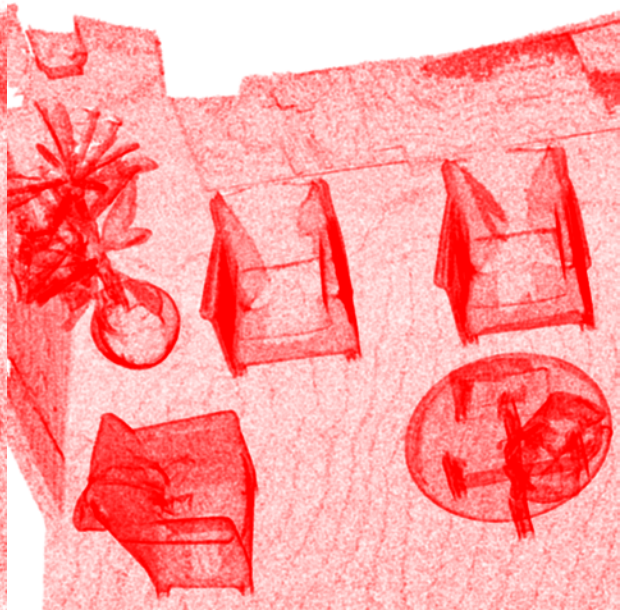
(a) 3D-NDT: 59.12 PSNR



(b) 3D-NDT-Octree: 62.37 PSNR

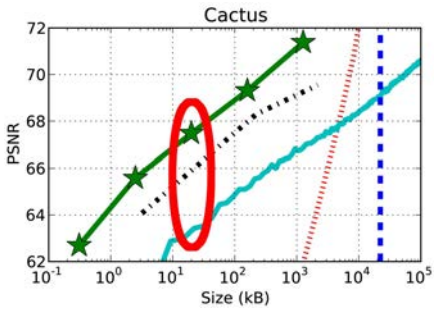


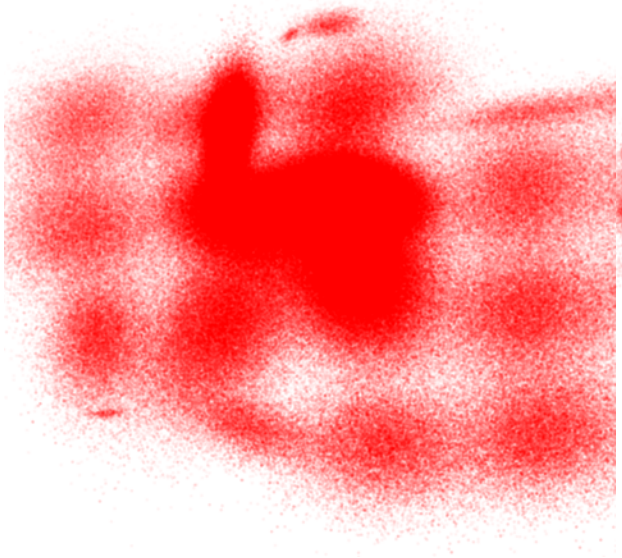
(c) Proposed Model: 65.15 PSNR



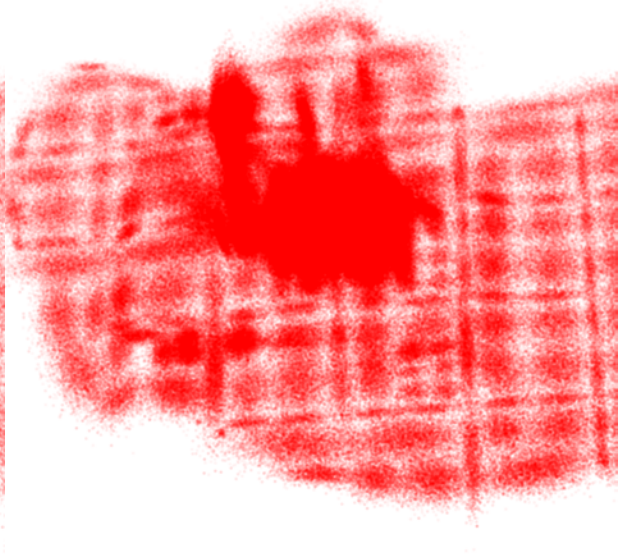
(d) Reference PCD

Figure 5.8: **Lounge:** Comparison of reconstruction fidelity for same-sized models.

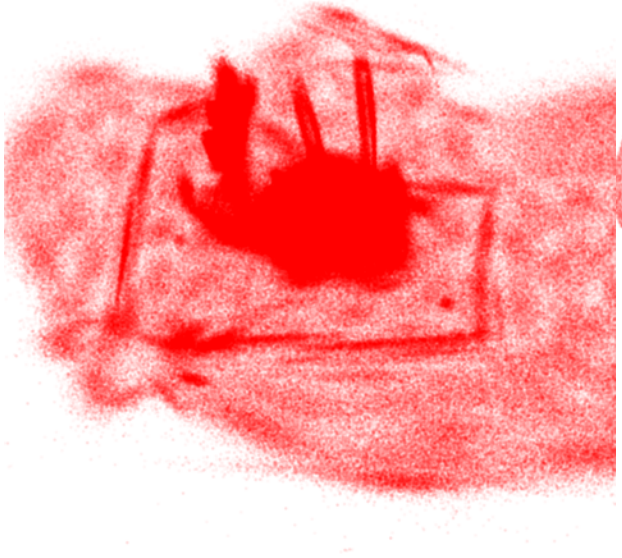




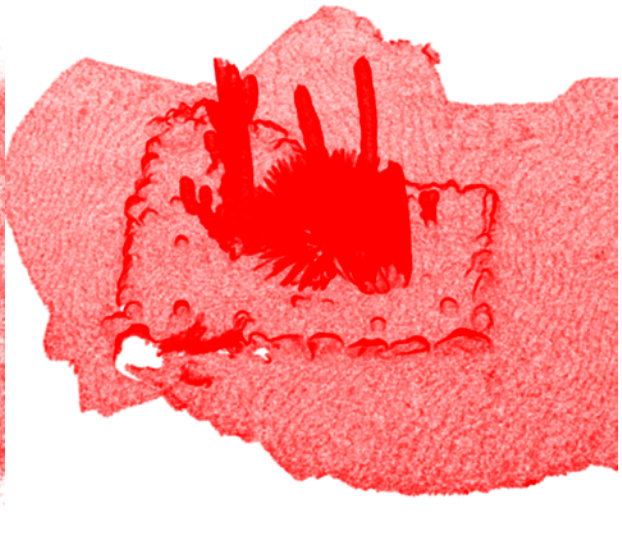
(a) 3D-NDT: 63.42 PSNR



(b) 3D-NDT-Octree: 65.83 PSNR

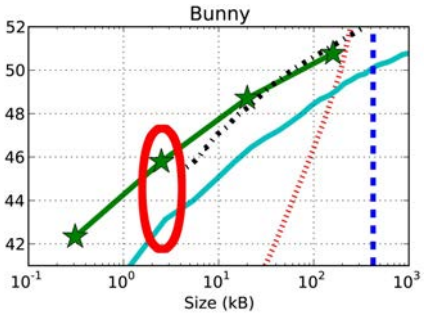


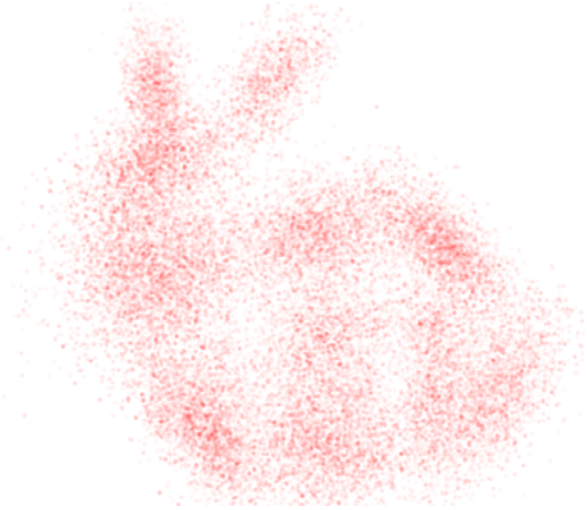
(c) Proposed Model: 67.48 PSNR



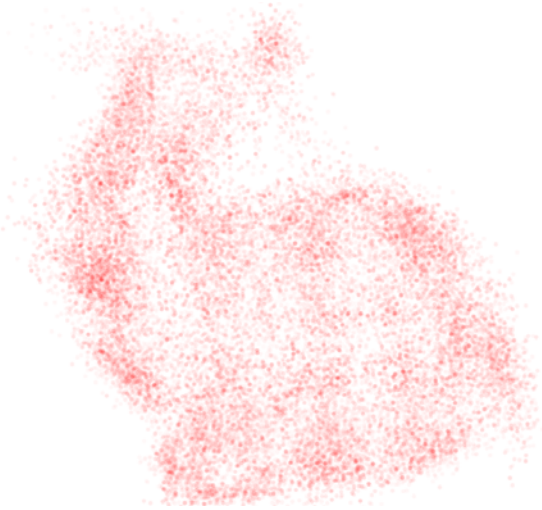
(d) Reference PCD

Figure 5.9: **Cactus Garden:** Comparison of reconstruction fidelity for same-sized models.

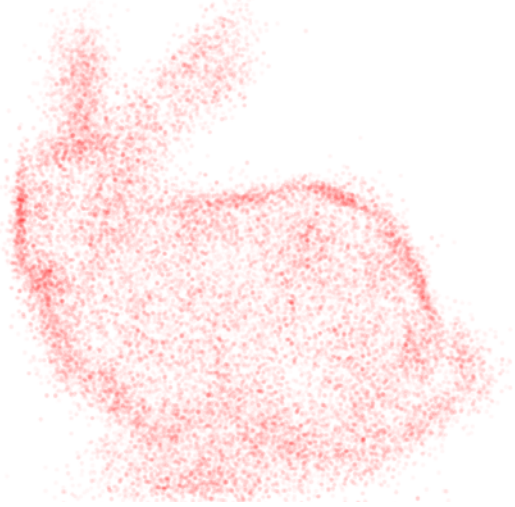




(a) 3D-NDT: 43.02 PSNR



(b) 3D-NDT-Octree: 44.47 PSNR



(c) Proposed Model: 45.79 PSNR

(d) Reference PCD

Figure 5.10: **Stanford Bunny:** Comparison of reconstruction fidelity for same-sized models.

Chapter 6

Inference and Visualization

6.1 Sampling and Integration

We have so far explained how one might efficiently construct a model (Chapter 5), but once the model is obtained, it is not trivial to see how one might sample and integrate over it, two operations that are fundamental for spatial processing applications. In this Chapter, we describe several algorithms to efficiently perform sampling and integration, which we apply to the tasks of point cloud reconstruction, occupancy grid generation, and isosurface meshing. In this chapter, we present a novel importance sampling algorithm for GMMs that allows us to significantly reduce the number of samples required during integration. Because these algorithms are not specific to hierarchical GMMs, we simplify the notation to the model as defined in Chapter 3, although the hierarchical models of Chapter 5 are used in our experiments.

6.1.1 Point Cloud Reconstruction

To regenerate a set of N points we sample the distribution defined by the GMM as shown in algorithm 4. First we determine how many samples H_j to generate from each cluster j in $[1, J]$ according to the mixture weights $\boldsymbol{\mu}_j$. Then we generate the H_j for each cluster according to the normal distribution defined by $\boldsymbol{\Theta}_j$. Details on this technique, known as *ancestral sampling*, can be found in Bishop *et al* [7]. We present it here for completeness as this is an important operation on our model. Refer to Figure 7.1e for a graphical example of this result.

6.1.2 Occupancy Grid Generation

Many applications in 3D vision require grid-based occupancy estimates of space, including path planning [138], semantic perception [26], and 3D modeling [53]. We show how our model may augment these applications by allowing dynamic run-time estimates of occupancy over sparse grids. Since the spatial extent and voxel resolution of these estimates can be determined dynamically at run-time, one can avoid many of the common problems with traditional techniques: constraint extent in large scenes, discretization artifacts as a result of coarse voxel sizes, or memory bottlenecks resulting from dense high-resolution voxel grids.

Algorithm 4 Point Cloud Reconstruction

procedure PCD_RECONSTRUCT(Θ, J, N)
    calculate $\Pi_j = \sum_{i=1}^{j} \pi_i, \forall j \in J$
    $S \leftarrow N$ random uniform samples in $[0, 1)$
    $H \leftarrow$ histogram(S, Π) , Π provides bins extents
    for $j = 1 \dots J$ **do**
        $P_j \leftarrow H_j$ points sampled from $\mathcal{N}(\mu_j|\Sigma_j)$
    end for
    return $P_j, \forall j \in J$
end procedure

To construct an occupancy grid we can stochastically sample points directly from the model to perform a Monte Carlo estimation of the probability that a given region of space is occupied. More formally, to build a discrete occupancy voxel grid we would like to spatially integrate our PDF over each voxel to obtain its probability estimate,

$$p(V_k|\mathbf{\Theta}) = \int_{V_k} \sum_{j=1}^{J+1} \pi_j p(v|\mathbf{\Theta}_j) dv, \tag{6.1}$$

where V_k is a particular indexed cube or voxel.

Since there is no analytical solution to this integral, we resort to Monte Carlo sampling. However, uniform sampling the PDF over the space of the voxel grid will likely yield estimates with very high variance since the previously discussed sparsity will render many areas to essentially zero probability. Thus, we employ the use of importance sampling. To see how importance sampling is quite efficient in this context, we need to re-interpret the GMM as a weighted sum of zero-mean isotropic Gaussians that have been skewed and shifted through 3D space. To do this, we perform a Cholesky decomposition on the covariances, $\mathbf{\Sigma}_j = \mathbf{U}_j^T \mathbf{U}_j$. Then the multivariate normal equation is,

$$\mathcal{N}(x_i|\mathbf{\Theta}_j) = \xi_j^{-1} e^{-\frac{1}{2}(x_i - \boldsymbol{\mu}_j)^T \mathbf{U}_j^T \mathbf{U}_j (x_i - \boldsymbol{\mu}_j)} \tag{6.2}$$

$$= \xi_j^{-1} e^{-\frac{1}{2}\|\mathbf{A}_j x_i - \mathbf{b}_j\|^2}, \tag{6.3}$$

where $\mathbf{A}_j = \mathbf{U}_j$ and $\mathbf{b}_j = \mathbf{U_j}\boldsymbol{\mu}_j$, and ξ_j is a normalization factor. Thus, we can interpret each input x_i as undergoing an affine transformation before being evaluated through a zero-mean Gaussian function with identity covariance. To efficiently sample from the GMM therefore we first sample uniformly over $[0, 1]$ in 3 dimensions and then transform the values through the Φ^{-1} (probit) function. Our derived affine transformations of the samples $X \sim \mathcal{N}(0|I)$ for each of J subsurfaces place them in the GMM space. Furthermore, since the GMM is simply a linear combination of many Gaussians, once we have a collection of transformed samples, one only needs to keep track of what proportion of these samples, per cluster, fall into a particular voxel and then multiply this ratio by the appropriate mixing parameter. Thus,

$$p(V_k|\mathbf{\Theta}) \approx \sum_{j=1}^{J+1} \frac{\pi_j}{N} \sum_{i=1}^{i=N} I_{V_k}(x_i) \tag{6.4}$$

where $x_i \sim \mathcal{N}(\boldsymbol{\mu}_j|\boldsymbol{\Sigma}_j)$ and I is an indicator function for whether x_i falls within the bounds of V_k.

Since the sampling function matches the underlying PDF (up to a multiplicative constant), these calculations yield unbiased estimates of the voxel probabilities and have low variance for even a fairly small number of samples. Furthermore, we precalculate all the samples X before reconstruction so that the entire process amounts to simply binning (voxelizing) the results of J different affine transformations over a relatively small static set of random or pseudorandom points. Since the model itself contains no voxels, as opposed to voxel-based or NDT methods, we are free to dynamically choose the extent and resolution of the occupancy grid at run-time, according to any number of application-specific constraints, arbitrarily defined axes, frustum culling, or locus of attention.

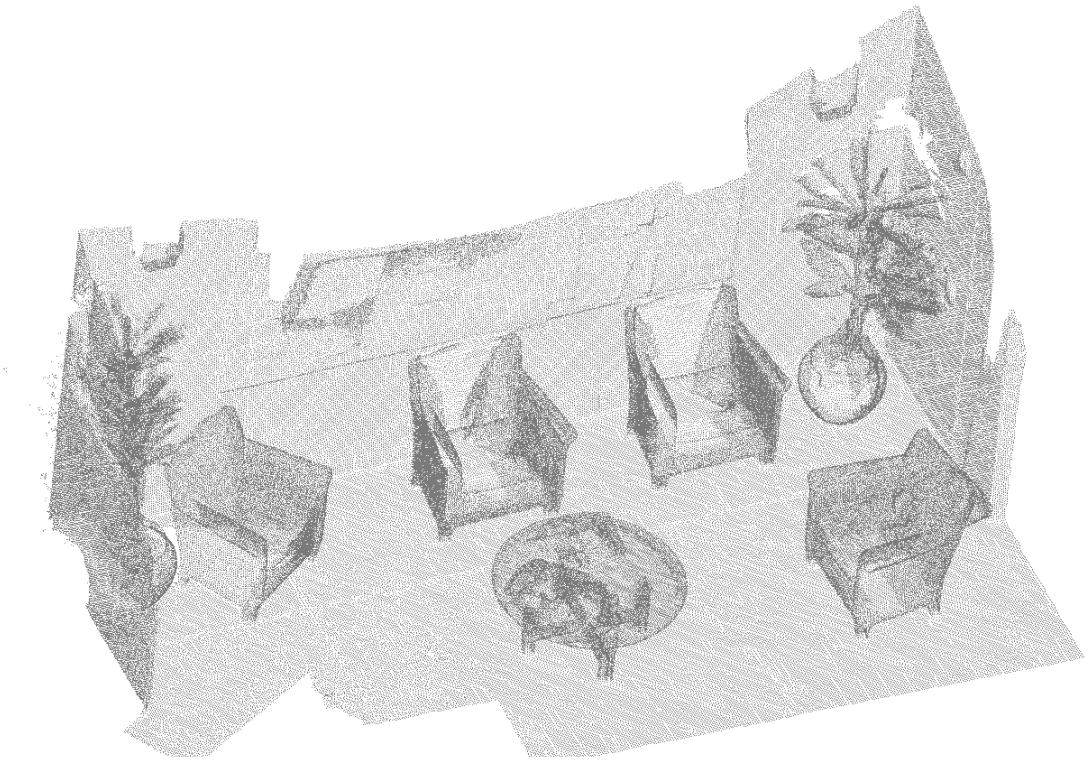
Figure 6.1 demonstrates this process for a large scene. Once we create a model from the points (in this case, a hierarchical GMM with a max depth of 5), we no longer need the raw PCD, and instead can at runtime produce a high quality occupancy map using the model only (Figure 6.1b).

