# Hierarchical Gaussian Distributions for Real-Time SLAM

Aditya Dhawale

CMU-RI-TR-20-04

May 2020


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Nathan Michael, Chair
Michael Kaess,
Ming-Fang Chang

*Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.*

# Abstract

We present Gaussian distributions as structure primitives in a hierarchical multi-fidelity framework to enable accurate real-time Simultaneous Localization and Mapping (SLAM) using uncertain depth data.

Real-time mapping and localization capabilities are essential components of an autonomous system deployed in real-world environments. An autonomous system must be able to create an understanding of the world from the history of observed sensor information in unknown environments and operate appropriately in response. A typical state-of-the-art mobile robot has multiple perceptual processes operating concurrently on the incoming sensor information to enable various autonomy subsystems. Each subsystem processes the sensor data independently to obtain information suitable for its operation that is often unusable by other subsystems. Such a disjoint autonomy system creates redundancy in terms of data processing, and increases the computational load on the mobile system, the overall memory footprint, and the modes of failure. However, mobile robots deployed in real-world scenarios are Size Weight and Power (SWaP) constrained. SWaP constrained platforms impose constraints on the computational and memory resources available onboard thus introducing unique challenges in deploying such disjoint perceptual models in real-world. In this thesis, we propose a SLAM framework using a memory and computationally efficient map representation that can be utilized for various low level autonomy tasks and unify the perceptual architecture.

The real-time performance of active perception algorithms is dependent on the memory complexity of the used map representation. Higher memory footprint increases the computational complexity of active perception algorithms with marginal benefit to their performance. State-of-the-art SLAM techniques generate extremely high fidelity 3D reconstruction of the world. These 3D maps are often not suitable for active perception due to their high memory requirements. They must be post-processed, down-sampled and converted to a map representation more suitable for active perception such as voxel grids. Voxel grids provide high computational benefits and low memory footprint at the cost of loss of map accuracy and fidelity.

A family of generative map models have recently been proposed that compress point cloud data using hierarchical Gaussian Mixture Models (GMMs) by modeling the structural correlations evident in the input data. These generative models are more memory efficient and accurate than voxel based map representations. The generative nature of GMMs enable them to be elegantly converted to more commonly used map representations. Alongside dense 3D reconstruction, there is a wide range of work showing the utility of a GMM based map representation for scan matching, global and local pose estimation, collision avoidance, autonomous exploration and various other real-world robotic applications. However, the loss in mapping accuracy and map fidelity is not yet addressed and there is a significant gap between the quality of reconstruction obtained from state-of-the-art SLAM pipelines and generative mapping representations such as GMMs.

In this work, we combine the mathematical benefits of a generative map representation such as GMMs with the accuracy of reconstruction obtained from dense map

representations such as surfels. Specifically, we create a hierarchical Gaussian distributions based map that is capable of reconstructing the world with high accuracy at lower hierarchical levels and retain the memory efficiency of GMMs at higher hierarchical levels. We propose a novel model fitting approach to raw point cloud data, that uses the projective constraints enforced by depth cameras to reduce the computational complexity of fitting a GMM to large scale data. Further, we propose a frame-to-model localization approach, that exploits the hierarchical structure of the map to obtain a more robust and reliable camera tracking performance. The reduced memory complexity of the proposed map representation enables deployment of our proposed approach on SWaP constrained systems.

Additionally, we highlight the computational benefits obtained by using the proposed map representation for two family of problems:

- Global pose estimation and re-localization using a multi-hypothesis particle filter in a GMM map
- Reactive collision avoidance using Gaussian distributions as geometric primitives

We demonstrate the superior qualitative and quantitative performance obtained by using the proposed map representation for applications such as SLAM, global localization and collision avoidance as compared to their respective state-of-the-art approaches. We also highlight the increased computational efficiency, reduced memory footprint and high reconstruction accuracy achieved using this map representation in simulated and real-world environments.

*To my Baba.*

## Acknowledgments

I am extremely thankful to my advisor, Prof. Nathan Michael. He gave me an opportunity to pursue robotics at CMU right after graduation, provided me constant guidance, support, and introduced me to the exciting world of research. As a Master's student, he supported my ideas, motivated me, made sure I was on the right track, and encouraged me to pursue research that I was interested in. I would also like to express my immense gratitude towards Kumar Shaurya Shankar for everything he has taught me over the duration of my stay at CMU. This thesis would not have been possible without his wisdom, guidance, and all the extremely engaging, fun and endless research discussions.

I would like to thank my committee members, Prof. Michael Kaess, and Ming-Fang Chang, for their valuable feedback, constant motivation, and, encouraging research discussions.

I would like to extend my sincerest thanks to all my past and present colleagues in the Resilient Intelligent Systems Lab at CMU for helping me grow as a researcher and for providing me multiple opportunities to collaborate on exciting research projects. Thank you Kumar Shaurya Shankar, Xuning Yang, Wennie Tabib, Vibhav Ganesh, Arjav Desai, Cormac O'Meadhra, Matt Collins, John Yao, Shobhit Srivastava, Curtis Boirum for creating an environment conducive to excellent research and personal growth. A special thanks to all my colleagues in the Robotics Institute, Jerry Hsiung, Cherie Ho, Pragna Mannam, Ada Taylor, Kevin Edelson, Keene Chin, Kate Shih, Thomas Weng and many more for all your love and encouragement, and for being a family away from home.

Thanks to my beautiful family for the constant strength, support and freedom that they have provided me throughout my entire adult life. Special thanks to my mother, and my brother for their immense strength and belief in me. Thank you dad, for everything you did for me and our family. None of this was possible without you.