6.2 Isosurface Extraction

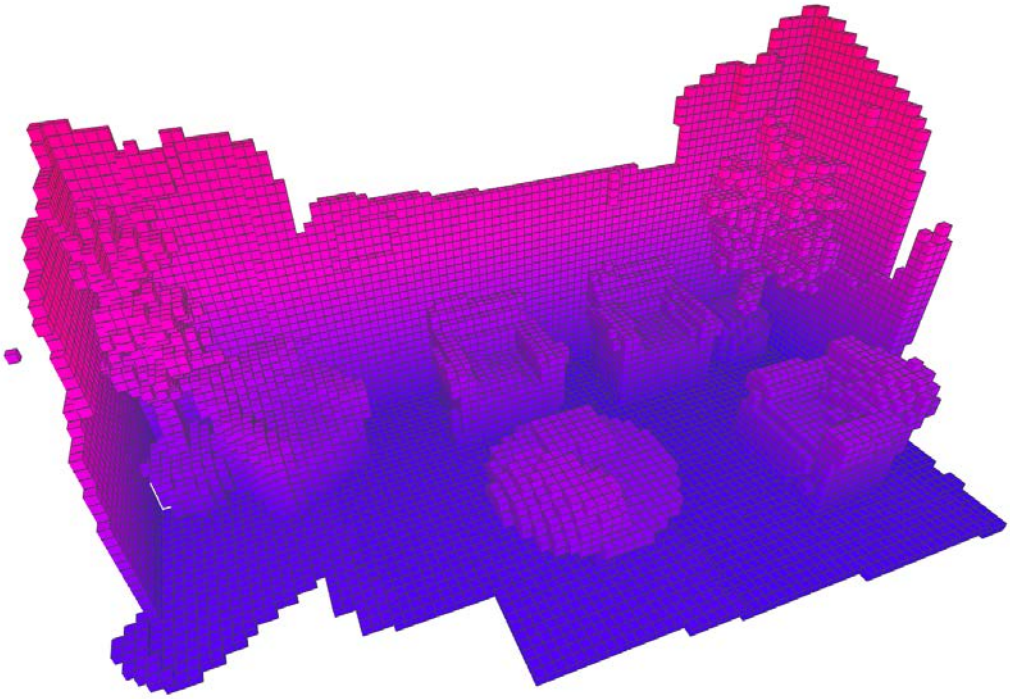
The proposed model's compact and parametric structure make it particularly well-suited for many common 3D spatial processing applications. We demonstrate one such application in this section, isosurface extraction, that uses stochastic sampling to accelerate fine-grained 3D mesh generation from probabilistic occupancy estimates in an augmented Marching Cubes framework. Being a statistical model based on data likelihood, one can derive an isovalue such that the isocontour forms a closed 3D surface representing a statistical guarantee of the presence of solid matter. Note that this is fundamentally different in nature than surface extraction; instead, extracting an isosurface from a PCD-derived GMM can be seen as a probabilistic generalization of a 3D occupancy grid, where the isocontour represents a probabilistic "bound" around the surface. Thus, isosurface extraction is applicable for cases when a probabilistic guarantee is needed in the presence of noise as to whether a given space is free or occupied. Fig. 6.2 shows an overview of our method.

6.2.1 Marching Cubes

The Marching Cubes (MC) algorithm is one of the most popular methods for isosurface extraction from volumetric data [79], with applications in medical imaging and computer graphics. In its original form, MC divides 3D space into discrete voxels and samples over the entire volume. It is a sequential algorithm that traverses the volume in raster order, causing the algorithm to scale $O(V^3)$ with the number of voxels V per dimension. Unfortunately, this cubic scaling can cause complete voxel traversal to be a slow process, in particular if a large percentage of the cells in the volume are non-active. This fact is especially true when applying these methods to data from range sensors, since the data typically inhabits only a small subset of the entire volume. Roughly speaking, the occupied voxels from points observed by range sensors grow $O(V^2)$ with the number of voxels per side of a volume. The quadratic scaling of active voxels to cubic scal-



(a) Input PCD (1.6 *mil* pts) [156]



(b) Occupancy grid map

Figure 6.1: **Occupancy estimates:** Top: Raw PCD. Bottom: an example high resolution occupancy map obtained by sampling a hierarchical GMM (max depth 5) produced from the points.

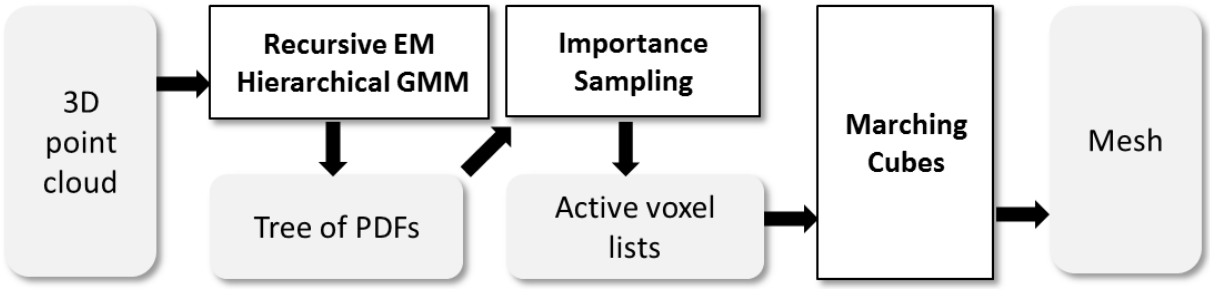


Figure 6.2: **Overview of our approach.** We first build a hierarchical GMM by recursively applying EM (Chapter 5). Secondly, we run stochastic importance sampling on the highest detail PDF to build a list of active voxels (Sec. 6.2) that we feed to marching cubes for the final isosurface reconstruction (Sec. 6.2.2).

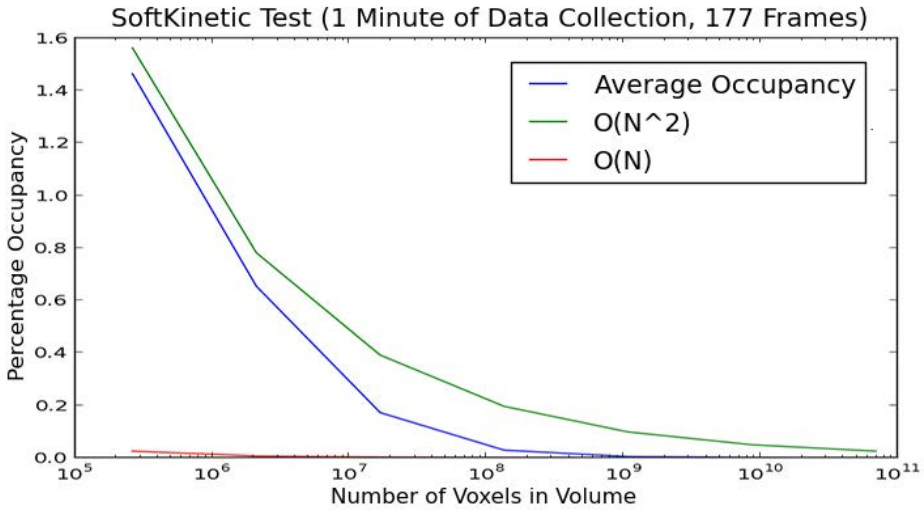


Figure 6.3: **Sparsity of point cloud data** Sparsity is a common scenario for many modern range sensors. For a sample point cloud data captured from a depth sensor, the number of occupied voxels in a sample grows close to $O(V^2)$, where the total number of voxels grows as $O(V^3)$.

ing of total voxels indicates that the sparsity with respect to the volume only increases as the resolution gets smaller.

To see this visually, we conducted a simple occupancy test of voxelizing point cloud data captured from a commodity depth sensor. Figure 6.3 shows the result of voxelizing point cloud data over a volume and then computing occupancy statistics. Occupied voxels grow roughly $O(V^2)$ as the total voxels grow $O(V^3)$.

To facilitate the discovery of active cells, most traversal acceleration algorithms pre-process the data into some form of intermediate data structure: spatial data structures such as Octrees [152], active lists [38] and span space-based representations [78], among others. These methods, however, can have a high setup cost. For example, Octrees have a creation time that is superlinear in the number of points ($O(N \lg N)$), and so can accelerate the process only if there are multiple isosurface extractions performed, since then one can reuse the data structure. Other methods

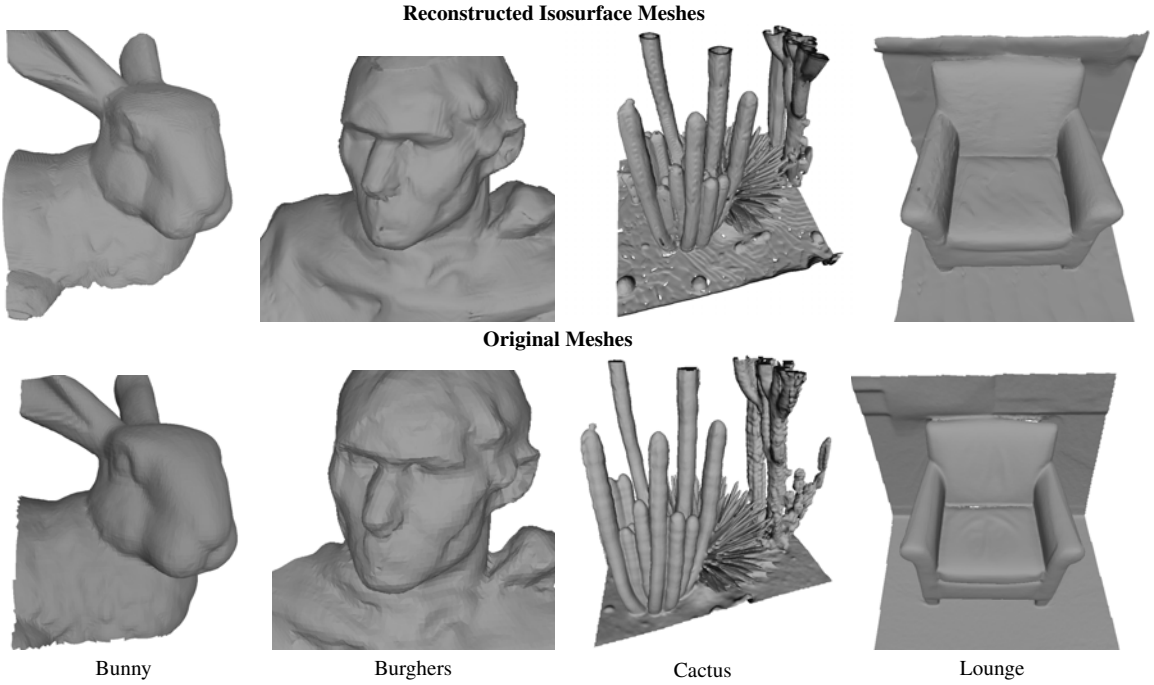


Figure 6.4: **Qualitative comparison between our reconstruction and original meshes:** Images in 1st row show isosurfaces generated from our L5 models (cropped to show detail) using stochastic marching cubes. The images in the 2nd row show the original mesh [156].

reduce the number of cells visited by starting from a known active cell and propagating to nearby active cells [131]; such an approach reduces the execution time, but picking the propagation point is not trivial. For an extensive survey of methods related to MC, we refer to Newman and Yi [101].

6.2.2 Sparse Augmentation of Marching Cubes

We would like utilize our Gaussian Mixture hierarchy to perform the isosurface extraction sparsely over the volume. Given that the underlying structure is a PDF composed of many Gaussians, in the context of point cloud data, each Gaussian has a certain spatial extent beyond which its individual contribution to the PDF is negligible. Isosurface extraction through Marching Cubes is nothing but the process of finding voxels with isovalue crossings. If all corners are above the isovalue or all corners are below the isovalue, then it is a wasted sample. By leveraging the GMM's PDF, we can we can quickly find candidates for active voxels.

Given the GMM model, we know the location and shape of areas of high probability. To construct a sparse list of voxel probabilities, we can employ the sampling technique discussed in Section 6.1.2. Because the GMM's active region over the entire volume is sparse, the voxel list of nonzero probabilities is sparse, allowing the sparse augmented MC method to be able to operate very fast even for very small voxel sizes.

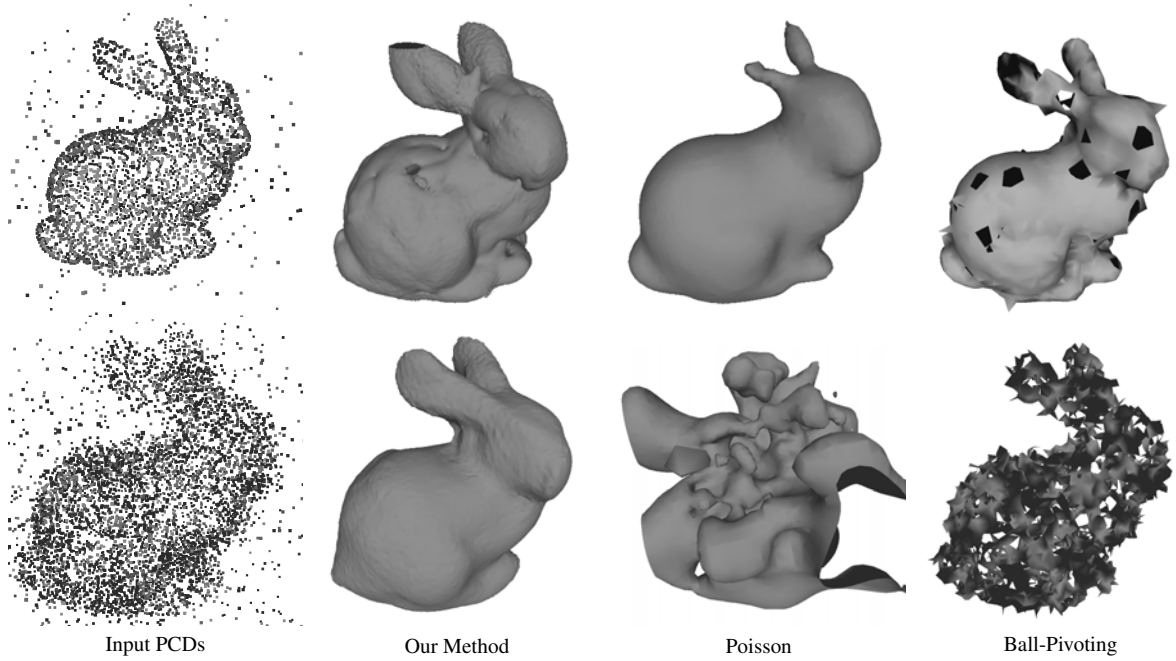


Figure 6.5: **Isosurface reconstruction under noise:** (1st column): Stanford bunny with randomly added noise (Normal distribution), (2nd column): *our results*, (3rd column): Poisson reconstruction [65], (4th column): Ball-pivoting [2]. Note that even with more aggressive noise (2nd row), our approach can provide better overall model quality.

6.2.3 Results on Datasets

Isosurface Reconstruction (Noise-Free): Note that in the case of noise-free data, any nonzero positive isovalue along with a sufficiently detailed GMM-tree will produce an isocontour that approximates the true surface of the model. Likewise, a sufficiently fine discretization of MC will produce the correct meshification of surfaces of the scene. We show these results qualitatively in Fig. 6.8 for L5 models and a 256 resolution voxel grid. The isosurface reconstruction from the point cloud data is virtually indistinguishable from the original mesh. Thus, when the models have such high PSNR, as demonstrated in Chapter 5, very high quality fine meshes can be procedurally generated from the regions of interest efficiently and accurately.
Robustness to noise: To qualitatively evaluate the robustness to noise in the point cloud data, we created synthetic data sets with added noise to the Bunny model, and compared the reconstructed isosurface meshes against other common surface reconstruction methods, as shown in in Fig. 6.5. Due to the probabilistic nature of our isosurface extraction, our meshes approximate the true shape of the Bunny better, while others, (Poisson surface reconstruction [65] and Ball-Pivoting [2]) can fail fairly dramatically under large amounts of noise.

6.2.4 Adaptive Coarse-to-Fine Scene Sampling

Even when using a sparse voxel list to track the non-zero probability voxels, we still need to define a voxel resolution and volume size for marching cubes. To produce high quality surfaces,

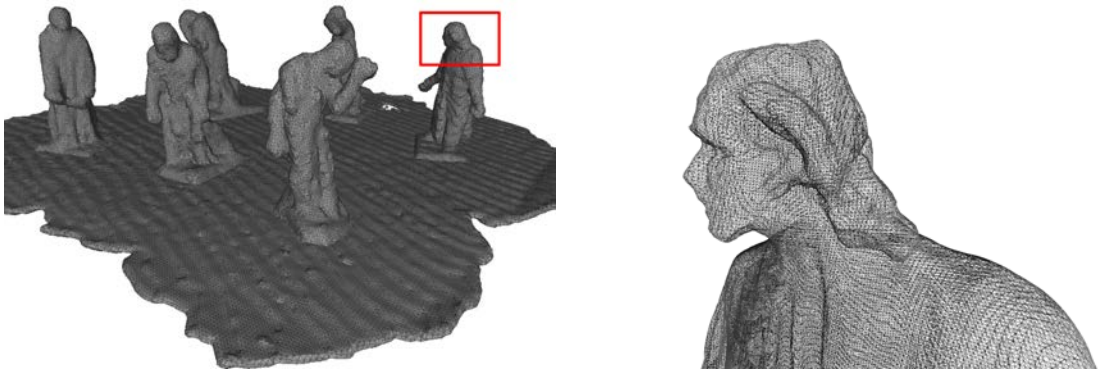


Figure 6.6: **Adaptive scene sampling:** Left : entire scene reconstruction, notice the coarseness of the triangulation (originally 3.47M points), Right: a zoomed-in region (denoted as red rectangles in Right) is finely triangulated. In both cases the sampling volume is fixed at 256x256x256.

the voxel size needs to be small enough to capture the fine detail in the underlying GMM model. On mobile devices, for example, this can be problematic, since for large scenes the number of active voxels could increase to the point of saturating system resources. One advantage of using our GMM model is that we can *decouple* the surface representation from the volume size that we use for surface reconstruction within MC. Therefore, we can fix the maximum MC volume size to keep interactive rendering rates and adaptively change which region of the scene the volume is sampling according to the current viewing frustum. Thus, when the user is viewing the entire scene, the MC volume will sample the entire scene, which will result in a coarse triangulation; on the other hand, when the user zooms into the scene, the MC volume will sample the current zoomed-in region, resulting in a finer triangulation and higher level of detail. Our current implementation handles a volume of 256^3 voxels for real-time performance. Fig. 6.6 shows an example of this coarse-to-fine adaptive sampling strategy.

Note that the discretization of space into a voxel volume is only required when a discrete surface is needed, and that our 3D model representation remains a continuous function. As such, we can define the mapping of the discrete voxel volume to the 3D space dynamically, based on need. This is different to existing methods that use discrete representations to model 3D space which work with a fixed-resolution voxel space [10, 100]. A deterministic spatial decomposition approach is much more rigid in this regard. By keeping a compact probabilistic model as our main data structure, we are afforded the freedom to dynamically adjust our level of discretization detail without needing to keep the original points in memory after the model is computed.

As an additional example, in Fig. 6.8, we enumerate the results from all our datasets, showing a slice of the sparsely sampled volume, lower level meshes of entire scenes, and high quality meshes generated from regions of interest.

6.2.5 Real-Time Operation on Mobile Device

One goal of these proposed methods is to enable 3D point cloud data processing on mobile and embedded devices. These devices range from phones and tablets, with applications in augmented reality and casual 3D model acquisition, to automobiles and mobile robots, where 3D point data

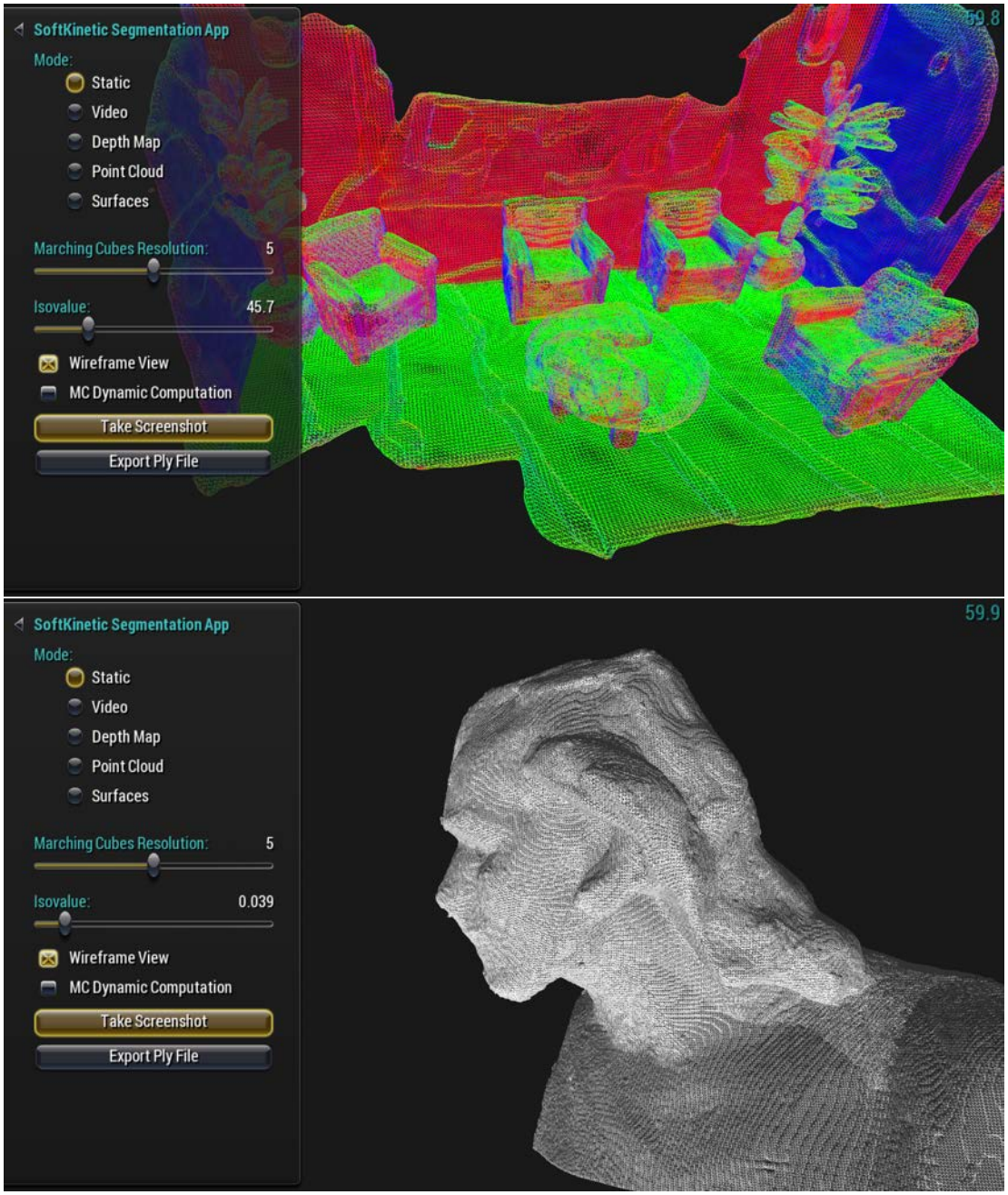


Figure 6.7: **Modeling on the Android platform** We run our approach on a mobile platform at interactive rates.

is used for navigation and path planning.

To demonstrate our computational efficiency, we deployed this pipeline onto an NVIDIA Shield Tablet. Our hierarchical segmentation is able to stream SoftKinetic data and segment in greater than real-time (356 FPS for a coarse level 1 segmentation and 124 FPS for a high fidelity level 5 segmentation). Our sparse sampling strategy for a 256x256x256 Marching Cubes grid takes about 2.5 ms (400 FPS) for a level 4 segmentation in CUDA. Fig. 6.7 shows a screenshot of our implementation running on the tablet at video rate.

Related methods for surface rendering on volumetric structures, such as KinectFusion [100] and its variants [10], require ray casting over a signed distance field. In addition, while they demonstrate video rate reconstruction on high-end CPU/GPUs, their computational complexity and storage requirements do not allow execution on mobile or embedded platforms. In contrast, our method builds a mesh from the continuous PDF, allowing us to define the discretization of the surface dynamically.

Furthermore, compared to regular surface reconstruction problems, extracting an isosurface does not require the solution of determining the "inside" or "outside" of an object. In our case, the isocontour itself has the definition that inside the surface is all the space for which the likelihood of occupation is above a certain statistical threshold. However, in the case of noise-free input, for any probability threshold above zero, a sufficiently detailed GMM-tree, and a sufficiently fine discretization of Marching Cubes will, in fact, produce the correct collection of surfaces of the scene, since the isosurface shell will have near zero-variance.

6.3 Summary

In this chapter we discussed various techniques for sampling, integrating, and visualizing the models. We demonstrated that the model's PDF can be used to reconstruct point clouds of arbitrary sizes, efficiently compute probabilistic occupancy estimates using stochastic importance sampling, and augment a Marching Cubes procedure for fine-grained isosurface meshing. The marching cubes volume is decoupled from the 3D surface representation, which allows us to set the voxel and volume size dynamically based on the availability of computing resources. These applications form fundamental components of many spatial perception and 3D modeling applications, particularly for cases like collision detection and user interface design, where high fidelity meshification and visualization is desirable. Furthermore, we showed how, given the parallel and recursive construction of our model, these techniques can be efficiently implemented in CUDA, allowing for real-time rates on a mobile devices equipped with GPUs. Finally, note that the models used in this chapter are the exact same models we use for segmentation and registration; a single model can therefore be used for all of these distinct perceptual tasks without incurring any additional computational overhead during model creation.

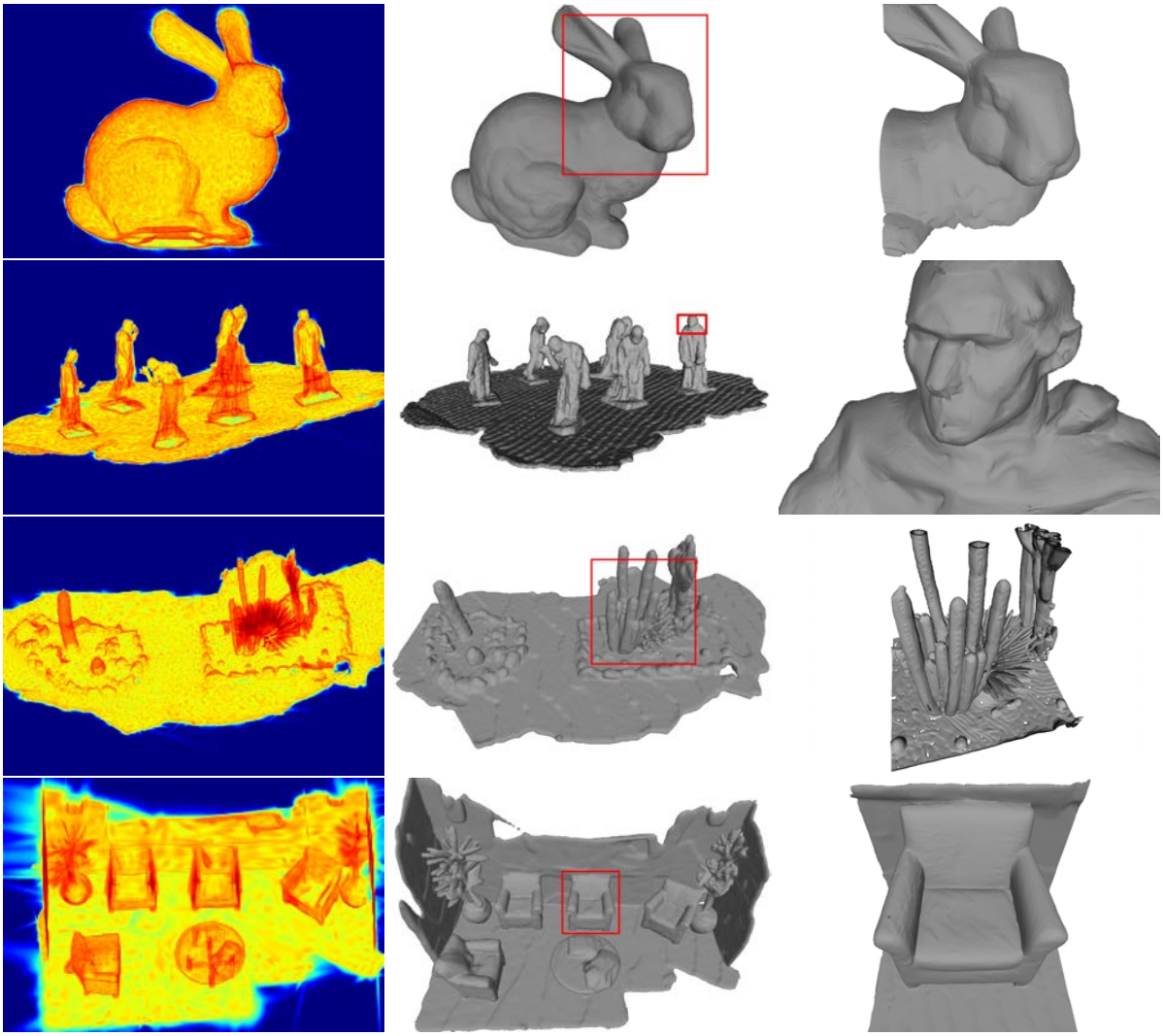


Figure 6.8: **PDFs and reconstructed structures:** (1st column): Heat maps represent the areas of high data probability (PDFs) for each model, (2nd column): The reconstructed scenes with our hierarchical GMM and stochastic marching cubes in low voxel resolution (256^3). The red rectangle denotes the region of selection where we reconstruct with higher resolution. (3rd column): Our reconstruction with higher quality. Note that the original meshes are from [156]

Chapter 7

Context-Aware Registration, Fusion, and Mapping

Point cloud registration and mapping sits at the core of many important and challenging 3D perception problems including autonomous navigation, SLAM, object/scene recognition, and augmented reality. In Chapter 4, we introduced several methods for registration that utilize the generative models described in Chapter 3. In this chapter, we build on these techniques to present a new statistical registration method that can achieve high performance even in extremely difficult situations where large pose displacement, uneven sampling patterns, or non-overlapping geometry would otherwise render traditional registration methods intractable.

Our proposed registration method combines two new techniques: 1) Mixture Decoupled Mahalanobis Estimation (MDME) (see Chapter 4) and 2) Expected Overlap Estimation (EOE). MDME calculates the most likely 6 DoF transformation between two point clouds using Gaussian Mixture Models (GMM) and a generalization of the ICP point-to-plane minimization metric, while EOE robustly estimates non-overlapping regions by using an iterative EM-like procedure to directly encode the known sensor view dynamically into the registration process. Our experiments indicate favorable performance both in speed and accuracy against the state-of-the-art, with the largest benefit appearing in situations where the amount of overlap between point clouds is very small.

7.1 Introduction

Point cloud registration is the task of aligning two or more point clouds by estimating the relative poses between them, and it has been an essential part of many computer vision and graphics algorithms such as 3D object matching [19], localization and mapping [104], dense 3D reconstruction of a scene [100], and object pose estimation [106].

Over the last several decades, point set registration has remained a popular research topic, though recently new methods [135] have been gaining more importance due to the growing commercial interest of virtual and mixed reality [89], commercial robotics, and autonomous driving applications [49, 74]. In most of these applications, massive amounts of 3D point cloud data (PCD) are directly captured from various active sensors (i.e., LiDAR and depth cameras)

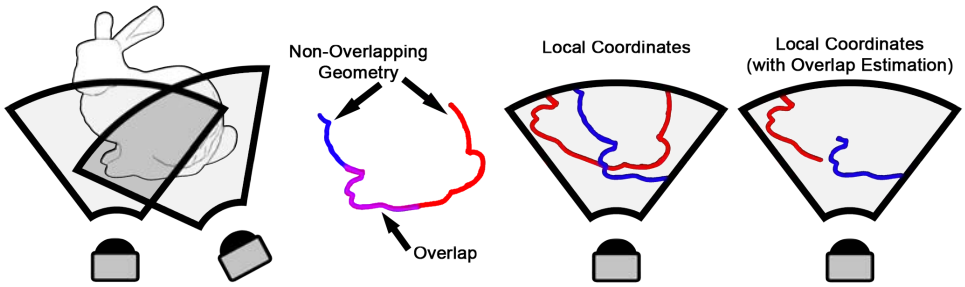


Figure 7.1: **Point Cloud Registration with Limited Overlap** Leftmost: Two range sensors at different poses capturing different slices of geometry. Note that the geometry is illustrated in 2D contours for simpler visualization: Blue and red contours for the geometry captured from each view, and magenta for overlapped region. During the registration process, the non-overlapping pieces of geometry are problematic as they may be mistakenly matched. For generative model-based methods, this situation manifests itself as a violation of the paradigm's most fundamental assumption that both point sets are *iid* samples of a single model. Rightmost: If overlap can be directly estimated as part of the generative model, then non-overlapping regions could be pruned, leading to a more well-formed registration problem both practically and theoretically.

but at different times under different poses or local coordinate systems. The task of point cloud registration is then to try to find a common coordinate system, which is done by estimating some type of geometric similarity in the point data that can be recovered through optimization over a set of spatial transformations.

One of the oldest and most widely used registration algorithms, Iterative Closest Point (ICP) [3, 11], is based on an *ad hoc* iterative matching process where point proximity heuristically establishes candidate point pair sets. Given a set of point pairs, the rigid transformation that minimizes the sum of squared point pair distances can be calculated efficiently in closed form. Given its simplistic construction, ICP and its dozens of variants [116] often fail to produce correct results in many common but challenging scenarios, where the presence of noise, uneven point cloud density, occlusions, or large pose displacements can cause a large proportion of points to be without valid matches.