# Contents

# List of Figures

# List of Tables

# Symbols

| | |
|---|---|
| ${}^{a}\mathbf{X}_{N}$ | A set of N points in 3D or a XYZ point cloud in a coordinate frame $a$ |
| ${}^{a}\mathbf{x}_{i}$ | $i^{\text{th}}$ 3D vector describing the $\{x, y, z\}$ coordinates of a 3D point in a coordinate frame $a$ |
| $\boldsymbol{\mu}_{i}$ | Mean of $i^{\text{th}}$ Gaussian distribution |
| $\boldsymbol{\Sigma}_{i}$ | Covariance of $i^{\text{th}}$ Gaussian distribution |
| $N_{i}$ | Number of points used to fit the $i^{\text{th}}$ Gaussian distribution |
| $\mathbf{c}_{i}$ | color of $i^{\text{th}}$ Gaussian distribution (RGB) |
| ${}^{a}\boldsymbol{\theta}_{j}^{l}$ | A 3D Gaussian distribution at $l^{\text{th}}$ hierarchical level defined using mean, covariance, color, $\left\{{}^{a}\boldsymbol{\mu}_{j}, {}^{a}\boldsymbol{\Sigma}_{j}, N_{j}, \mathbf{c}_{j}\right\}$ in a coordinate frame $a$ |
| ${}^{a}\boldsymbol{\Theta}^{l}$ | A set of 3D Gaussian distributions at $l^{\text{th}}$ hierarchical level in a coordinate frame $a$ |
| ${}^{a}\bar{\boldsymbol{\Theta}}$ | A 3D Gaussian Mixture Model in a coordinate frame $a$ |
| $\mathbb{SE}(3)$ | Special Euclidean Group |
| $\mathbb{SO}(3)$ | Special Orthogonal Group |
| ${}^{b}\mathbf{R}_{a}$ | A rotation matrix $\in \mathbb{SO}(3)$ that rotates points in coordinate frame $a$ in to coordinate frame $b$ |
| ${}^{b}\mathbf{t}_{a}$ | A translation vector $\in \mathbb{R}(3)$ that translates points in coordinate frame $a$ in to coordinate frame $b$ |
| ${}^{b}\mathbf{T}_{a}$ | A transformation matrix $\in \mathbb{SE}(3)$ that transforms points in coordinate frame $a$ in to coordinate frame $b$ |

$${}^{b}\mathbf{T}_{a} = \begin{bmatrix} {}^{b}\mathbf{R}_{a} & {}^{b}\mathbf{t}_{a} \\ \mathbf{0} & 1 \end{bmatrix}$$

# Chapter 1

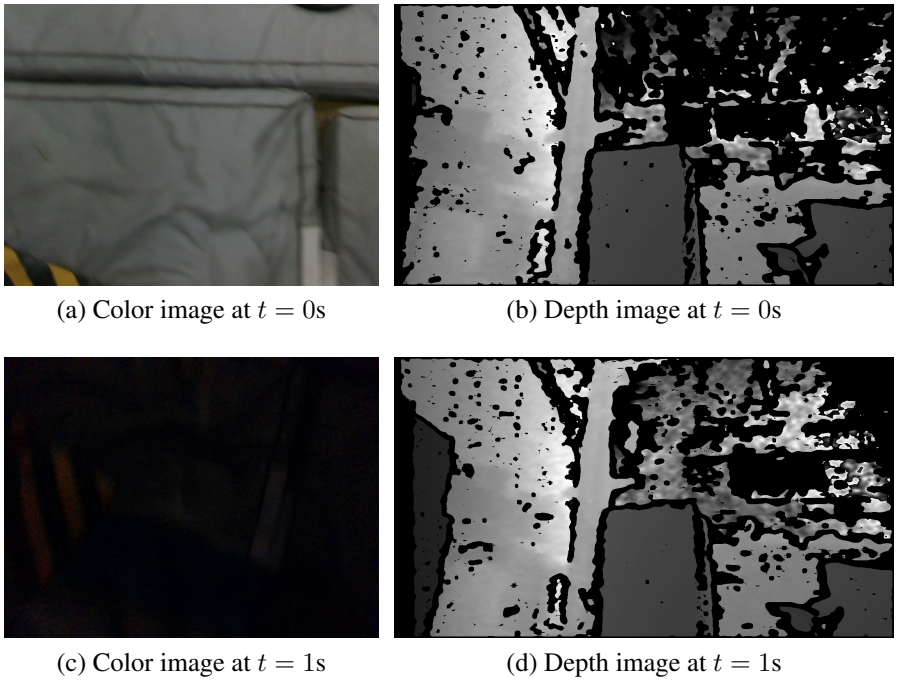
# Introduction

The motivation behind this thesis is to enable real-time dense Simultaneous Localization and Mapping (SLAM) on Size Weight and Power (SWaP) constrained systems using noisy depth sensor information. Depth sensors provide structure information about the objects in the scene. Pose estimation and scan alignment using such structural 3D information has been shown to perform robustly in challenging environments with high accuracy [17, 45]. Such dense 3D data also enables high accuracy mapping and scene reconstruction [58]. However, accurate laser based 3D sensors like a Velodyne LIDAR [1] are expensive and heavy thus restricting their application to ground robots. The advent of cheap and light weight depth sensors such as a Kinect [2] have made it possible to equip a mobile aerial robot with depth information and facilitate more robust and reliable operation. However, depth measurements obtained from COTS depth sensors are uncertain and unreliable [19]. Improper handling of this uncertainty leads to propagation of errors throughout the autonomy pipeline.

Enabling a SWaP constrained robot to create dense, accurate maps using only depth information is especially valuable in challenging environments where cameras operating in visual spectrum

[1] https://velodynelidar.com/products
[2] https://en.wikipedia.org/wiki/Kinect

fail, like poorly illuminated environments as illustrated in Fig. 1.1 where the color camera does not observe useful information about the scene but the depth sensor provides dense structural information about the scene or texture-less regions [2]. Indeed, the depth sensor has modes of failure specific to its mode of operation, when the structure information in the scene is repetitive and indistinguishable from multiple perspectives as shown in Fig. 1.2 A reliable autonomous system must use multiple sensors providing independent information about the scene to be robust to various challenging environments. For the scope of this thesis, we focus on only depth information and push the limits of robustness of a depth based localization and mapping framework.

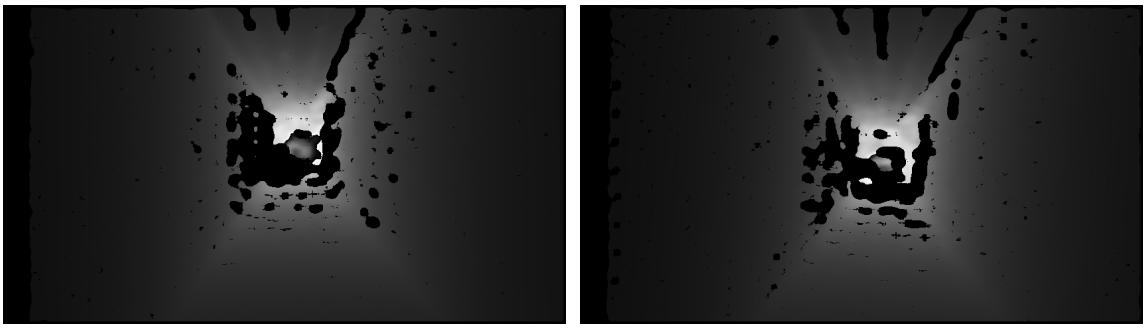


(a) Color image at $t = 0$s (b) Depth image at $t = 0$s

(c) Color image at $t = 1$s (d) Depth image at $t = 1$s

*Figure 1.1: Challenging dataset for a visual odometry system using a forward facing depth camera and a downward facing color camera. As the robot moves in a structured environment, the lights are turned off and the color image fails to observe useful information. The depth camera still provides informative structural data that enables a depth based odometry system to perform reliably.*