Compared to traditional ICP-based approaches, much research has been done on the use of statistical models for registration, which in principle can provide better estimates for outlier rejection, convergence, and geometric matching. [40, 98, 136]. In particular, many statistical methods have been designed around the Expectation Maximization (EM) algorithm [18] as it has been shown that EM generalizes the ICP algorithm under a few basic assumptions [43, 130]. Specifically, if each point has a Gaussian noise parameter, the entire set of points can be interpreted as Gaussian Mixture Model (GMM). Therefore, many statistical registration techniques have explicitly utilized a GMM along with EM to deliver better robustness and accuracy [14, 33, 43, 51]. Unfortunately, however, these algorithms tend to be much more complex than ICP, their robustness coming at a high computational cost.

In addition to these problems, the model's underlying assumptions can often be violated and cause performance to degrade. In particular, any statistical registration method employing EM and/or GMM's as the basis for point cloud representation utilizes the following assumption: *That*

there exists some probability distribution, characterized by a generative model, for which point cloud data can be viewed as a set of independent and identically (iid) distributed samples.

The amount to which the *iid* assumption is true in practice has a direct effect on the accuracy and robustness of these statistical models. As we show in this chapter, these GMM/EM algorithms tend to break down in many common real-world scenarios due to their inability to adapt to changing views.

These limitations motivate the need for a new type of generative approach that can preserve the *iid* assumption, specifically in cases of non-overlapping point cloud regions (See Fig. 7.1). Given these stated goals, our contributions in this Chapter are summarized as follows:
First, we establish a view-aware methodology for generative model-based registration that explicitly considers only overlapped regions of geometry. We then provide an EM-like iterative procedure to dynamically estimate view overlap during registration, which we call Expected Overlap Estimation (EOE) (Sec. 7.3.2). We deploy this method as part of a new, GPU-parallel registration algorithm that uses a novel Mahalanobis distance generalization of ICP's point-to-plane distance minimization metric, called Mixture Decoupled Mahalanobis Estimation (MDME) (See Chapter 4). We then demonstrate the types of scenarios where view-awareness is most useful, how these methods compare against traditional robust outlier rejection techniques and they can be augmented with EOE, and also how MDME+EOE performs alongside other state-of-the-art generative methods and robust ICP variants. We show that MDME should be preferred in most cases to current state-of-the-art generative registration MLE metrics, and that EOE succeeds most in situations where large portions of non-overlapping geometry or noise are present in one or both point clouds.

7.2 Related Work

Our proposed technique of Expected Overlap Estimation is similar in spirit to previous work on robust outlier detection and handling under ICP. For example, FICP [113], IRLS-ICP [50], and Trimmed ICP (TrICP) [12, 13] utilize robust matching metrics in order to handle cases of partial overlap in order to down-weight and/or avoid spurious point matches. We similarly seek to detect points that have no geometry with which to match in the other point cloud, though we use an iterative EM-like approach where a known sensor field-of-view is dynamically estimated and updated. We will show experimentally that both approaches are complementary to one another.

Another related approach to exclude non-overlapping regions is to use projective association for the matching step of ICP [116]. This efficient method first transforms a pair of 2D depth maps into a single coordinate system and then pairs depth pixels sharing the same 2D index. With projective association, this pairing method can be seen as a type of implicit ray casting solution to remove data currently believed to be outside the region of overlap. Though projective ICP has seen a lot of success with range cameras such as the Kinect [100], it is not particularly well-suited for operation with sensors that do not produce a dense depth map (e.g. LIDAR), which is our main focus in this Chapter.

A crucial difference between this work and the robust and projective ICP methods previously mentioned is that we employ a statistical model-based approach instead of ICP's heuristic approach. As explained in Chapter 4, our method builds on previous work in GMM-based meth-

ods such as GMM-Reg [59][57], JRMPC [33]. By adopting a GMM-based paradigm, we gain robustness in situations of large pose displacement, optimal solutions in the form of maximum likelihood estimates, and an ability to more easily leverage point-level parallelism on the GPU (Chapter 3).

The earliest statistical methods placed an isotropic covariance around every point in the first set of points and then registered the second set of points to it under an MLE framework (MPM [14], EM-ICP [43], CPD [98, 99]). More modern statistical approaches utilize a generative model framework, where a GMM is usually constructed from the points explicitly and registration is solved in a MLE sense using an EM or ECM [90] algorithm (ECMPR [51], JRMPC [33]), though some utilize a max correlation approach (Kernel Correlation [142], GMM-Reg [59][57]). Since a statistical framework for point cloud registration tends to be more heavyweight than ICP, some have tried methods such as decimation (EM-ICP [43]) or voxelization (NDT methods [4, 6, 133, 134]) to create smaller or more efficient models, while others have devised ways to exploit point-level parallelism and GPU-computation for increased computational tractability and speed [21, 136].

In all cases, these statistical techniques share the same underlying theoretical assumption that points are *iid* samples of a probability distribution, and thus suffer from the same types of failure cases when scenes being registered only partially overlap.

Though not the focus of this thesis, it is important to note the work done on global registration. Unlike ICP and generative methods, which are by definition local methods (meaning that their performance degrades as a function of the amount of pose displacement), global methods try to find a set of pose-invariant features in each point cloud upon which to perform globally invariant matching [42, 82, 119, 157]. Similar to various common approaches using 2D image features (e.g. SIFT [81]), if a feature can be matched regardless of the sensor pose with high sensitivity and then combined with a highly specific pruning process (e.g. some type of RANSAC-based approach) or robust matching scheme (e.g. histogram correlation), one could then feasibly register point clouds under high amounts of occlusion. Ma *et al.* [82] address the problem of small overlapping regions by performing correlative histogram matching on the set of normals between two point clouds. Rusu *et al.* define point-angle features around k-neighborhoods and use sample consensus. Zhou *et al.* extend Rusu's work using robust distance functions that discount spurious matches instead of sample consensus [157].

Though global registration algorithms are promising, we choose to utilize a local, generative model approach in this work, as these global algorithms tend to be limited in applicability. First, these algorithms rely on feature repeatability, which is highly dependent on an ability to construct well-defined normals or point-angle properties. As past research has demonstrated, accurately calculating normals can be difficult due to sparsity, noise, and varying support regions, especially with LIDAR data [72, 145]. Me *et al.* further require scenes where occlusions or non-overlapping regions still nevertheless share segments of the same planar surfaces (i.e. structured indoor environments with minimal clutter). Third, feature computation tends to be very expensive since many different point neighborhoods often need to be considered (and potentially at multiple scales) and then pruned for saliency [157]. Other global techniques, like Go-ICP [154], rely on expensive branch-and-bound techniques that make it impractical for real-time usage. Thus, these algorithms are most suitable for highly rectilinear, structured environments where point density is high and real-time operation is not required.

7.3 Approach and Methodology

For standard GMM-based MLE registration methods, the probability distribution given by the model predicts the likelihood of a point in space, and thus with an *iid* set of samples, the model can estimate the likelihood of a set of points being included in a particular point cloud. In the case of algorithms like REM-Seg (4.3), MLMD (4.4) and JRMPC [33], for example, the GMM is produced by optimizing the data likelihood of the original set of points in the point cloud over a set of possible GMMs at some fixed level of fidelity, so that it therefore is the most likely GMM for describing the original PCD distribution.

Note then that the model parameters are coupled to the viewpoint at which the sensor data was captured. That is, the generating probability of a point in space given a point cloud model is valid only for the particular sensor view under which the model was first created. Thus, the registration process itself directly breaks its own basic theoretical assumptions every time it optimizes for a new sensor pose, and this degrades the accuracy and convergence of such methods.

7.3.1 Ω-Estimation

There are multiple ways in which the *iid* assumption behind GMM-based MLE registration methods are violated, but in this work we focus on one important source: non-overlapping geometry resulting from field-of-view differences.

A moving sensor with limited field of view will scan a different slice of the world each instant. Non-overlapping portions of geometry occur when a range sensor moves such that previously unseen geometry comes into view and/or previously seen geometry leaves the view. The assumption violation for GMM-based MLE methods occurs in these situations because non-overlapping portions of geometry produce point samples that occur in one point cloud but not the other.

The basic Maximum Likelihood Estimation (MLE) criterion most used by GMM/EM methods assumes both point sets are *iid* samples of a single generative model, and that one point set has undergone some latent spatial transformation. Thus, given two point sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$, the most likely estimate of the transformation T is the estimate that maximizes the data probability that the samples of the transformed point cloud $T(\mathcal{Z}_2)$ came from some probabilistic representation of spatial likelihood (generative model parameterized by $\hat{\mathbf{\Theta}}$) and derived from the spatial distribution of the first point cloud $\mathcal{Z}_1$,

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, p(T(\mathcal{Z}_2)|\hat{\mathbf{\Theta}}_{\mathcal{Z}_1}) \tag{7.1}$$

Since the points are assumed to be *iid* samples of $\hat{\mathbf{\Theta}}$, the data likelihood further decomposes into a product over all points in $\mathcal{Z}_2 = \{\mathbf{z}_i\}, i = 1..N_2$,

$$\hat{T} = \underset{T}{\operatorname{argmax}} \prod_{i}^{N_2} p(T(\mathbf{z}_i)|\hat{\mathbf{\Theta}}_{\mathcal{Z}_1}) \tag{7.2}$$

By adopting this framework, we assume that there is a latent probabilistic function controlling the generation of points, and that the specific form of $p(z|\hat{\mathbf{\Theta}})$ dictates individual point

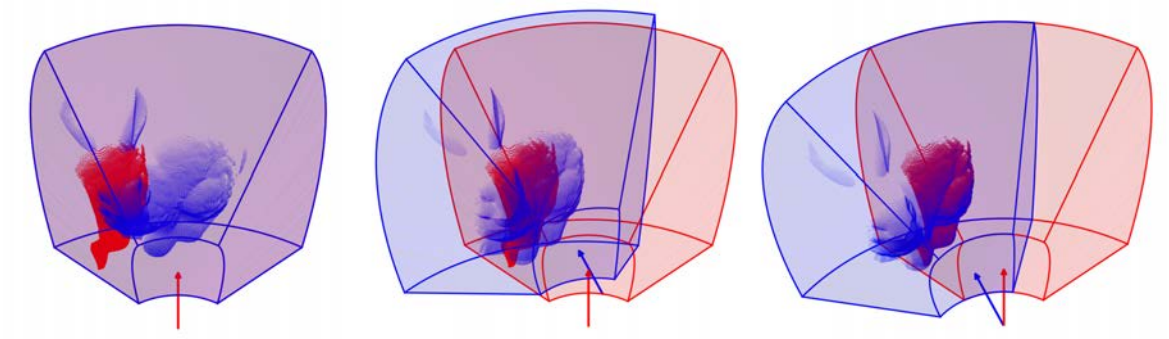


Figure 7.2: **Expected Overlap Estimation.** A range sensor from two different viewpoints (A and B) has two different fields-of-view (Red and Blue). Building the sensor view into the registration process allows robust estimation of overlapping regions.

probabilities of 3D space. The most common form for parametrizing this probability distribution is through a GMM, whose data probability is defined as a convex combination of J Gaussians weighted by the vector π,

$$p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_{j=1}^{J} \pi_j \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j) \tag{7.3}$$

Specifically, points from geometry present in *non-overlapping view regions* of each point set make the current MLE construction ill-posed. Since a probability distribution describing each point set will have fundamentally different spatial extents, these view differences make it impossible to represent both point sets, $\mathcal{Z}_1$ and $\mathcal{Z}_2$, as the *iid* sampling of the same latent model. This limits these algorithms' applicability in cases where large transformations, limited field-of-view, or occlusions between point sets cause large degrees of non-overlapping geometry.

See Fig 7.2 for a depiction of this violation. Let A and B be the views from which $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are captured, respectively. Under the traditional maximum data likelihood paradigm, interpreting $\mathcal{Z}_2$ as a set of rigidly transformed *iid* samples of the model constructed from $\mathcal{Z}_1$ ($\mathbf{\Theta}_{\mathcal{Z}_1}$) is problematic since $\mathbf{\Theta}_{\mathcal{Z}_1}$ only "knows" about the space within region A. If $B \cap \neg A \neq \emptyset$, then any $\mathbf{z}_i \in \mathcal{Z}_2$ inside this region will be unexplainable by ($\mathbf{\Theta}_{\mathcal{Z}_1}$). Similarly, if $A \cap \neg B \neq \emptyset$ and $\int_{A \cap \neg B} p(z|\mathbf{\Theta}_{\mathcal{Z}_1}) dz > 0$ then $\mathcal{Z}_2$ will become more and more unlikely to be considered an *iid* sampling from the model described by $\mathbf{\Theta}_{\mathcal{Z}_1}$ as the cardinality of $\mathcal{Z}_2$ increases, since there will always be a portion of the model ($A \cap \neg B$) from which samples never appear. Even in the case of a symmetric global model where Θ is derived from both $\mathcal{Z}_1$ and $\mathcal{Z}_2$, as in JRMPC [33], it is still the case that no single point cloud can be considered an *iid* sampling of the global model if $A \neq B$, and thus non-overlapping point data may be erroneously matched during registration.

7.3.2 Expected Overlap Estimation

In order to begin to solve the problems discussed in the previous section, we start with a goal of being able to incorporate and leverage known sensor view constraints into the registration process. Since we usually know the field-of-view properties of the sensor *a priori* (i.e. minimum and maximum range distance, field-of-view angles, etc), if we knew the relative pose change between

A and B, we could also calculate where the overlapping regions occur. Figure 7.2 illustrates this idea using a partially non-overlapping Stanford Bunny. Denoting Ω as our overlapping region, we can then augment the registration problem as follows:

$$\hat{T} = \underset{T}{\operatorname{argmax}}\, \Omega^*(T(\mathcal{Z}_2)|\Theta_1), \forall Z_2 \in \Omega \tag{7.4}$$

Where $\Omega^*(\cdot)$ denotes the functional,

$$\Omega^*(p) = \begin{cases} \eta p(z), & \text{if } z \in \Omega \\ 0, & \text{otherwise} \end{cases} \tag{7.5}$$

where η is a normalization constant to enforce $p(z)$ summing to 1 over all Ω.

This formulation better conforms to the *iid* assumption underlying statistical point cloud models in that it defines registration to occur only over previously seen regions of space. Equation 7.4 maximizes the probability that the points in $\mathcal{Z}_2$ occurring in regions of previously seen geometry were generated by a probabilistic model of that same region constructed from $\mathcal{Z}_1$. We therefore avoid the *iid* violations resulting from the normal MLE construction, where new points in $\mathcal{Z}_2$ from previously unseen regions must still be interpreted as *iid* samples from the existing model, and where regions of $\mathbf{\Theta}_{\mathcal{Z}_1}$ that are not in the current field-of-view of $\mathcal{Z}_2$ still contribute to the spatial probability distribution function $\mathbf{\Theta}_{\mathcal{Z}_1}$ from which $\mathcal{Z}_2$ is supposed to be a set of *iid* samples.

Of course, the problem here is obvious: if we knew the overlapping region between A and B (and thus the relative pose change between A and B), we wouldn't need to be registering A and B together in the first place: the relative pose change between A and B is exactly what we are trying to find when we perform registration.

Fortunately, this formulation does present us with an iterative EM-like approach for imposing a general view-awareness for any asymmetric registration algorithm. That is, if we impose a probability function over the estimation of Ω using known sensor view properties Ψ and relative transformation T, we can iteratively hold Ω fixed to estimate T and then hold T fixed to re-estimate Ω (and also therefore the functional Ω^*).

Algorithm 5 Expected Overlap Estimation (EOE)

$$\textbf{Step 1: } \hat{\mathbf{\Theta}}_{\mathcal{Z}_1} = \underset{\mathbf{\Theta}}{\operatorname{argmax}}\, p(\mathcal{Z}_1|\mathbf{\Theta}) \tag{7.6}$$

$$\textbf{Step 2: } \hat{T} = \underset{T}{\operatorname{argmax}}\, \hat{\Omega}^*(T(\mathcal{Z}_2)|\hat{\mathbf{\Theta}}_{\mathcal{Z}_1}), \forall Z_2 \in \hat{\Omega} \tag{7.7}$$

$$\textbf{Step 3: } \hat{\Omega} = \underset{\Omega}{\operatorname{argmax}}\, p(\Omega|\hat{T}, \Psi) \tag{7.8}$$

$$\textbf{Step 4: } \text{If not converged, go to Step 2.} \tag{7.9}$$

Implementation

Algorithm 5 shows each step of EOE algorithm. Note that the first two steps can be replaced with any standard registration algorithm. The details of each step are described as follows:

Step 1: As in Chapter 4, we adapt the *Mixture Decoupling* algorithm from the model described in Chapter 3. Note that for Step 1 we could have chosen to use an NDT construction [6, 133, 134] for further gains in speed (though with a loss in model fidelity), or alternately by simply placing covariances around every point (such a construction is $O(1)$ but will produce a much slower Step 2).
Step 2: The basic optimization underlying Step 2 has been solved in many different ways, each with their own trade-offs. Many methods rely on simplifications to the structure of $\mathbf{\Theta}$, including: imposing the same covariance to every mixture, restricting covariances to be isotropic, or equal mixture weighting so that a closed form solution may be applied (typically Horn's method [52]). ICP, EM-ICP, CPD, and JRMPC can be considered examples of this. Other approaches put less restrictions on the structure of $\mathbf{\Theta}$, but using gradient descent [24] in lieu of a closed form solution. Others have approximated the optimization criterion itself to a more tractable form, for example, by using direct simplification [21] or SDP relaxation [51]. We follow the latter strategy, introducing a highly accurate Mahalanobis distance approximation. To provide view-awareness, we repeatedly apply $\Omega^*(p)$ and cull from $\mathcal{Z}_2$ those points that do not fall within the current estimate of Ω.
Step 3: For Step 3, we adopt a simple binary likelihood approach. Given certain simple sensor intrinsics Ψ that can be inspected *a priori* (e.g. min and max range distance and field-of-view angle), we recalculate Ω as the intersection of both field-of-view regions using the current MLE guess for T.
Step 4: Convergence is defined by a set number of max iterations, but with an early stopping condition if ΔT falls below a threshold.