Depending on the perceptual requirements, different perception frameworks process the input sensor point cloud data differently. For dense 3D reconstruction of the world, every individual point is processed independently and stored either as a high resolution point cloud, or a set of surfels (surface elements) [58], or is used to update a fine resolution voxel grid [26]. For pose esti-



(a) Depth image at $t = 0$s (b) Depth image at $t = 1$s

*Figure 1.2: Challenging dataset for a depth based odometry system. The robot moves in a hallway environment, where the depth camera does not observe distinguishable information from sequential sensor measurements, thus leading to failure of depth based odometry system.*

mation and scan alignment, the point cloud is usually heavily sub-sampled and a small subset of the original data is operated upon to reduce the computational burden on the system [12]. For applications such as collision avoidance and safe local navigation, a low resolution voxel grid is typically created centered around the robot that compresses the sensor information aggressively [6, 49]. For exploration, a global occupancy grid is maintained that contains information about the observed and unobserved sections of the map [18]. A mobile robot deployed in real world scenarios must be able to perform all these operations concurrently. Due to the limited availability of computational resources all the subsystems may need to compromise on accuracy of performance for real-time operation. Failure on part of any of these subsystems to perform in real-time may lead to catastrophic failure and damages. The redundancy created by processing sensor data independently and storing as different map representations creates additional burden on the computational and memory resources. The map representation proposed in this work, however, is succinct, continuous, high fidelity, memory efficient and is generalizable for various applications. We represent the structure information as a set of independent hierarchical generative distributions, that elegantly incorporate noisy sensor information, can accurately generate high fidelity 3D reconstruction of the world [11] and is readily usable for localization [12], frame-to-model tracking [14], collision avoidance [13] and exploration [55].

## 1.1 Thesis Statement

In this thesis, we present a SLAM approach using Gaussian distributions as structure primitives in a hierarchical framework to enable accurate real-time 3D reconstruction of the world. We propose that the reduced computational complexity of the proposed map representation enables real-time implementation of the proposed SLAM framework on computationally constrained systems. Additionally, we demonstrate the applicability of Gaussian distributions as structure primitives for tackling challenging problems in robotics such as efficient global localization and motion primitive based collision avoidance.

## 1.2 Thesis Outline

This thesis is structured as follows:

- Chapter 2: Provides a review of related literature and draws comparison of state-of-the-art approaches to the proposed work
- Chapter 3: Provides a summary of the mathematical preliminaries used throughout this work
- Chapter 4: Provides a detailed description of the hierarchical 3D reconstruction framework and provides qualitative and quantitative evaluation of the accuracy of map reconstruction alongside the memory footprint of the map representation. Unlike state-of-the-art generative mapping approaches our approach minimizes the error in 3D reconstruction explicitly and therefore generate an accurate and succinct representation with lower computational complexity. [11]
- Chapter 5: Introduces our frame-to-model localization algorithm that tracks a live sensor observation with respect to the estimate of the 3D map representation. Quantitative comparison to state-of-the-art high fidelity mapping approaches shows superior tracking and mapping performance while being more robust to failures on challenging datasets. [14]

- Chapter 6: Presents two applications of the proposed map representation: multi-hypothesis localization with respect to a global map of the world, and, safe navigation and collision avoidance in unknown environments using the low fidelity map representation obtained at the highest hierarchical level from our mapping framework, on SWaP constrained systems. [12, 13]
- Chapter 7: Summarizes the contributions of this thesis work and describes the variety of future research directions that this work enables.

# Chapter 2

# Related Work

This thesis seeks to enable depth based SLAM on SWaP constrained systems using a Gaussian distributions in a multi-fidelity hierarchical framework as structure primitives. In this chapter, we provide a brief overview of the state-of-the-art approaches in the domains of map representation and SLAM. As the motivation behind this work is to enable SLAM on computationally constrained systems, we focus on the computational efficiency and complexity trade-off of the state-of-the-art approaches and highlight how the proposed map representation differs from them and mitigates their drawbacks.

## 2.1 Representation of 3D Structure Data

The advent of sensors that can detect the 3D structure in the scene is a milestone for autonomous robotics and has enabled the use of robotics in a plethora of challenging problems. Starting from the ultrasonic sensors that can provide a single point measurement along a direction, to Kinect [1] and LIDAR [2] sensors, the sensing accuracy has improved, and the amount of information obtained in a single scan has increased, yet the most commonly used map representations for robotics ap-

[1] https://en.wikipedia.org/wiki/Kinect
[2] https://velodynelidar.com/products/

plications have remained the same. In this section,

- first, we discuss the information provided by standard time-of-flight, structured light, or stereo depth sensors
- second, we discuss the state-of-the-art map representations used to efficiently and accurately represent depth sensor information
- finally, we discuss multiple state-of-the-art SLAM approaches that are built upon these map representations

### 2.1.1 Structure Sensors

Commercially available Off-The-Shelf (COTS) depth sensors can be broadly classified into the following categories:

- Time-of-Flight (TOF) sensors: These sensors operate in the infrared spectrum. A ray of light is emitted from the emitter which is then received by the receiver after reflection with the surfaces in the environment. The receiver measures the difference between the time the ray was emitted to the time when the ray was received. This time difference is directly proportional to the distance of the closest surface along the direction of the ray. This distance can be computed as half the speed of the ray multiplied by the time difference. Sensors like the Kinect One and LIDAR operate using this technology. These sensors are extremely accurate at high depth ranges. However, they have a slow frame-rate and suffer from motion blur.
- Structured light sensors: These sensors also operate in the infrared spectrum. A fixed pattern of IR light is projected onto the environment in the sensor Field-Of-View (FOV) from the projector as shown in Fig. 2.1. This pattern gets distorted according to the spread of the surfaces of objects in the scene. An infrared camera captures this distortion and estimates the depth of the surfaces along each ray corresponding to the extent of the distortion. These

sensors are typically less accurate than TOF sensors.

- Stereo sensors: A pair of color or IR cameras form a stereo pair. A well calibrated stereo pair can be used to compute the depth of the scene by finding the pixel locations corresponding to the same 3D point in both the images, using the camera intrinsic parameters, and the extrinsic calibration between the stereo pair. However, since the modality of measurement is completely passive unlike the TOF sensors or the structured light sensors, stereo sensors are the least accurate depth sensors and have a high bias and uncertainty in their measurements.

All the types of sensors listed above provide observations as a set of discrete depth measurements along individual rays in 3D. The most common way to represent this information is as 3D point clouds. Further, these sensors operate in the IR, or visible spectrum. This spectrum of radiation does not penetrate through solid surfaces with an exception of transparent surfaces like glass. Therefore, a COTS depth sensor can only provide distance to the closest surface along each ray direction. Therefore, the point cloud data obtained from a depth sensor are samples drawn from an underlying continuous function that represents the surfaces in the scene. Formally, the point cloud data in 3D is spread on a lower dimensional manifold of $\mathbb{R}(3)$. This understanding is exploited in Chapter 4 of this thesis work. The sensor measurements obtained from the COTS depth sensors are arranged in an ordered pattern. Due to the projective nature of these sensors, observed 3D points that are in the proximity of each other in 3D space are proximate to each other in pixel space. Additionally, since the observed points are spread along the surfaces of objects, neighboring sensor observations in image space often belong to the same surface and provide substantial information about the same surface. Formally, we can state that sensor observations are sampled from a biased underlying distribution and are not IID in $\mathbb{R}(3)$.

In the following section, we describe state-of-the-art map representations and SLAM frameworks that depending upon the computational, or memory constraints and mathematical convenience, exploit or relax the understanding about a depth sensor described above. Later, we describe

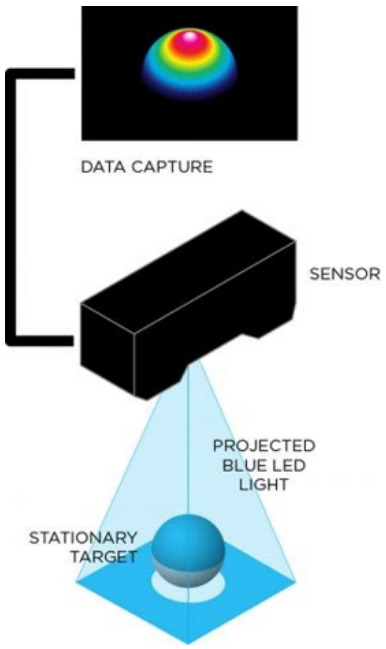



*Figure 2.1: Illustration of the method of working of depth sensors. Depth sensors measure distances to the closest surface along each ray with some uncertainty.*
***Source**: https://www.abizsensor.com/products/3D-smart-sensor/lmi.html*

how the proposed map representation incorporates these constraints to reduce the computational complexity of the mapping algorithm.

### 2.1.2 Voxel Grids

Voxel grids are the most commonly used representation of 3D data. Voxel grids provide an efficient and elegant way to represent dense 2D and 3D structure data using grid cells, known as voxels. Many variants of the voxel grid based maps have been proposed and depending on the task at hand one may be more advantageous over the others.

Voxels grids were first proposed as an efficient representation of 2D data by Moravec and Elfes [34]. The world is divided into a uniform 2D grid of fixed size and each grid cell (voxel) stores a probability of occupancy. The authors propose a novel idea of interpreting a sensor return from a depth sensor as a ray of measurement that provides information about the 2D space that is free until the first surface is observed. Occupancy grids have since been used extensively for various appli-

cations in trajectory planning, safe navigation and mapping. 2D voxel grids were soon extended to 3D, however, due to the high memory requirements of 3D voxel grids, their applications were limited to small scale environments [55]. The memory complexity scales quadratically with the size of the environment and size of the voxels for 2D voxel grids and cubically for 3D voxel grids. Applications that require fine resolution voxel grids [26] even today are limited by the size of the environment they can operate in. However, voxel grids provide a constant time memory lookup in terms of computational complexity and are therefore very efficient for run-time performance. Multiple adaptive variants of voxel grids have been proposed to overcome the high memory footprint while maintaining the low computational complexity of vanilla voxel grids.

A major challenge with voxel grid mapping is knowing the extent of the map and the resolution of voxel grid. The finer the resolution of the map, the smaller the map extents are given the limited availability of memory resources on a system. A standard engineering solution to circumvent this problem is to use a sliding window voxel grid [55]. A dynamically sliding fixed size voxel grid is maintained, centered around the robot and as the robot moves, the extents of the voxel grid are moved. This enables the robot to utilize a locally consistent occupancy map for safe navigation and local trajectory planning. This approach has a constant memory footprint and a constant time occupancy lookup. However, the robot can only keep track of local information and therefore is not usable for applications such as map exploration or global 3D reconstruction where information of the global world is required.

A novel approach proposed by Hornung et al. [24] uses an adaptive Octree based mapping approach that enables creating a voxel grid map of large scale environments at fine resolutions. An Octree is recursively created by dividing each voxel grid into 8 smaller cells. This enables the robot to store the range data at a high resolution in the observed parts of the map and large unobserved sections of the map can be represented using a single low resolution Octree root node. An OctoMap is a collection of Octrees where each octree can be recursively divided in to smaller

octrees. However, due to the recursive and adaptive nature of OctoMap, the lookup complexity of OctoMap is $\mathcal{O}(\log n)$, where $n$ is the depth of octrees. The memory complexity for OctoMap is dependent on the highest resolution voxel size and therefore scales cubically as larger map environments are observed.

Voxel grids encode information about the free space as well as the occupied space that is used for various perceptual applications in an autonomous system. For accurate 3D reconstruction, the representation of free space is not required, and therefore, only the occupied voxels are used to represent the surfaces in the world. The following map representations model the spread of observed 3D measurements and do not represent the free space. For the scope of this work, our proposed map representation does not represent free space information either.

### 2.1.3 Parametric Surface Elements

#### Surfels

As described in Sec. 2.1.1, 3D point clouds obtained from sensors are spread on the surfaces in the FOV of the sensor. This suggests that the 3D observations are locally spread on small planar surfaces with small curvature. Therefore, the 3D point cloud data can be approximated as tiny planar surfaces, which together represent the solid surfaces in the scene. Various state-of-the-art SLAM approaches such as [43, 58] use 2D circular surface elements (surfels), to represent surfaces of objects of in scene, as illustrated in Fig. 2.2. Each surfel is parameterized by the 3D location, 3D normal, color information and confidence weight. Surfels enable a high fidelity 3D reconstruction of the scene where the number of surfels depend on the scale and the complexity of the environment. Surfels are computationally efficient to fit however, they have a large memory footprint and the memory lookup complexity is $\mathcal{O}(n)$ where $n$ is the number of surfels in the scene.