Remarks on Efficiency

Over traditional GMM-based registration algorithms, we use the functional $\Omega^*(\cdot)$ instead of $p(z|\mathbf{\Theta})$ and also cull points from $\mathcal{Z}_2$ that fall outside Ω. Note that $\mathbf{\Theta}_{\mathcal{Z}_1}$ needs to be calculated only once, however, since $\operatorname{argmax}_{\mathbf{\Theta}} p(\mathcal{Z}_1|\mathbf{\Theta})$ does not depend on either T or Ω.

By representing Ω simply as the intersection of the field-of-view extents for the sensor (cf. Figure 7.2), the additional operations in Steps 2 and 3 add insignificant overhead, as each point requires only a single constant-time boundary check. In fact, in some implementations these checks may speed up the base algorithm since points outside Ω need not be considered, resulting in a smaller sized registration problem overall.

7.4 Evaluation

7.4.1 Robustness to Partially Overlapping Views

To measure robustness to non-overlapping regions, we created five partially overlapping point clouds of the Stanford Bunny. These views were created by casting 512x512 rays outward with a viewing angle of 60 degrees at five different poses. We registered together the point clouds produced from these depth map outputs, frame-by-frame, in sequence. To visually compare the results, we placed all five frames into a global coordinate system by compounding successive

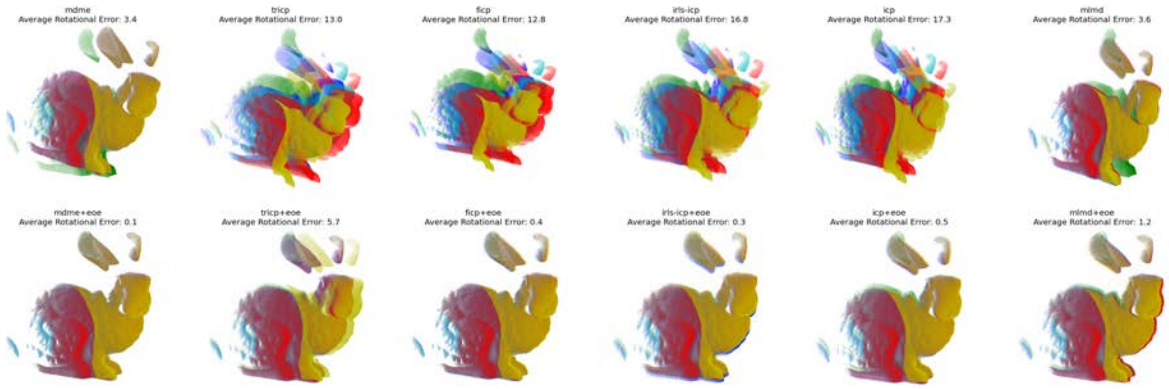


Figure 7.3: **Partially Overlapping Bunnies** Five partially overlapping depthmaps of the Stanford Bunny were registered together in sequence and then placed into a global coordinate system. Each color refers to points sets from different views. The top row consists of algorithms run normally, and the bottom row shows the performance of those same algorithms augmented with EOE. Though MDME+EOE (bottom-left) has the lowest error, the addition of EOE generally improves all algorithms.

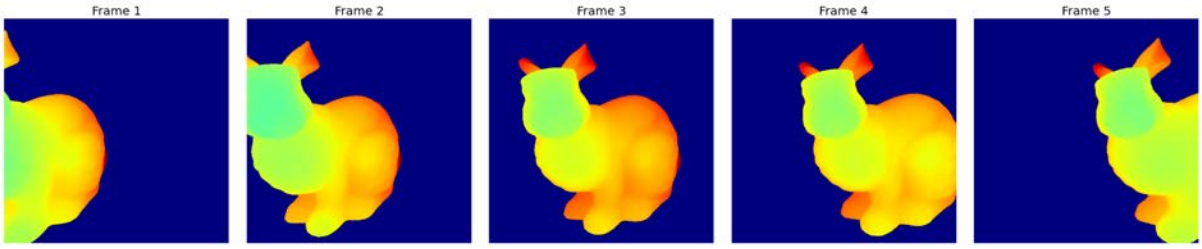


Figure 7.4: Input depth stream used for bunny overlap experiments.

poses. Refer to Fig. 7.3 for the results of this test with comparisons against ICP [11], Trimmed ICP (TrICP [12], MLMD [21], FICP [113], and Iteratively Reweighted Least Squares ICP (IRLS-ICP) [50]. We chose ICP since it is still widely used in this domain, MLMD because it represents a current state-of-the-art generative method, and TrICP, FICP, and IRLS-ICP as they represent state-of-the-art methods for robust outlier handling. Note that for purposes of generality, we have disincluded methods like projective ICP that require dense 2D depth maps [116], which are not always available depending on the range sensor. All our tested algorithms operate over unstructured point cloud data for all our experiments.

Shown in Figure 7.3, our proposed method with overlap estimation (MDME+EOE) works well while other methods suffer from serious registration misalignment. Note that EOE should be considered as a standard augmentation of previously existing registration methods to improve their performance.

7.4.2 Kinect Data

In this test, we calculate the frame-to-frame accuracy on the Stanford Lounge dataset (depicted in Figure 7.6), which consists of range data produced by moving a handheld Kinect around

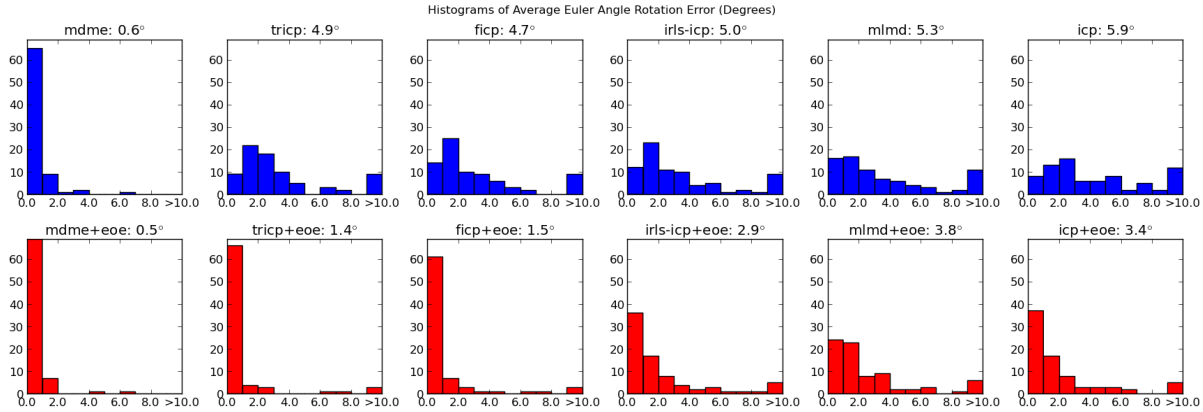


Figure 7.5: **Lounge Error Distributions** Histograms of frame-to-frame error as measured by average Euler angular deviation from ground truth, for the Lounge dataset. The top row show the results of algorithms without EOE and the bottom row includes EOE. EOE improves the overall accuracy of all algorithms tested.

Table 7.1: Comparison to other methods.

Method	Avg. Error (degrees)	Avg. Time(s)
ICP [3]	5.9	1.32
MLMD [21]	5.3	0.49
FICP [113]	4.7	0.30
IRLS-ICP [50]	5.0	0.29
TrICP [12]	4.9	0.29
MDME	0.6	0.24
MDME+EOE	**0.5**	**0.23**

an indoor environment. We register together every 5th frame for the first 400 frames, each downsampled to 5000 points. To measure the resulting error, we calculate the average Euler angle deviation from ground truth. For EOE, we set the horizontal field-of-view to 56.5 and the vertical field-of-view to 43.6 degrees. Our mixture model was statically set to 512 clusters. All our experiments were run on a Intel Q6600 and NVIDIA GTX670. The average computation time for each registration is given in Table 7.1. Note that EOE actually *decreased* the running time: each iteration takes slightly longer but it takes fewer iterations to converge.

7.4.3 Velodyne LIDAR Data

Though the Velodyne VLP-16 LIDAR has a 360 degree horizontal field-of-view, it suffers from a relatively narrow vertical field-of-view of 30 degrees. Similarly, the angular resolution of the pitch and yaw axes vary dramatically. These properties lead to the formation of dense rings of sampled geometry around the sensor, which often will degrade the accuracy of point-to-point matching algorithms. Figure 7.7 shows a depiction of the view model used in conjunction with EOE.

For our field tests, we placed a Velodyne VLP-16 on a moving platform and collected a

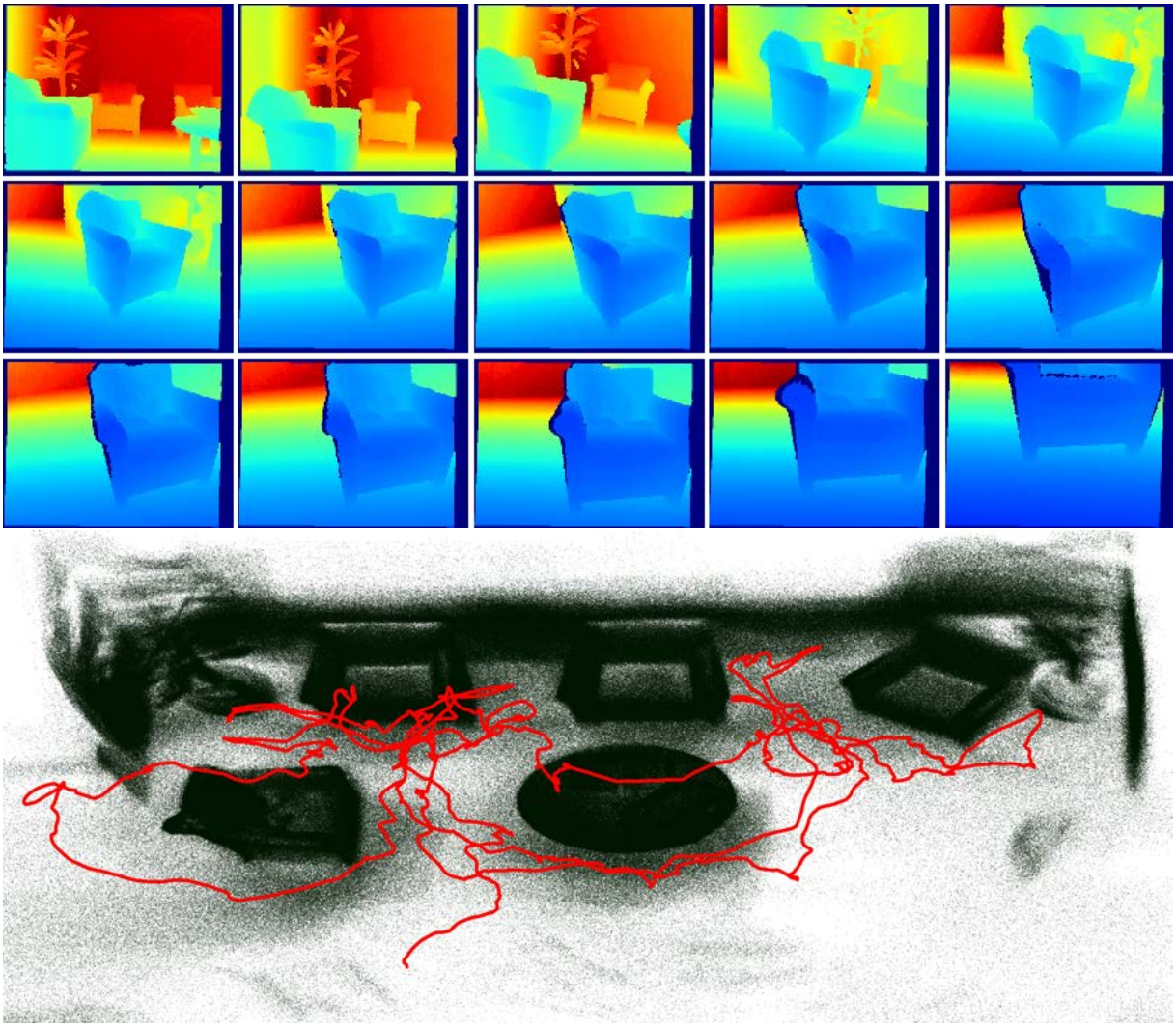


Figure 7.6: Lounge data for additional experiments. **Top**: Sample depth map stream. **Bottom**: Ground truth pose used for accuracy calculations.

single sweep of data (roughly 0.10 seconds worth) every 3 seconds inside a large factory-like setting (NREC High Bay). We first compare MDME+EOE with Trimmed ICP. Registering each sweep successively into a global frame gives us the results in Figure 7.9. Trimmed ICP, even with robust outlier detection, mistakenly tries to align the circular sampling pattern on the floor instead of real geometry. MDME with EOE enabled is able to excise spurious floor points by using the Velodyne's relatively narrow vertical field-of-view (30 degrees). Refer to Figure 7.8 for a close-up view of this situation.

The qualitative results of running the rest of the algorithms tested over the Velodyne can be seen in Figures 7.10 and 7.11. Through visual inspection, one can see how most algorithms when not using EOE tend to underestimate pose change. This again is due to the fact that these extremely uneven LIDAR sampling patterns break the normal assumptions behind traditional registration methods. Thus, without an understanding of the sensor sampling properties, these

algorithms are unable to properly capture the sensor pose change. Although MDME performs the best out of all algorithms tested, EOE is still necessary in order to produce an error-free registration over all frames.

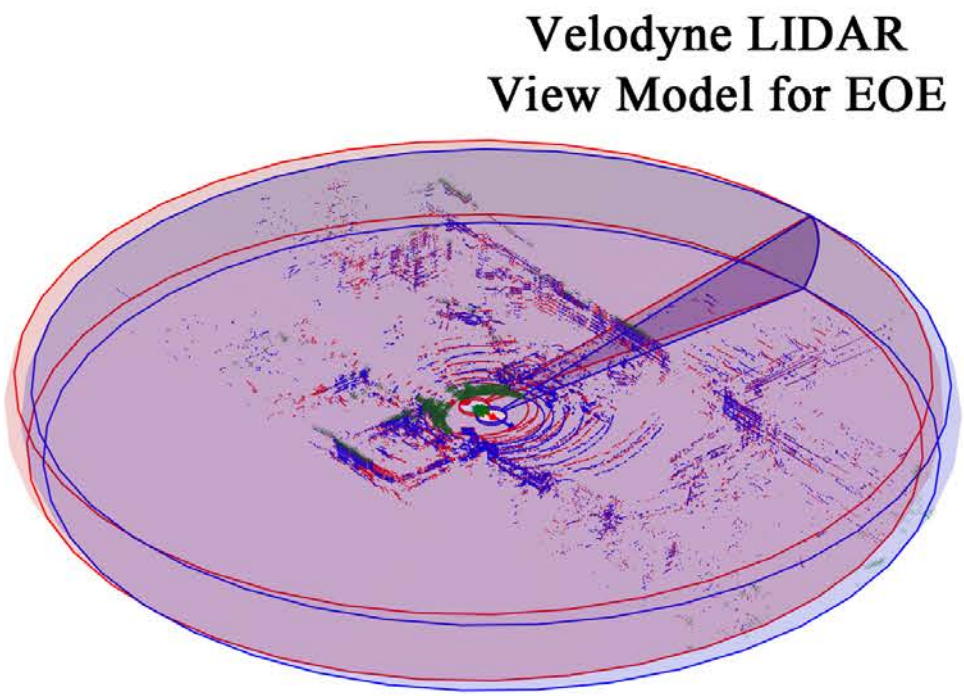


Figure 7.7: **Velodyne View Model**: Though the Velodyne VLP-16 LIDAR has a 360 degree horizontal field-of-view, it suffers from a relatively narrow vertical field-of-view of 30 degrees. Similarly, the angular resolution of the pitch and yaw axes vary dramatically. These properties lead to the formation of dense rings of sampled geometry around the sensor, which often will degrade the accuracy of point-to-point matching algorithms. Expected Overlap Estimation (EOE) utilizes the knowledge of this narrow vertical field-of-view by automatically culling points belonging to dense rings that can no longer be seen (shown in green).

7.5 GMM Fusion and Mapping: A Unified Pipeline

So far in this chapter, we have established a very efficient robust frame-to-frame modeling and registration procedure (MDME+EOE) that operates well even under challenging real-world conditions. To further demonstrate its effectiveness in real-world situations, we offer a complete pipeline for large-scale mapping over time. We would like to note that the methods in this section are not meant to compete with state-of-the-art SLAM systems, but rather provide a proof-of-concept of how the statistical models presented in this thesis may naturally be leveraged in a large-scale mapping system. That is, we offer these results as further validation of the efficacy of the segmentation and registration techniques discussed in this and previous chapters and not as a standalone SLAM system.