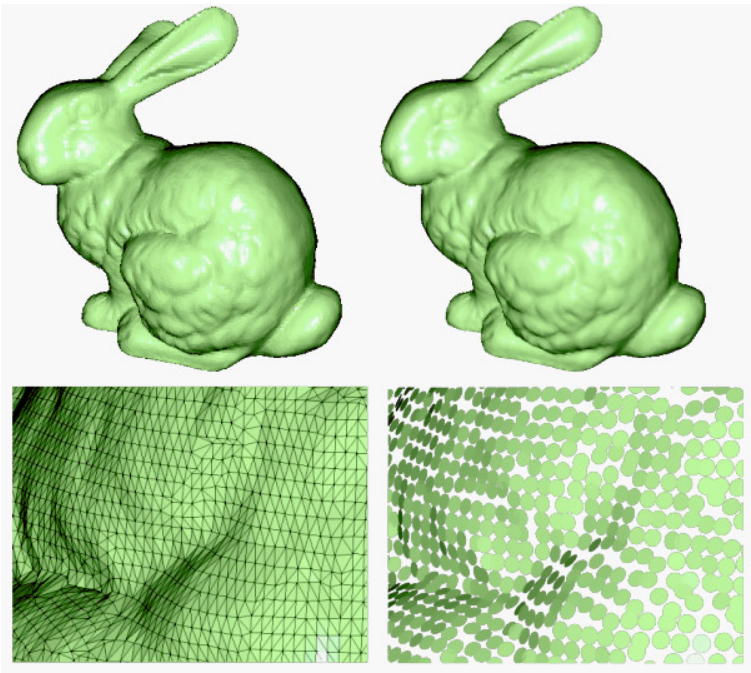


*Figure 2.2: Illustrative example of parametric surface representations: surfels and triangulated meshes. Top: Input dense structure data, Bottom Left: A triangulated mesh representation (see Sec. 2.1.3) of the input point cloud data, Bottom Right: Surfel based representation (see Sec. 2.1.3) of the input data.*
***Source** [9]*

### 3D Planes

Many environments, especially indoor environments consist of large planar structures. Such planar environments enable the representation of 3D structure using high level features like large planes over 3D point clouds or voxels. Extending upon the idea of planar surface primitives, Kaess [28] proposed a SLAM approach that fits large parametric planes to indoor scenes and uses only planar information for 3D reconstruction and localization. Dominant planes in the scene are segmented using standard clustering algorithms and the 3D map is represented as a set of dominant planes. This approach provides an extremely efficient and parametric map representation that can readily be used for robotic applications like pose estimation and collision avoidance. However, such approaches fail to represent majority of the sensor information in more complex environments where the geometry of the scene is not planar and are unable to reconstruct objects with large surface curvatures.

#### Triangulated Mesh

Surfels are discrete circular elements where each surfel is independent of the others in the map. This creates discontinuities in the map and leads to holes in the reconstructed structure. Similar to surfels, meshes create a high fidelity representation of the surfaces in the scene from point cloud data [10, 26]. Sets of neighboring 3D points are combined together to form triangles that represent locally planar surfaces in the scene. Meshes are widely used as a map representation in graphics community for 3D object reconstruction and manipulation. Due to the graphical nature of triangulated meshes, they are extremely conducive to mesh manipulation, deformation and high fidelity 3D reconstruction. However, they are computationally expensive to construct as incremental updates affect a large section of the mesh unlike surfels. Further, similar to surfels, meshes have a large memory footprint making them infeasible to be deployed on SWaP constrained systems for real-world applications.

### 2.1.4 Generative Volumetric Models

All the map representations discussed above fit a specific representation to raw 3D point cloud data using a non-invertible mapping function. Thus the information lost from converting point cloud to map representation cannot be reconstructed. Recently, a new class of mapping techniques have been proposed that attempt to compress the input sensor data using generative parametric models, and reconstruct the raw sensor measurement as required. The incoming sensor data is assumed to be IID in 3D and sampled from some unknown underlying distribution. As more 3D points are observed from the stream of sensor information, the model parameters are updated to best approximate the underlying distribution with large incremental data. Jian and Vemuri [27] proposed a novel approach of representing 3D point clouds by a GMM by fitting a Gaussian distribution over each point. However, this approach is not scalable with large scale data obtained from 3D sensors. Stoyanov et al. [51] modified this approach using concepts from voxel grid based mapping

to fit Gaussian distributions to points within fixed sized voxels. Since a parametric distribution is fitted within each voxel, this approach creates a more accurate reconstruction of the 3D structure than simple voxel grids. A semi-continuous representation of the 3D occupancy is reconstructed, however, it has larger memory footprint than vanilla voxel grids as more memory are required to store the parameters of a Gaussian distribution. To mitigate some of the issues posed by NDT map, Eckart et al. [16] and Srivastava and Michael [50] proposed a novel hierarchical GMM learning technique on 3D point cloud data, as shown in Fig. 2.3. A globally consistent GMM is fitted to the entire history of sensor observation at each time step thus creating a continuous, smooth and accurate map representation. Both approaches quantify the accuracy of map representation using the log-likelihood of the 3D point cloud data having sampled from the current estimate of the GMM, as a heuristic. An iterative Expectation Maximization (EM) technique is employed to find the best set of parameters that find a local maxima of this log-likelihood.