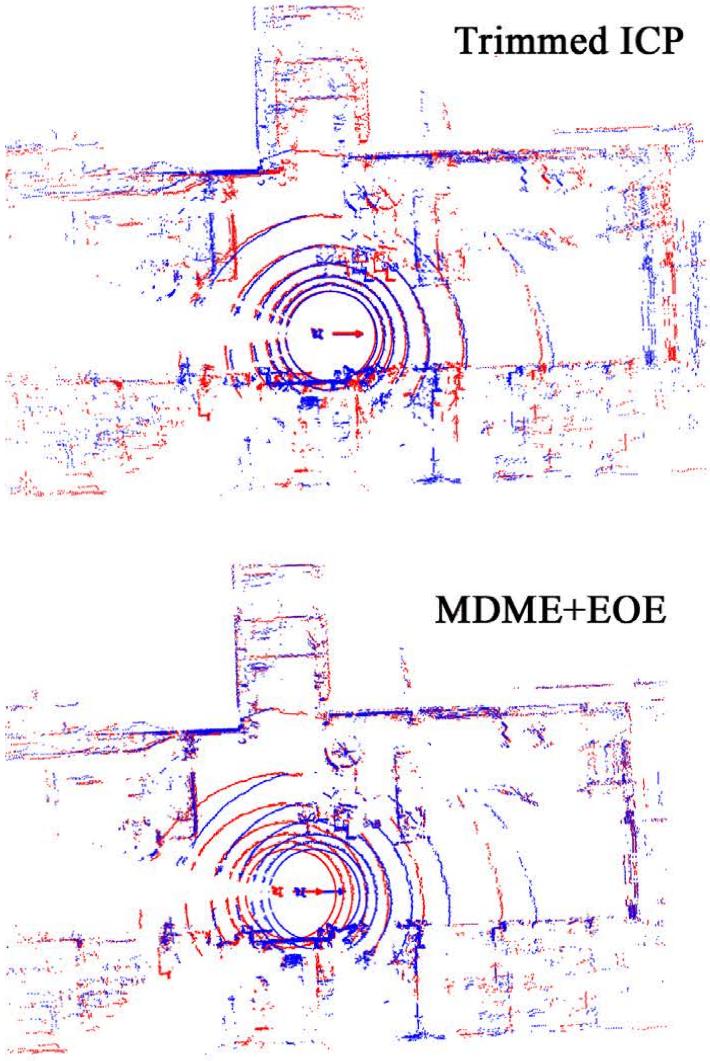


Figure 7.8: **Sample Problematic Registration**: Here we show one problematic example of 2 frames of data (red and blue) being registered together. This is the final registered output of both Trimmed ICP and our proposed method, MDME+EOE. The dense concentric rings of points serve to degrade the robust ICP-based Trimmed ICP algorithm (top). Given that these rings of points are returns from flat ground, their appearance will not change very dramatically with any pose change on the x, y, or yaw axes. Thus, a point-to-point registration algorithm is likely to produce a no-movement result as it will try to maximally overlap the rings of points. The result on the bottom shows a correctly registered result: note that the rings are not overlapped.

Fusion and Mapping Pipeline: Given successive frames of LIDAR data, our mapping pipeline operates as follows:

1. Perform frame-to-frame registrations for some N frames of data
2. Aggregate transforms to put each frame in a common global frame

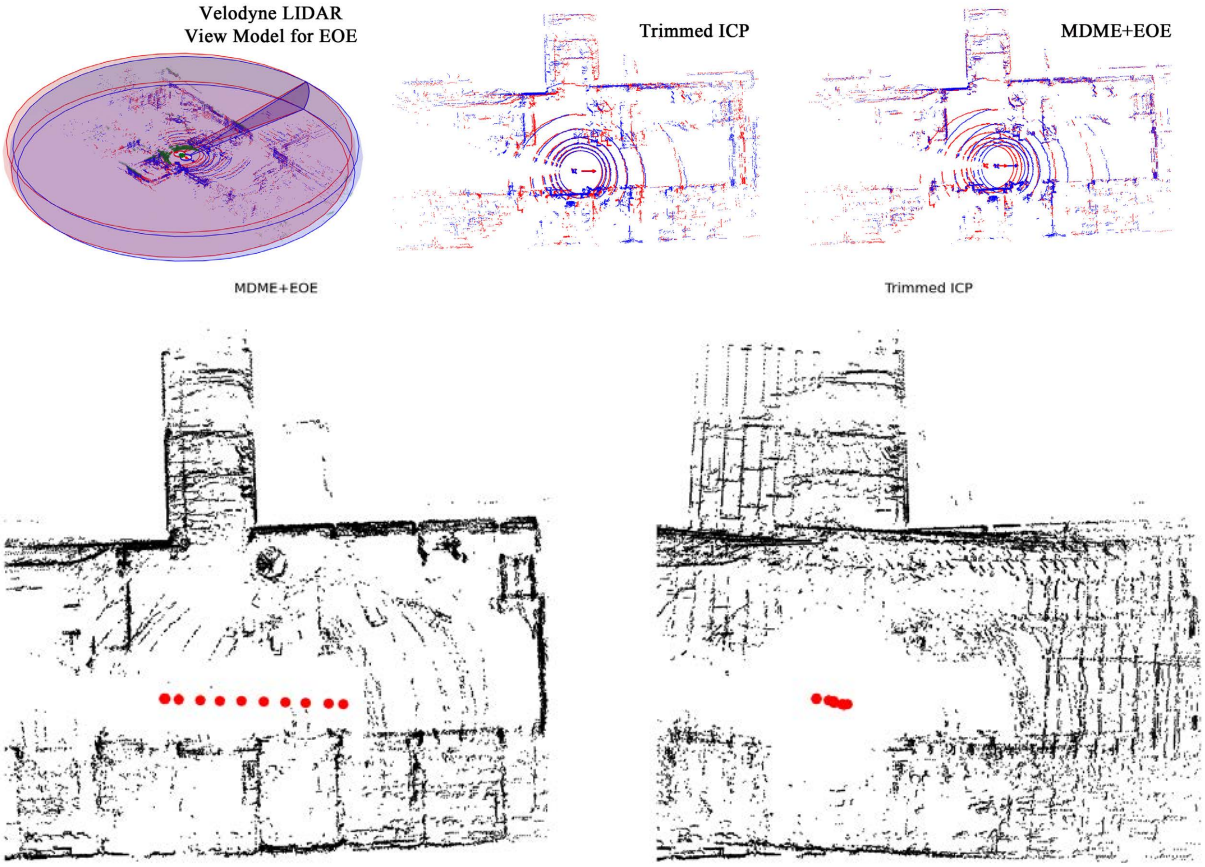


Figure 7.9: **Top Left**: Velodyne view model of 360 degrees horizontal and 30 degrees vertical field-of-view. **Top Right**: A single registration result showing how TrICP tries to match the circular sampling patterns, while MDME+EOE correctly matches real overlapping geometry. **Bottom**: The LIDAR data in global frame. The smeared wall pattern on the right shows TrICP's inability to register this type of LIDAR data.

3. In a separate thread, randomly pick pairs of point clouds within some proximity to each other in global coordinate space and try to register them
4. For each randomly picked pair, look at the log-likelihood of the MLE registration result and if above a threshold (indicating a correctly registered result), denote as a loop closure
5. Add each loop closure as soft constraint to a pose graph
6. In a separate thread, optimize pose graph (we used the open source *Toro* software [45])

To test our pipeline, we used a large urban outdoor dataset of VLP-16 LIDAR data frames covering roughly an area of 400x400 meters (2200 frames total). Our open-loop result is shown in Figure 7.12. The ground truth is shown by the red dots. The above trajectory was calculated faster than sensor rate using a Velodyne VLP-16 sensor (10 Hz) on a GTX 660. The registration is performed in 3D, though the view is overhead, with ground points removed for clarity. As a comparison, we also included the best result from a state-of-the-art robust point-to-plane ICP algorithm (*libpointmatcher*), which runs over 5 times more slowly. The quantitative error shown

in the figure is the average frame-to-frame rotation angle error measured against a highly accurate INS+GPS system. Even without looking at the average error (0.152 for ICP vs 0.096 for our method), one can see qualitatively that our system is much better able to provide long-term drift-free estimates for mapping in a global frame.

Our loop closure detection system is a concurrent but asynchronous process to our frame-to-frame segmentation and registration system. It loops by randomly testing pairs of point clouds within a thresholded proximity. If the log-likelihood of the final registration result is above a threshold, we add a weighted edge between the poses attached to this pair of point clouds. This process thereby slowly builds up a weighted pose graph. A depiction of this pose graph on our urban dataset can be see in Figure 7.13. We have highlighted certain areas of drift with two red ellipses. The pose graph optimization library used is a Tree-Based Network Optimized (*toro* [45]). After a few iterations of the graph optimizer, we can correctly recover from drift that occurs over time.

7.6 Summary

In this chapter, we derived a new point cloud registration algorithm coupled with a general robust outlier detection method by looking at how real-world conditions cause the underlying assumptions of the state-of-the-art generative model based registration algorithms to fail. We showed that the concept of view-awareness can be applied to any registration process to mitigate these common failure conditions and provide more robust performance under real-world conditions of limited field-of-view, non-overlapping regions, and uneven sampling density. MDME and EOE work well on both Kinect data and LIDAR data, and EOE is straightforward to add to any existing registration algorithm. Finally, we applied these techniques along with hierarchical segmentation to a mapping system. To detect loop closures, we utilized the point-to-model log likelihood registration result and then simply applied an off-the-shelf graph optimization package (Toro [45]) to refine our pose estimates over time. These experiments show that these statistical techniques provide a natural fit for various complex tasks related to mapping and registration.

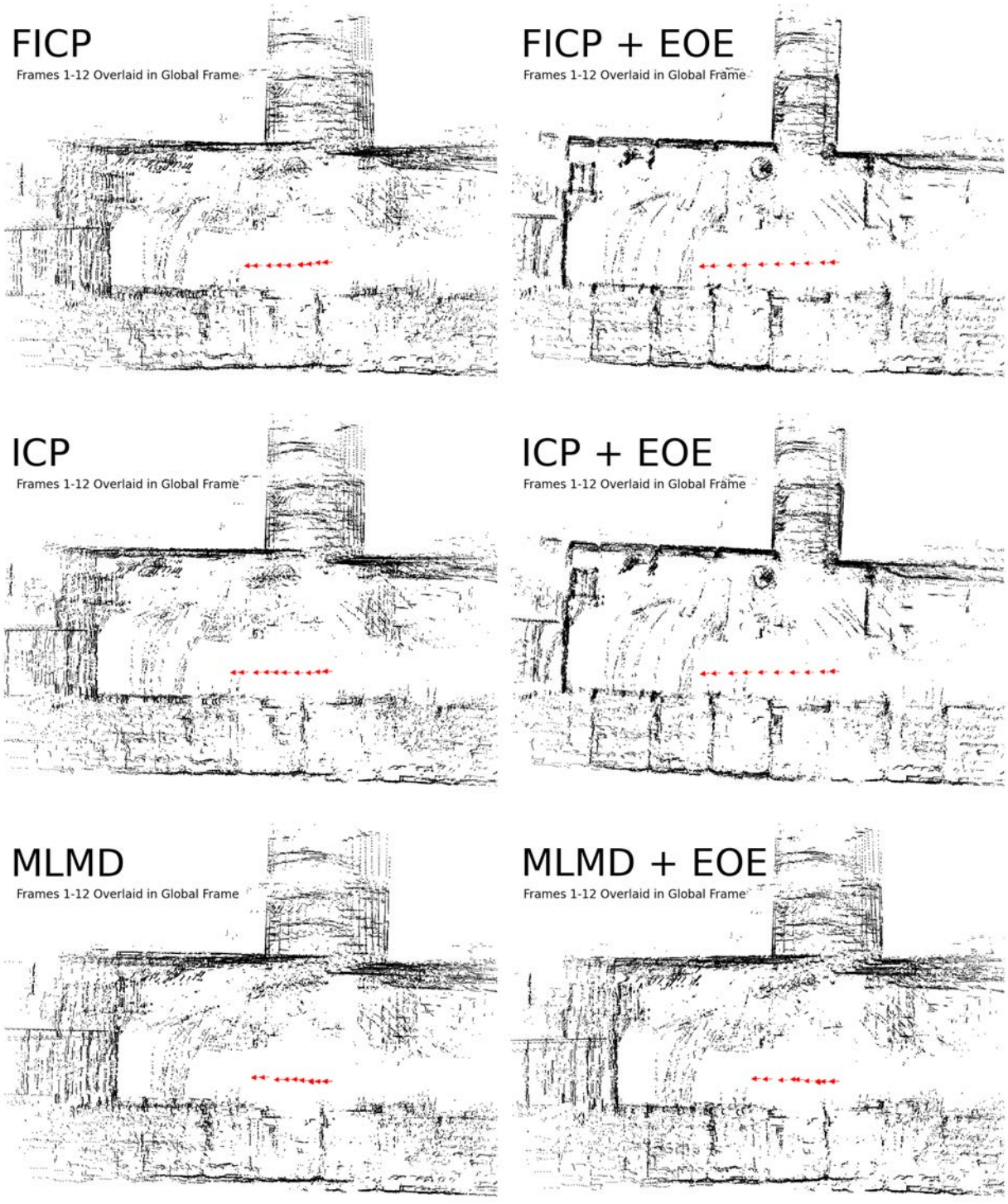


Figure 7.10: **FICP, ICP, MLMD**: This is a top-down view of twelve frames of LIDAR data that were registered frame-to-frame and then plotted in a global coordinate system. The ground points have been removed for clarity. Look to the northern corridor for the clearest signs of registration error. ICP+EOE and FICP+EOE produce the best results.

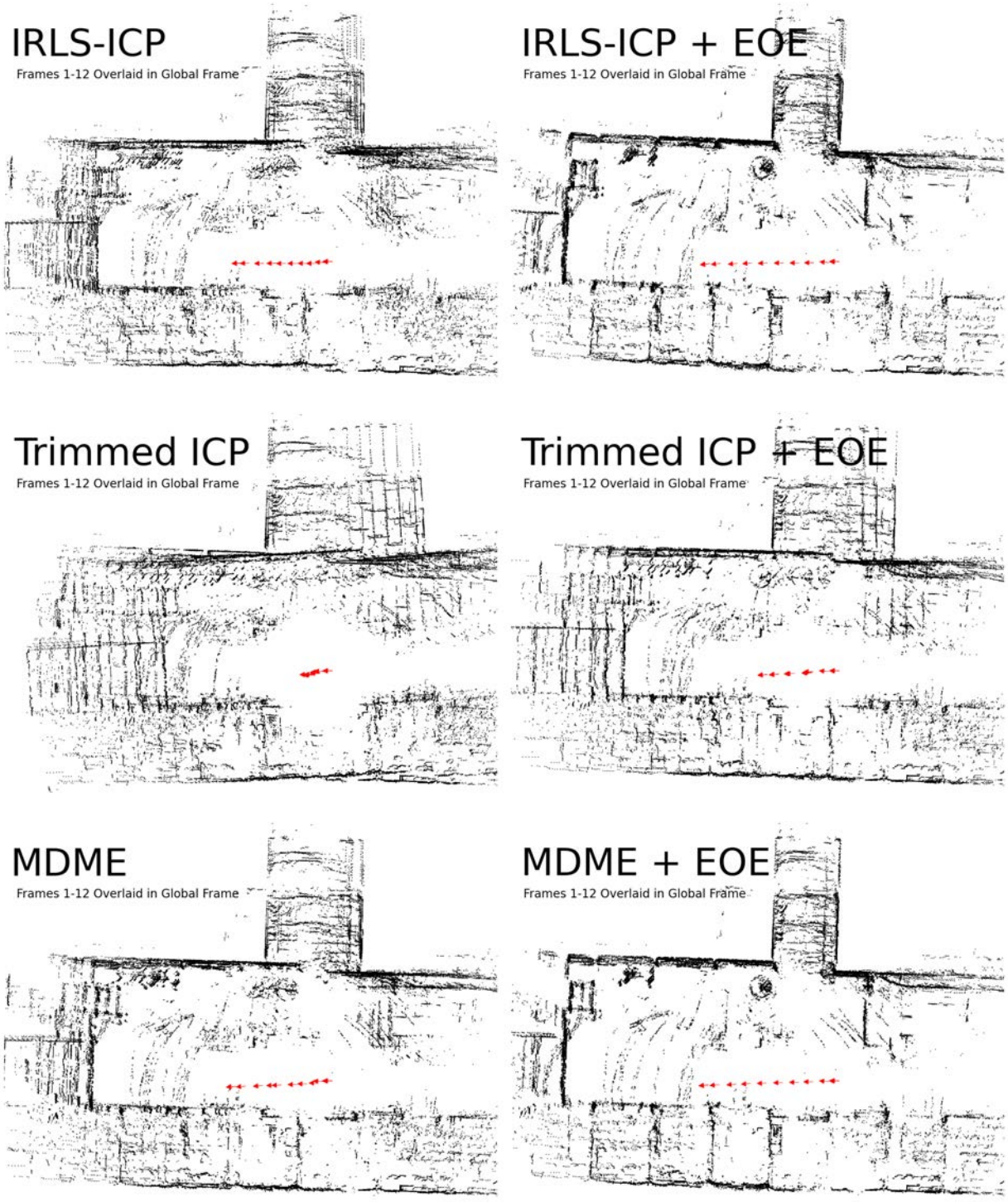
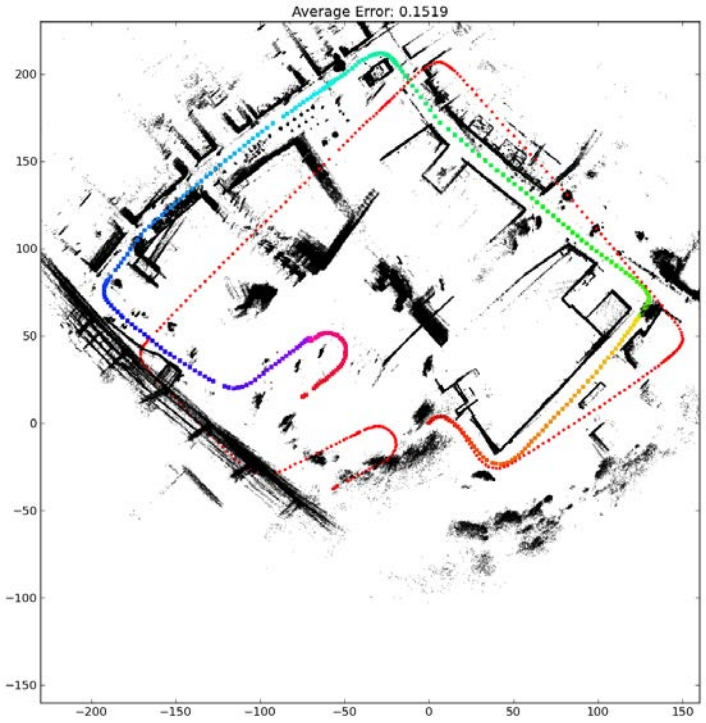


Figure 7.11: **IRLS-ICP, Trimmed ICP, MDME (our proposed method)**: This is a top-down view of twelve frames of LIDAR data that were registered frame-to-frame and then plotted in a global coordinate system. The ground points have been removed for clarity. Look to the northern corridor for the clearest signs of registration error. Note the severe inadequacy of Trimmed ICP, which produces too little movement. Our proposed method without EOE does fairly well, but performs best with the addition of EOE.

State-of-the-Art Robust Point-to-Plane ICP Algorithm (*libpointmatcher*)



Our algorithm (MDME+EOE)

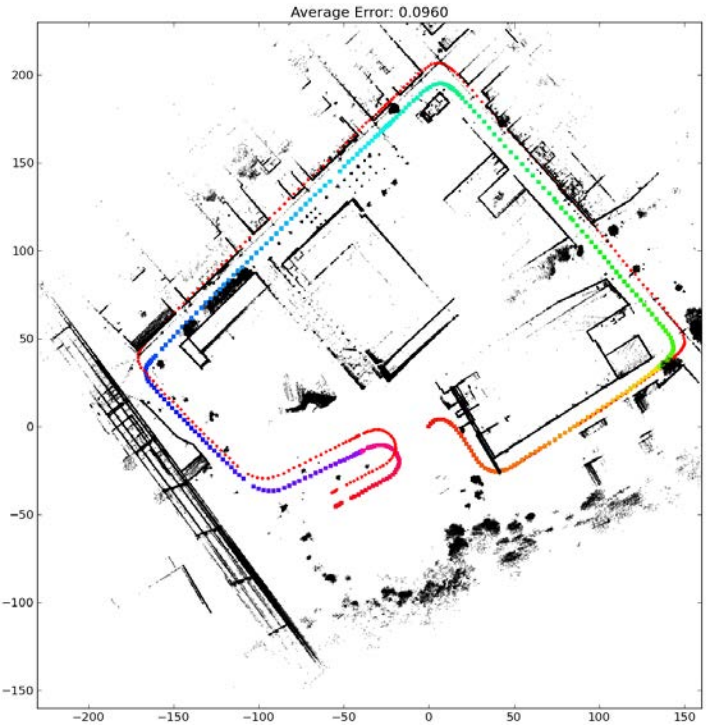


Figure 7.12: Frame-to-frame mapping over many frames in an outdoor urban environment. Units are in meters. The calculated path is the multicolor trajectory. The red dotted path is ground truth. We compared the best performing algorithm in the *libpointmatcher* library without EOE (a point-to-plane ICP with robust outlier rejection using a trimmed distance outlier filter) against MDME+EOE (the best performing algorithm with EOE). Note the increased drift of point-to-plane ICP when compared to our method, especially when turning around corners.

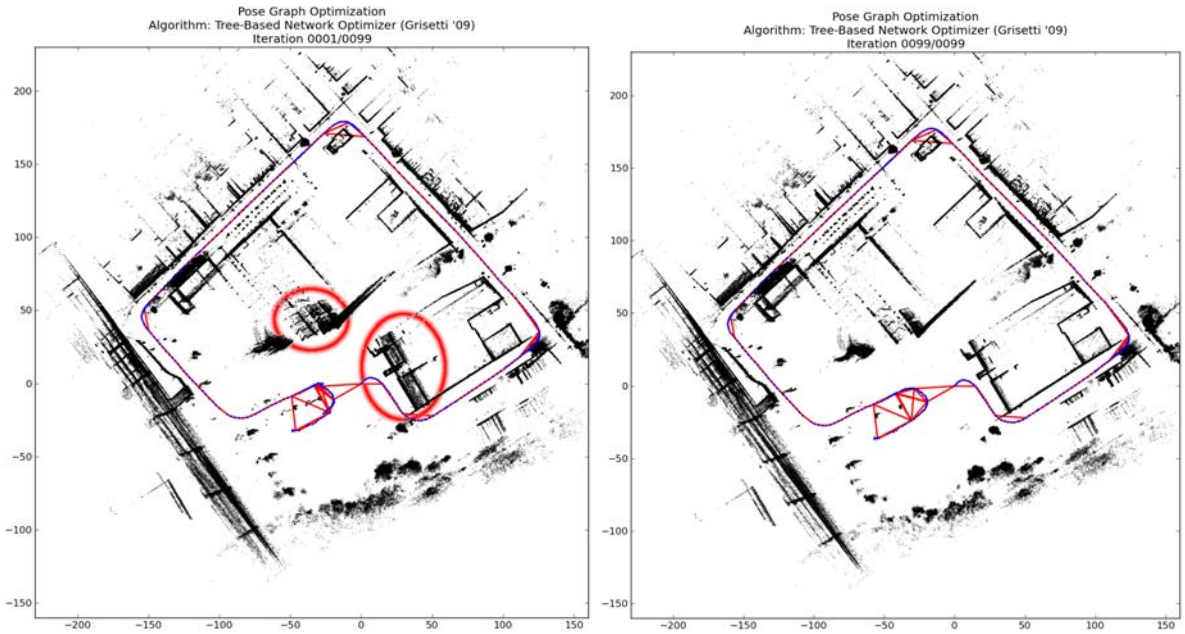


Figure 7.13: **Left** Pose graph without yet performing optimization. Edges found during our matching process are represented by red lines, and poses by blue dots. We have circled two areas of accumulated drift with red ellipses. **Right** Graph optimization has provided pose corrections to remove the small amount of drift that accumulated from driving around a city block.

Chapter 8

Conclusion

The choice of sensor data representation has a direct consequence on the effectiveness of any robotic perception application. A proper data representation functions as an intermediary between the often noisy and unorganized measurements of the world and a high-level scene understanding, aiding in such tasks as localization, tracking, and object recognition, which in turn are necessary for other higher level perception and intelligence tasks such as planning and control.

In this thesis, we have established techniques for accelerated construction of a family of hierarchical generative models for point cloud data. We discussed various registration methods which closely tie model creation to an efficient registration algorithm, beating current state-of-the-art. We then showed how to develop a hierarchical variant of the generative models used for registration along with many experiments showing superior size/fidelity characteristics vs state-of-the-art. We then introduced two methods for inference and visualization based on stochastic sampling of the model: dynamic occupancy grid creation, and isocontour visualization with a modified sparse marching cubes algorithm. We have implementations of these for both PC and embedded systems. Finally, we augmented the model to include known sensor properties and used this to form the basis of a new type of registration algorithm that can robustly register together real-world LIDAR data even when sampling sparsity and non-overlapping geometry would otherwise prevent existing algorithms from being applicable.

Compact generative models of point cloud data provide a new paradigm for low level 3D point processing. As we've demonstrated in this thesis, through a robust yet compact statistical modeling of point data, these models can not only facilitate existing spatial processing applications, but they also enable fundamentally new possibilities for the unification of many related perception processes, included segmentation, sampling, inference, visualization, and 3D mapping. We hope our work finds widespread applicability as a general statistical point cloud processing paradigm that could benefit many 3D point processing applications in the field of mobile robotics and 3D perception.

Bibliography

[1] Sven Bachthaler and Daniel Weiskopf. Efficient and adaptive rendering of 2-D continuous scatterplots. *Computer Graphics Forum*, 28(3):743–750, 2009. ISSN 01677055. doi: 10.1111/j.1467-8659.2009.01478.x. URL http://doi.wiley.com/10.1111/j.1467-8659.2009.01478.x. 2.1.2

[2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. (document), 6.5, 6.2.3

[3] P.J. Besl and H.D. McKay. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, 1992. ISSN 0162-8828. doi: 10.1109/34.121791. 1.1, 4.1, 4.1.2, 7.1, 7.1

[4] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *IEEE Conf. on Intelligent Robots and Systems*, volume 3, pages 2743–2748, 2003. 2.2.3, 4.1.2, 7.2

[5] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2743–2748, 2003. 2.2.2, 2.2.3, 5.2, 5.3, 5.6.1

[6] Peter Biber, Sven Fleck, Wolfgang Strasser, Carl Rasmussen, Heinrich Blthoff, Bernhard Schlkopf, and Martin Giese. A probabilistic framework for robust and accurate matching of point clouds. In *Pattern Recognition*, volume 3175, pages 480–487. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22945-2. URL http://dx.doi.org/10.1007/978-3-540-28649-3_59. 2.2.3, 4.1.2, 7.2, 7.3.2

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 5.4.1, 6.1.1

[8] Daniel Blandford. Compact data structures with fast queries, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.2981. 2.1.2

[9] Rener Castro, Thomas Lewiner, Hlio Lopes, Geovan Tavares, and Alex Bordignon. Statistical optimization of octree searches. *Computer Graphics Forum*, 27(6):1557–1566, 2008. ISSN 01677055. doi: 10.1111/j.1467-8659.2007.01104.x. URL http://doi.wiley.com/10.1111/j.1467-8659.2007.01104.x. 2.1.2

[10] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.*, 32(4):113:1–113:16, 2013. 6.2.4, 6.2.5

[11] Yang Chen and Grard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992. Range Image Understanding. 4.1, 4.5, 7.1, 7.4.1

[12] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 545–548. IEEE, 2002. 7.2, 7.4.1, 7.1

[13] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299–309, 2005. 7.2

[14] Haili Chui and Anand Rangarajan. A feature registration framework using mixture models. In *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 190–197, 2000. 4.1.2, 4.1.2, 7.1, 7.2

[15] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. 5.6.1

[16] C. Connolly. Cumulative generation of octree models from range data. In *Proceedings. 1984 IEEE International Conference on Robotics and Automation*, pages 25–32, Atlanta, GA, USA, 1984. doi: 10.1109/ROBOT.1984.1087212. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1087212. 2.1.2

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 4.3.4

[18] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977. 2.2.1, 3.1, 4.1.2, 4.4.1, 7.1

[19] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR, 2010 IEEE Conference on*, pages 998–1005, 2010. 7.1

[20] Ivan Dryanovski, William Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1553–1559, Taipei, Taiwan, 2010. doi: 10.1109/IROS.2010.5652494. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5652494. 2.1.1

[21] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *IEEE International Conference on 3D Vision*. IEEE, 2015. 2.2.2, 4.5, 5.3, 7.2, 7.3.2, 7.4.1, 7.1

[22] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *3D Vision (3DV), 2015 International Conference on*, pages 241–249. IEEE, 2015. 4.1.2, 2

[23] Benjamin Eckart and Alonzo Kelly. REM-Seg: A robust EM algorithm for parallel segmentation and registration of point clouds. In *IROS*, pages 4355–4362, 2013. 2.2.2, 5.2,

5.2, 5.3

[24] Benjamin Eckart and Alonzo Kelly. REM-seg: A robust EM algorithm for parallel segmentation and registration of point clouds. In *IEEE Conf. on Intelligent Robots and Systems*, pages 4355–4362, 2013. 4.1.2, 1, 4.3, 4.4.1, 7.3.2

[25] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987. ISSN 0882-4967. doi: 10.1109/JRA.1987.1087096. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1087096. 2, 2.1.1

[26] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 5.3, 6.1.2

[27] Alberto Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1989. AAI9006205. 2, 2.1.1

[28] Alberto Elfes and Larry Matthies. Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representataion. In *26th IEEE Conference on Decision and Control*, pages 1802–1807, 1987. doi: 10.1109/CDC.1987.272800. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4049608. 2, 2.1.1

[29] Bharani Kumar Ellore. *Dynamically expanding occupancy grids*. PhD thesis, Texas Tech, 2002. URL http://etd.lib.ttu.edu/theses/available/etd-07312008-31295018555556/. 2.1.1

[30] Jan Elseberg, Dorit Borrmann, and Andreas Nüchter. One billion points in the cloud–an octree for efficient processing of 3d laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76:76–88, 2013. 5.2

[31] David Eppstein, Michael Goodrich, and Jonathan Sun. The skip quadtree: A simple dynamic data structure for multidimensional data. http://arxiv.org/abs/cs.CG/0507049, 2005. URL http://arxiv.org/abs/cs.CG/0507049. 2.1.2

[32] Georgios D Evangelidis, Dionyssos Kounades-Bastian, Radu Horaud, and Emmanouil Z Psarakis. A generative model for the joint registration of multiple point sets. In *Computer Vision–ECCV 2014*, pages 109–122. Springer, 2014. 2.2.2, 5.2, 5.3

[33] Georgios D Evangelidis, Dionyssos Kounades-Bastian, Radu Horaud, and Emmanouil Z Psarakis. A generative model for the joint registration of multiple point sets. In *ECCV 2014*, pages 109–122. 2014. 4.1.2, 4.1.2, 4.4.4, 4.5, 7.1, 7.2, 7.3, 7.3.1

[34] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974. ISSN 0001-5903. doi: 10.1007/BF00288933. URL http://www.springerlink.com/content/x7147683u3241843/. 2.1.2

[35] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003. 4.1.1

[36] Vincent Garcia, Frank Nielsen, and Richard Nock. Levels of details for Gaussian Mixture Models. In *ACCV*, pages 514–525. Springer, 2010. 2.2.2, 5.3

[37] Sarah Gibson. Calculating the distance map for binary sampled data. Technical report, 1999. 2.2.1

[38] Mike Giles and Robert Haimes. Advanced interactive visualization for cfd. *Computing Systems in Engineering*, 1(1):51–62, 1990. 6.2.1

[39] S. Gold, A. Rangarajan, C.P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching:: pose estimation and correspondence. *Pattern Recognition*, 31(8): 1019–1031, 1998. 4.3.4

[40] S. Gold, A. Rangarajan, C.P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching:: pose estimation and correspondence. *Pattern Recognition*, 31(8): 1019–1031, 1998. 1.1, 4.1.2, 4.1.2, 7.1

[41] Jacob Goldberger and Sam T Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems*, pages 505–512, 2004. 2.2.2, 5.2, 5.3

[42] Vladislav Golyanik, Bertram Taetz, and Didier Stricker. Joint pre-alignment and robust rigid point set registration. In *Proceedings of the International Conference on Image Processing —. International Conference on Image Processing (ICIP-16), Phoenix,, AZ, USA*. IEEE Xplore, 2016. 7.2

[43] S. Granger and X. Pennec. Multi-scale EM-ICP: A fast and robust approach for surface registration. *ECCV 2002*, pages 69–73, 2002. 1, 1.1, 4.1.2, 4.1.2, 4.2.1, 4.3.4, 4.4.4, 4.5, 7.1, 7.2

[44] S. Granger and X. Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. *Computer VisionECCV 2002*, pages 69–73, 2006. 4.3.4

[45] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009. 6, 7.5, 7.6

[46] Stefan Gumhold, Zachi Kami, Martin Isenburg, and Hans-Peter Seidel. Predictive point-cloud compression. In *ACM SIGGRAPH 2005 Sketches*, page 137, Los Angeles, California, 2005. ACM. 2.1.2

[47] Karan M. Gupta and Karan M. Gupta. *Monte Carlo localization for robots using dynamically expanding occupancy grids*. PhD thesis, Texas Tech University, April 2005. URL http://hdl.handle.net/2346/1010. 2.1.1

[48] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003. 2.2.1

[49] Dirk Hahnel, Sebastian Thrun, and Wolfram Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 915–920, 2003. 7.1

[50] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977. 7.2, 7.4.1, 7.1

[51] Radu Horaud, Florence Forbes, Manuel Yguel, Guillaume Dewaele, and Jian Zhang. Rigid and articulated point registration with expectation conditional maximization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(3):587–602, 2011. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.94. 4.1.2, 4.1.2, 4.4.4, 7.1, 7.2, 7.3.2

[52] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987. 4.1.1, 4.4.5, 4.4.5, 7.3.2

[53] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, pages 189–206, 2013. 5.2, 5.3, 6.1.2

[54] Yan Huang, Jingliang Peng, C.-C.J. Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):440–453, 2008. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70441. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4378368. 2.1.2

[55] Erik Hubo, Tom Mertens, Tom Haber, and Philippe Bekaert. The quantized kd-Tree: efficient ray tracing of compressed point clouds. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 105–113, Salt Lake City, UT, USA, 2006. doi: 10.1109/RT.2006.280221. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4061552. 2.1.2

[56] B. Jian and B.C. Vemuri. A robust algorithm for point set registration using mixture of gaussians. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1246–1251. IEEE, 2005. 2.2.2

[57] Bing Jian and Baba C. Vemuri. A robust algorithm for point set registration using mixture of Gaussians. In *IEEE Intern. Conf. on Computer Vision*, pages 1246–1251, 2005. 4.1.2, 4.1.2, 4.4.1, 7.2

[58] Bing Jian and Baba C. Vemuri. Robust point set registration using Gaussian mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1633–1645, 2011. URL http://gmmreg.googlecode.com. 4.3.4

[59] Bing Jian and Baba C. Vemuri. Robust point set registration using Gaussian mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1633–1645, 2011. URL http://gmmreg.googlecode.com. 1.1, 4.1, 4.1.2, 4.4, 4.4.6, 7.2