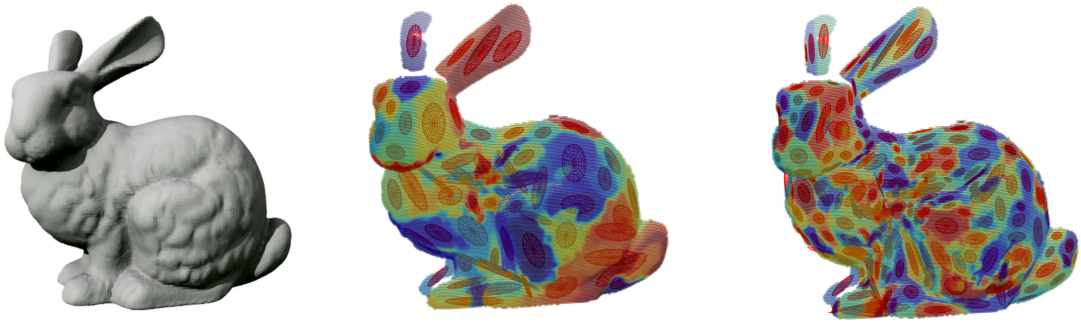


*Figure 2.3: Illustrative example of a hierarchical GMM based representation of structure data.*
***Source** [16]*

### Model Complexity Estimation

The bottom-up hierarchical model-based mapping approach presented in [50] initializes the model with an over estimated model complexity, which is subsequently reduced via merging components while maintaining required representational fidelity. This approach however assumes sensor information to be perfect and does not accurately update the map representation with the incrementally observed sensor information. Further, for real-time application, this approach fits a empirically

determined fixed number of Gaussian distributions to every sensor data at the highest fidelity level. The top-down hierarchical mapping approach presented in [16] assumes the model complexity to be fixed at each hierarchical level and decomposes components into smaller Gaussian distribution to reach the desired model complexity. Their highly optimized mapping and registration algorithms run in real-time on a mobile GPU, and the results are comparable to the state-of-the-art trajectory tracking approaches. This top-down hierarchical approach can, however, be improved upon further by adaptively computing the model complexity at each hierarchical level from the data itself. An accurate estimate of the model complexity at each hierarchical level will avoid under-fitting or over-fitting the model to the sensor observation. For real-world data, the number of components is unknown and should be inferred from the data itself [59] rather than using a-priori estimate.

Estimation of the optimal model complexity is usually addressed by using various information criteria such as the Akaikes Information Criterion (AIC) [1], Bayesian Information Criterion (BIC) [30], Minimal Description Length (MDL) [21], among others. AIC is a measure of the divergence between the true distribution of the data likelihood and the likelihood distribution of the fitted model. BIC estimates the posterior probability of a fitted model being true, under a certain Bayesian setup. The complexity estimates provided by BIC are shown to be consistent with the results from MDL [21]. Both BIC and MDL are closely related to AIC but have higher penalties for the number of parameters in the model. These criteria are based on a fundamental assumption that the input data is distributed in the exponential family. However, real world data obtained from depth sensors do not conform to this assumption. Consequently, application of these criteria causes over-estimation of true model complexity. We direct the reader to [7] for an overview of the performance comparison of various model selection criteria. Additionally, these approaches require the model to be first fitted over a range of complexity values before selecting the complexity value that minimizes a desired criteria value. Therefore, these methods become computationally intractable as data size and the maximum number of components increase. A characteristic function based

model complexity estimation approach is proposed in [59]. This approach avoids over-fitting the model but is computationally expensive for real-time applications.

**Model Fitting: Expectation-Maximization**

Expectation-Maximization is the most commonly used parameter fitting technique for GMMs. Jian and Vemuri [27] finds EM algorithm to be extremely sensitive to parameter initialization, since EM algorithm is only guaranteed to converge to the nearest local maxima. The work described by Eckart et al. [16] scales the input data to a unit cube and initializes a fixed number of Gaussian distributions uniformly over the cube. The model fitting approach proposed by Srivastava and Michael [50] converts the input 3D data to a uniformly sampled voxel grid and initializes an empirically estimated fixed number of Gaussian components over this uniform point cloud. Tabib et al. [55] also initializes a GMM with fixed complexity over each individual sensor point cloud data using "KMeans++" algorithm. Expectation-Maximization is then employed by all the approaches to refine the initialized GMM parameters.

EM aims to find a set of GMM parameters $\bar{\mathbf{\Theta}}$ that maximize the lower bound of the log-likelihood of the input 3D point cloud $\mathbf{X}$ having been sampled by the GMM $\bar{\mathbf{\Theta}}$. A set of latent variables $\mathbf{c}$ is introduced that represent the log-likelihood contribution of each Gaussian distribution for each point. Given an initial estimate of the parameters $\bar{\mathbf{\Theta}}$:

- Expectation step first computes the expected value of the log-likelihood of $\mathbf{X}$ using the current conditional distribution $\mathbf{c}$

$$\mathbf{c}_{nm} = \frac{\pi_m \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right)}{\sum_{j=1}^{M} \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{2.1}$$

- Maximization step then computes the set of parameters $\bar{\mathbf{\Theta}}$ that maximize the expected log-

likelihood

$$\boldsymbol{\mu}_m^{i+1} = \frac{\sum_n^N \mathbf{c}_{nm}\mathbf{x}_n}{\sum_n^N \mathbf{c}_{nm}} \tag{2.2}$$

$$\boldsymbol{\Sigma}_m^{i+1} = \frac{\sum_n^N \mathbf{c}_{nm}\left(\mathbf{x}_n - \boldsymbol{\mu}_m^i\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_m^i\right)^T}{\sum_n^N \mathbf{c}_{nm}} \tag{2.3}$$

$$\pi_m^{i+1} = \sum_n^N \frac{\mathbf{c}_{nm}}{N} \tag{2.4}$$

- Check if the log-likelihood has converged

$$\ln p(\mathbf{X}, \bar{\boldsymbol{\Theta}}) = \sum_n^N \ln\left(\sum_m^M \pi_m \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right)\right) \tag{2.5}$$

The algorithmic complexity of EM is $\mathcal{O}(MNK)$ where $M$ is the model complexity of the GMM, $N$ is the number of points in the input data $\mathbf{X}$ and $K$ is the maximum number of iterations EM is executed for. High computational complexity of EM poses unique challenges in terms of real-time implementation on SWaP constrained systems. Various heuristic based approximations have been proposed to reduce the computational complexity of EM, however, each approach has its set of drawbacks. As EM cannot guarantee convergence to global minima, parameter initialization becomes an equally important problem and still remains an open challenge. Further, EM requires the model complexity to be known a-priori, which is not available in real-world scenarios. The state-of-the-art mapping approaches mentioned above aim to fit the best parametric distribution to the input 3D point cloud data obtained from LIDAR or Kinect like sensors. As discussed in Sec. 2.1.1 this data is spread along the surfaces of objects in the scene and therefore lie on a lower dimensional manifold in 3D. The generic nature of EM makes it harder to enforce the structure of the input data on the fitting procedure.

In this work, we use a similar generative map representation with a key difference: instead

of representing our map as a GMM, we represent the map as a set of independent hierarchical Gaussian distributions. Similar to NDT mapping approach, each Gaussian distribution represents some local sensor information. However, we do not force a Gaussian distribution to lie within a voxel and fit the Gaussian parameters, depending upon the input data. Further, by enabling Gaussian distributions to be independent of each other, we reduce the computational complexity of fitting parameters unlike EM. Instead of using a generic EM like approach, we using pixel-space search based model estimation approach to fit Gaussian distributions to the structure in the scene. Therefore, our approach is able to fit more accurate Gaussian distributions to the scene with lower computational complexity than the approaches described above. Additionally, Gaussian distributions fit according to the structure information in the scene, have smaller memory footprint than map representations such as meshes and surfels.

### 2.1.5 3D Deep Learning and Neural Representations

Commonly used 3D representation described above such as voxel grids, surfels, meshes, or point clouds, are capable of reconstructing the observed map at high fidelity and generalize to any complex shape, subject to the resolution constraints. Generative mapping approaches such as NDTMap or GMMs approximate the 3D structure data as normal distributions to trade-off the accuracy of reconstruction for compactness of map storage. These approaches attempt to approximate the underlying continuous function that sensor measurements are sampled from, by combining multiple simplified representations. More recently, Deep Implicit Functions (DIF) proposed in [33, 38], represent input data as latent feature vectors and estimate an occupancy grid or a Signed Distance Field. These approaches are capable of reconstructing small scale 3D objects at high resolution. However, these approaches are not generalizable and do not scale for large scale environments.

A Recurrent Neural Network based map representation is proposed by Henriques and Vedaldi [22]. Each sensor measurement is projected on the ground plane in a discrete grid stored as a

allocentric spatial memory. Each sensor measurement is converted into an implicit representation and registered to a 2D map for accurate path planning. DeepVoxels [47] proposed a hybrid 2D/3D approach that condenses the input data into a latent representation but stores the scene information in a fixed sized spatial voxel structure. This idea was extend by Sitzmann et al. [48] to be able to model the 3D scene geometry and render color images of the scene without storing the map as a fixed resolution map. A Scene Representation Network(SRN) is proposed that represents the 3D data in the scene as a continuous function that maps the input sensor information to a $n$D feature. The resolution of this representation is limited by the capacity of the multi-layer perceptron used as the feature mapping.

These approaches convert the 3D structure information into high dimensional feature vectors. The feature information is typically stored as voxel grids [31, 47] or as continuous functions [48]. However, these approaches do not scale to large scale environments. The high computational complexity renders them infeasible for deployment on SWaP constrained systems. Further, learning based mapping approaches do not generalize well to environments outside their training datasets.

## 2.2 Simultaneous Localization and Mapping

The state-of-the-art depth based SLAM techniques enable high fidelity and accurate 3D reconstruction of a limited scale environments on Desktop grade GPUs. A typical SLAM framework can be interpreted as an Expectation-Maximization (see Sec. 2.1.4) routine. The aim of this expectation maximization is to estimate the map parameters $\mathbf{\Theta}$ and the trajectory of the sensor $\{\mathbf{T}_t\}$, that best explain the history of observed sensor information. As a stream of sensor observations is obtained, at each time step $t$:

- Localization (Expectation) step: finds the set of parameters $\mathbf{T}_t$ that maximize the log-likelihood of the history of sensor scans at time $t$, $\mathbf{X}_t$ having been generated from the global map $\mathbf{\Theta}$

- Mapping (Maximization) step: computes the best estimate of the map parameters $\mathbf{\Theta}$, with respect to the stream of sensor observation observed until time $t$, and the history of sensor pose $\mathbf{T}_t$

In this section, we discuss some state-of-the-art SLAM techniques that follow a similar structure mentioned above with various definitions of $\mathbf{\Theta}$. We also draw comparisons to the proposed SLAM approach and highlight the key differences.

### 2.2.1 Volumetric SLAM

The advent of high resolution, high frequency hand-held RGBD sensors like the Kinect, enables a plethora of robotics applications. Dense depth camera based SLAM approach presented by Izadi et al. [26] was a milestone in dense SLAM that drove a wide variety of research in dense RGBD based SLAM. Similar to the voxel grids described in Sec. 2.1.2 a fixed sized map representation of the world is maintained and updated as novel sensor observations are obtained. However, instead of saving the probability of occupancy, the proposed map representation stores the distance of each voxel to the closest observed surface as the voxel value. This map representation is also known as a Truncated Euclidean Signed Distance Field or simply a Truncated Signed Distance Field (TSDF). The TSDF enables elegant incorporation of noisy sensor information into a incremental map representation. The surfaces in the scene can be extracted by computing the zero crossing along each sensor ray. However, this map representation is not readily usable for localization. At each time step $t$, points along the observed surfaces are extracted using zero crossing from the current TSDF and a dense triangulated mesh of the world is reconstructed. The current sensor observation $\mathbf{X}_t$ is then aligned with respect to this intermediate map representation using a variant of the well known Iterative Closest Point (ICP) algorithm [4, 44]. At each iteration $i$ of ICP:

- a set of correspondences are computed between points in $\mathbf{X}_t$ and the planar triangles in the intermediate map representation $\mathbf{\Theta}$

- a cost function is formulated as the sum of distances of each point to its corresponding triangle, along the direction of the normal of the triangle
- a Gauss-Newton routine is employed to find a set of transformation parameters that minimizes this cost function

This approach enables 3D reconstruction of dense complex environments on a Desktop grade GPU from noisy depth sensor data as shown in Fig. 2.4. However, the TSDF map representation requires a high resolution voxel grid and therefore, KinectFusion has large memory footprint. Further, maintaining two different map representations for mapping and localization creates redundancy in the system and increases the computational and memory complexity of the proposed SLAM pipeline.