[60] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1633–1645, 2011. 2.2.2, 5.3

[61] Andrew E Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. 1

[62] D. Jung and K.K. Gupta. Octree-based hierarchical distance maps for collision detection. In *Proceedings of IEEE International Conference on Robotics and Automa-*

tion, pages 454–459, Minneapolis, MN, USA, 1996. doi: 10.1109/ROBOT.1996.503818. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=503818. 2.1.2

[63] Aravind Kalaiah and Amitabh Varshney. Statistical geometry representation for efficient transmission and rendering. *ACM Transactions on Graphics*, 24(2):348–373, 2005. 5.3

[64] S. Kambhampati and L. S Davis. Multiresolution path planning for mobile robots,. Technical report, May 1985. URL http://stinet.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA158630. 2.1.2

[65] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, 2006. (document), 6.5, 6.2.3

[66] Alonzo Kelly. *An intelligent predictive control approach to the high speed cross country autonomous navigation problem*. PhD thesis, Carnegie Mellon University, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.3292. 2.1.1

[67] Y. Kitamura, T. Tanaka, F. Kishino, and M. Yachida. 3-D path planning in a dynamic environment using an octree and an artificial potential field. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 474–481, Pittsburgh, PA, USA, 1995. doi: 10.1109/IROS.1995.526259. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=526259. 2.1.2

[68] Gerhard Kraetzschmar, Guillem Gassull, Klaus Uhl, MI Ribeiro, and Santos Victor. Probabilistic quadtrees for Variable-Resolution mapping of large environments. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*. Elsevier Science, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.338. 2.1.2

[69] James J Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *IEEE Intern. Conf. on Robotics and Automation*, volume 4, pages 3993–3998, 2004. 4.4.6

[70] N. Kumar, S. Satoor, and I. Buck. Fast parallel expectation maximization for gaussian mixture models on gpus using cuda. In *High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on*, pages 103–109. IEEE, 2009. 2.3, 3.3

[71] Rainer Kmmerle, Rudolph Triebel, Patrick Pfaff, Wolfram Burgard, Christian Laugier, and Roland Siegwart. Monte carlo localization in outdoor terrains using Multi-Level surface maps. In *Field and Service Robotics*, volume 42, pages 213–222. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-75403-9. URL http://dx.doi.org/10.1007/978-3-540-75404-6_20. 2.1.1

[72] J. Lalonde, R. Unnikrishnan, N. Vandapel, and M. Hebert. Scale selection for classification of Point-Sampled 3-D surfaces. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, pages 285–292, Ottawa, ON, Canada, 2005.

doi: 10.1109/3DIM.2005.71. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1443257. 4.5, 7.2

[73] J.-F. Lalonde, N. Vandapel, and M. Hebert. Data structures for efficient dynamic processing in 3-D. *The International Journal of Robotics Research*, 26(8):777–796, 2007. ISSN 0278-3649. doi: 10.1177/0278364907079265. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364907079265. 2.1.1

[74] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Sren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Michael Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium*, pages 163–168. IEEE, 2011. 7.1

[75] Y. Liu. Automatic registration of overlapping 3d point clouds using closest points. *Image and Vision Computing*, 24(7):762–781, 2006. 4.3.4

[76] Yufeng Liu, Rosemary Emery, Deepayan Chakrabarti, Wolfram Burgard, and Sebastian Thrun. Using EM to learn 3D models of indoor environments with mobile robots. In *ICML*, volume 1, pages 329–336, 2001. 2.2.1

[77] Yufeng Liu, Rosemary Emery, Deepayan Chakrabarti, Wolfram Burgard, and Sebastian Thrun. Using EM to learn 3D models of indoor environments with mobile robots. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 329336, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://portal.acm.org/citation.cfm?id=645530.655822. 2.2.1

[78] Yarden Livnat, Han-Wei Shen, and Christopher R Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996. 6.2.1

[79] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987. ISBN 0-89791-227-6. 6.2.1

[80] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4, 2004. 4.5

[81] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 7.2

[82] Yanxin Ma, Yulan Guo, Jian Zhao, Min Lu, Jun Zhang, and Jianwei Wan. Fast and accurate registration of structured point clouds with small overlaps. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016. 7.2

[83] L. Machlica, J. Vanek, and Z. Zajic. Fast estimation of gaussian mixture model parameters on gpu using cuda. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*, pages 167–172. IEEE, 2011. 2.3, 3.3

[84] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, pages 803–827, 2007. 5.2, 5.6.1

[85] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007. 4.1.2

[86] C. Martin and S. Thrun. Real-time acquisition of compact volumetric 3D maps with mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, pages 311–316, Washington, DC, USA, August 2002. doi: 10.1109/ROBOT.2002.1013379. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1013379. 2.2.1, 4.3.1

[87] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 727–733, Philadelphia, PA, USA, 1988. doi: 10.1109/ROBOT.1988.12145. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=12145. 2, 2.1.1

[88] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, June 1982. URL http://dx.doi.org/10.1016/0146-664X(82)90104-6. 2.1.2

[89] Soham Uday Mehta, Kihwan Kim, Dawid Pajak, Kari Pulli, Jan Kautz, and Ravi Ramamoorthi. Filtering Environment Illumination for Interactive Physically-Based Rendering in Mixed Reality. In *Eurographics Symposium on Rendering*, 2015. 7.1

[90] Xiao-Li Meng and Donald B Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993. 4.1.2, 7.2

[91] Bruce Merry, Patrick Marais, and James Gain. Compression of dense and regular point clouds. *Computer Graphics Forum*, 25(4):709–716, 2006. ISSN 1467-8659. URL http://dx.doi.org/10.1111/j.1467-8659.2006.00993.x. 2.1.2

[92] Niloy Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 22–31, Nice, France, 2004. ACM. ISBN 3-905673-13-4. URL http://dx.doi.org/10.1145/1057432.1057435. 4.1.1

[93] H Moravec and A Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121, January 1985. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087316. 2.1.1

[94] Hans P. Moravec. Three dimensional modelling and graphics with multiprocessors. http://www.frc.ri.cmu.edu/~hpm/project.archive/general.articles/1981/tridee, February 1980. URL http://www.frc.ri.cmu.edu/~hpm/project.archive/general.articles/1981/tridee. 2.1.2

[95] Hans P Moravec. 3D graphics and the wave theory. In *ACM SIGGRAPH Computer Graphics*, SIGGRAPH '81, page 289296, New York, NY, USA, 1981. ACM. ISBN 0-89791-045-1. doi: 10.1145/800224.806817. ACM ID: 806817. 2.1.2

[96] HP Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9:61–74,

1988. 2, 2.1.1

[97] W. Morris, I. Dryanovski, and J. Xiao. 3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids. In *RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras, Zaragoza, Spain*, 2010. 2.1.1

[98] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010. 4.1.2, 7.1, 7.2

[99] Andriy Myronenko, Xubo Song, and Miguel A Carreira-Perpinán. Non-rigid point set registration: Coherent point drift. In *Advances in Neural Information Processing Systems*, pages 1009–1016, 2006. 4.1.2, 4.1.2, 7.2

[100] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*, pages 127–136. IEEE, 2011. 2.1.1, 6.2.4, 6.2.5, 7.1, 7.2

[101] Timothy S Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006. 6.2.1

[102] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013. 2.1.2

[103] Nils J Nilsson. A mobile automaton: An application of artificial intelligence techniques. Technical report, January 1969. URL http://stinet.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA459660. 2, 2.1.2

[104] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam—3d mapping outdoor environments: Research articles. *J. Field Robot.*, 24 (8-9):699–722, 2007. 7.1

[105] Andreas Nchter, Kai Lingemann, and Joachim Hertzberg. 6D SLAM with cached K-D tree search. In *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, pages 101–106, W\ürzburg, Germany, 2007. ACTA Press. ISBN 978-0-88986-686-7. 1.1, 2.1.2

[106] In Kyu Park, Marcel Germann, Michael D. Breitenstein, and Hanspeter Pfister. Fast and automatic object pose estimation for range images on the gpu. *Machine Vision and Applications*, 21:749–766, 08/2010 2010. 7.1

[107] Sung-Bum Park and Sang-Uk Lee. Multiscale representation and compression of 3-D point data. *Trans. Multi.*, 11(1):177–182, 2009. 2.2.1

[108] Sung-Bum Park, H. Choi, and Sang-Uk Lee. Multiscale surface representation scheme for point clouds. In *Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004. Proceedings of 2004 International Symposium on*, pages 232–237, 2004. doi: 10.1109/ISPACS.2004.1439051. 2.2.1

[109] P. Payeur. Improving robot path planning efficiency with probabilistic virtual environ-

ment models. In *2004 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VCIMS).*, pages 13–18, Boston, MA, USA, 2004. doi: 10.1109/VECIMS.2004.1397177. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1397177. 2.1.2

[110] P. Payeur, D. Laurendeau, and C.M. Gosselin. Range data merging for probabilistic octree modeling of 3D workspaces. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, pages 3071–3078, Leuven, Belgium, 1998. doi: 10.1109/ROBOT.1998.680897. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=680897. 2.1.2

[111] Andreas Persson. 3D scan-based navigation using Multi-Level surface maps. http://www.essays.se/essay/496e9a83e5/, 2009. URL http://www.essays.se/essay/496e9a83e5/. 2.1.1

[112] Patrick Pfaff, Rudolph Triebel, and Wolfram Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *The International Journal of Robotics Research*, 26(2):217–230, February 2007. URL http://dx.doi.org/10.1177/0278364906075165. 2.1.1

[113] Jeff M Phillips, Ran Liu, and Carlo Tomasi. Outlier robust icp for minimizing fractional rmsd. In *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*, pages 427–434. IEEE, 2007. 7.2, 7.4.1, 7.1

[114] V Prieto-Maranon. Efficient plane detection in multilevel surface maps. In *Workshop of Physical Agents*, University of Castilla La Mancha, September 2010. 2.1.1

[115] Csar Rivadeneyra and Mark Campbell. Probabilistic multi-level maps from LIDAR data. *The International Journal of Robotics Research*, January 2011. doi: 10.1177/0278364910392405. URL http://ijr.sagepub.com/content/early/2011/01/20/0278364910392405.abstract. 2.1.1

[116] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001. 4.1.1, 7.1, 7.2, 7.4.1

[117] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, 2011 2011. 5.2

[118] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008. 1

[119] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 7.2

[120] Julian Ryde and Huosheng Hu. 3D mapping with multi-resolution occupied voxel lists. *Autonomous Robots*, 28(2):169–185, 2009. ISSN 0929-5593. doi: 10.1007/s10514-009-9158-3. URL http://www.springerlink.com/index/10.1007/s10514-009-9158-3. 2.1.2

[121] Julian Ryde and Huosheng Hu. 3d mapping with multi-resolution occupied voxel lists. *Auton. Robots*, 28(2):169–185, 2010. 5.2

[122] Julian Ryde and Huosheng Hu. 3d mapping with multi-resolution occupied voxel lists. *Autonomous Robots*, 28(2):169–185, 2010. 5.2

[123] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, Juha Ala-Luhtala, and Achim J Lilienthal. Normal distributions transform occupancy maps: Application to large-scale online 3d mapping. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2233–2238. IEEE, 2013. 2.2.3, 5.3

[124] Hanan Samet. Region representation: Quadtrees from binary arrays. *Computer Graphics and Image Processing*, 13(1):88–93, May 1980. URL http://dx.doi.org/10.1016/0146-664X(80)90118-5. 2.1.2

[125] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, June 1984. ISSN 0360-0300. URL http://dx.doi.org/10.1145/356924.356930. 2.1.2

[126] Hanan Samet. Applications of spatial data structures. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.73.6935, 1990. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.73.6935. 2.1.2

[127] Hanan Samet. *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN 0201502550. URL http://portal.acm.org/citation.cfm?id=77589. 2.1.2

[128] Hanan Samet. *Foundations of multidimensional and metric data structures*. Elsevier/Morgan Kaufmann, Amsterdam ;;Boston, 2006. ISBN 9780123694461. 2.1.2

[129] Ruwen Schnabel and Reinhard Klein. Octree-based Point-Cloud compression. *Proceedings of Symposium on Point-Based Graphics 2006*, July 2006. URL http://cg.cs.uni-bonn.de/aigaion2root/attachments/schnabel-2006-octree.pdf. 2.1.2

[130] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized ICP. *Robotics: Science and Systems*, 2(4), 2009. 4.1.2, 4.1.2, 4.5, 7.1

[131] Raj Shekhar, Elias Fayyad, Roni Yagel, and J Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In *Visualization*, pages 335–342, 1996. 6.2.1

[132] W. K. Stewart. A model-based approach to 3-D imaging and mapping underwater. In *Proceedings of the... International Conference on Offshore Mechanics and Arctic Engineering*, page 61, 1988. 2.1.1

[133] Todor Stoyanov, Martin Magnusson, and Achim J Lilienthal. Point set registration through minimization of the L2 distance between 3D-NDT models. In *IEEE International Conference on Robotics and Automation*, pages 5196–5201, 2012. 2.2.3, 4.1.2, 4.1.2, 4.4.1, 7.2, 7.3.2

[134] Todor Dimitrov Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim Lilienthal. Fast and accurate scan registration through minimization of the distance between compact

3D NDT representations. *International Journal of Robotics Research*, 2012. 1.1, 2.2.3, 4.1, 4.1.2, 4.4, 7.2, 7.3.2

[135] Gary K. Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank Langbein, Yonghuai Liu, A. David Marshall, Ralph Martin, Xianfang Sun, and Paul Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013. 7.1

[136] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda. Softassign and em-icp on gpu. In *Networking and Computing (ICNC), 2010 First International Conference on*, pages 179–183. IEEE, 2010. 4.3.4, 4.4.6, 4.4.7, 7.1, 7.2

[137] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard. A Real-Time Expectation-Maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–442, 2004. ISSN 1042-296X. doi: 10.1109/TRA.2004.825520. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1303689. 2.2.1

[138] Sebastian Thrun and Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. In *AAAI'96*, pages 944–950, 1996. 5.3, 6.1.2

[139] Sebastian Thrun, Christian Martin, Yufeng Liu, Dirk Hahnel, Rosemary Emery-Montemerlo, Deepayan Chakrabarti, and Wolfram Burgard. A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–443, 2004. 2.2.1

[140] Rudolph Triebel, Wolfram Burgard, and Frank Dellaert. Using hierarchical EM to extract planes from 3D range scans. In *IEEE ICRA*, pages 4437–4442, 2005. 2.2.1

[141] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-Level surface maps for outdoor terrain mapping and loop closing. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2276–2282, Beijing, China, October 2006. IEEE. ISBN 1-4244-0258-1. URL http://dx.doi.org/10.1109/IROS.2006.282632. 2.1.1

[142] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. *ECCV 2004*, pages 558–569, 2004. 4.1.2, 4.1.2, 4.4, 4.4.1, 7.2

[143] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH*, pages 311–318, 1994. (document), 4.8, 4.4.6

[144] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 311–318. ACM, 1994. 5.6

[145] Ranjith Unnikrishnan, Jean-Francois Lalonde, Nicolas Vandapel, and Martial Hebert. Scale selection for the analysis of Point-Sampled curves. In *3D Data Processing Visualization and Transmission, International Symposium on*, volume 0, pages 1026–1033, Los Alamitos, CA, USA, 2006. IEEE Computer Society. ISBN 0-7695-2825-2. doi: http://doi.ieeecomputersociety.org/10.1109/3DPVT.2006.123. 4.5, 7.2

[146] Nicolas Vandapel, Daniel Huber, Anuj Kapuria, and Martial Hebert. Natural terrain classification using 3-D ladar data, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.1402. 2.1.2

[147] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics*, 26(10):812–840, 2009. ISSN 15564959. doi: 10.1002/rob.20309. URL http://onlinelibrary.wiley.com/doi/10.1002/rob.20309/abstract. 2.1.2

[148] Saona Vazquez, I Navazo, and P Brunet. The visibility octree: A data structure for 3d navigation, 1999. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.3025. 2.1.2

[149] Miao Wang and Yi hsing Tseng B. Lidar data segmentation and classification based on octree structure. In *Geo- Imagery Bridging Continents 20th ISPRS Congress*, Istanbul, Turkey, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.9934. 2.2.1

[150] M.D. Wheeler and K. Ikeuchi. *Iterative estimation of rotation and translation using the quaternion*. School of Computer Science, Carnegie Mellon University, 1995. 4.3

[151] Ross Whitaker. A Level-Set approach to 3D reconstruction from range data. *Int. J. Comput. Vision*, 29(3):203–231, 1998. ISSN 0920-5691. URL http://dx.doi.org/10.1023/A:1008036829907. 2.2.1

[152] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Trans. Graph.*, 11(3):201–227, 1992. 6.2.1

[153] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: a probabilistic, flexible, and compact 3D map representation for robotic systems. In *ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010. URL Softwareavailableathttp://octomap.sf.net/. Software available at http://octomap.sf.net/. 1.1, 2.1.2

[154] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *ICCV 2013*, pages 1457–1464, 2013. 7.2

[155] Alan Yuille, Song-Chun Zhu, Daniel Cremers, Yongtian Wang, Yi-Hsing Tseng, Kai-Pei Tang, and Fu-Chen Chou. Surface reconstruction from LiDAR data with extended snake theory. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 4679 of *Lecture Notes in Computer Science*, pages 479–492. Springer Berlin / Heidelberg, 2007. URL http://dx.doi.org/10.1007/978-3-540-74198-5_37. 2.2.1

[156] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):112, 2013. (document), 4.8, 4.4.6, 5.6, 6.2.1, 6.4, 6.8

[157] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV 2016*, pages 766–782, 2016. 7.2