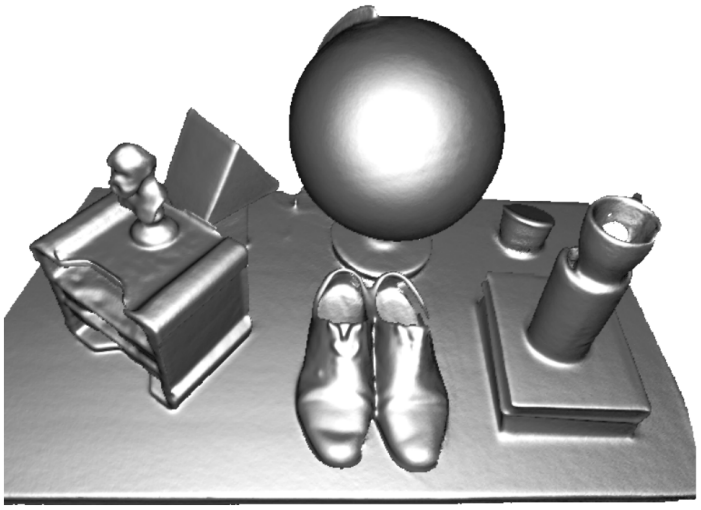


*Figure 2.4: Illustrative example of 3D reconstruction obtained from KinectFusion SLAM framework.* ***Source*** *[26].*

### 2.2.2 Surface Primitive Based SLAM

Instead of maintaining and updating two redundant map representations in real-time another family of dense SLAM technique aims to reconstruct the 3D surfaces in the scene explicitly and update the explicit surface map. A novel approach proposed by Keller et al. [29] uses a set of discrete

unordered 3D points as the map representation. Since the sensor input is obtained as 3D point clouds, this representation can provide a very high fidelity 3D reconstruction of the scene. Similar to KinectFusion, this approach attempts to minimize the distance between a point in the live sensor scan and its corresponding point in the global map, along the direction of its normal. Correspondences are computed using a computer graphics technique known as "surface-splatting". "Surface-splatting" creates a disk of set radius, at a given 3D location and a corresponding normal direction and replaces a 3D point by a 3D disk to provide more robust correspondences. The normal of each global map point is computed using central differences of the de-noised neighboring points. The final map reconstruction is obtained by triangulating the reconstructed 3D points and creating a connected mesh, as post-processing. Maintaining discrete set of points as the global map representation enables the authors to reason about sensor noise and dynamic objects by independently weighing each point using temporal information. Points observed over longer time duration are weighted higher and considered a part of the static map, while points with lower weights are considered dynamic or noisy points and discarded from the map. Further, due to the independent nature of the map representation, this approach is highly parallelizable. However, by definition 3D points do not have any area, and therefore, infinite points are required to be able to accurately reconstruct a dense surface. This increases the memory footprint of the algorithm and restricts its applications to small scale environments.

Extending upon this idea of "surface-splatting", Whelan et al. [58] proposed a SLAM pipeline, ElasticFusion, similar the point based approach, but replaced the surface primitives from 3D points to 3D planar disk like surface elements (surfels). Surfels are parameterized by their 3D position, radius, normal and color. Along with minimizing the distance between incoming sensor points and corresponding surfels, they incorporate the difference between the intensities of the raw sensor scan and a rendered image of the map into the localization cost function. Incorporating the intensity information enables the localization framework to be more robust to environments where depth

data is not informative enough to compute a unique solution (see Fig. 2.5). Along with providing a more robust localization cost function, the authors introduced a concept commonly used in the graphics community, known as "deformation graphs" to enable global map consistency and correct for local drifts in the localization pipeline. When a loop closure is detected by the system, the entire global trajectory is corrected for, by propagating the correction throughout the deformation graph and correcting the reconstructed map. Since the surfels are solid objects with a well-defined area, they can directly be used to represent the final 3D reconstruction of the scene without further post-processing. However, surfels have a very small radius and only capture a small amount of 3D neighborhood information. Further, due to the 2D disk structure, surfels lose the local structural properties and therefore, noisy sensor information often gets incorporated as high frequency noisy surfels. Although surfels are more memory efficient that TSDFs, they are still very expensive to store and maintain, numbering in millions for small room-scale environments. Exploiting the

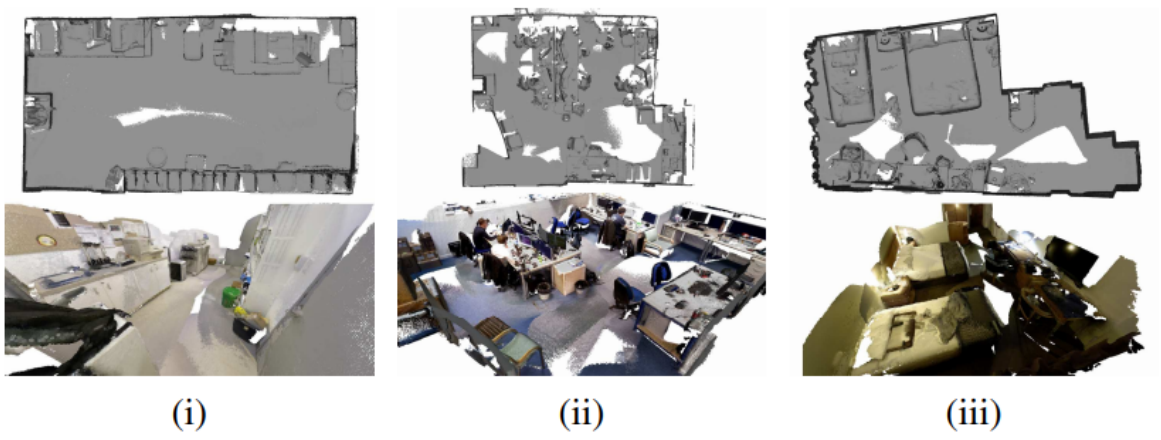



*Figure 2.5: Illustrative example of 3D reconstruction obtained from ElasticFusion SLAM framework.* ***Source*** *[58].*

highly structured nature of indoor environments, Kaess [28] presented a SLAM approach, Planar SLAM, that builds upon the observation that most indoor environments dominantly consists of large planar regions. Extending the idea of representing the sensor information with small planar

elements, the authors fit large 3D planes to the dominant planes in the scene while ignoring the surfaces with large curvature. A Gauss-Newton routine is employed similar to KinectFusion, ElasticFusion, to perform point-to-plane distance minimization, for frame-to-model localization. This work demonstrates the utility of exploiting the structure in the scene to create efficient and succinct representations of the scene. However, since this approach only reconstructs planes in the scene, it is not usable for high fidelity 3D reconstruction and in outdoor complex environments.

## 2.3 Comparison to the Proposed Approach

Our proposed hierarchical SLAM approach reconstructs a 3D map of the world that can be utilized for various robotic applications. Similar to the hierarchical mapping approaches described by Srivastava and Michael [50], and Eckart et al. [16], our proposed approach reconstructs a generative 3D model of the world. However, our approach attempts to generate a high fidelity 3D reconstruction of the world similar to the dense SLAM approaches described in Sec. 2.2 and have the compression capabilities of approaches similar to OctoMap, NDTMap and Planar SLAM.

Unlike the hierarchical approach described by Eckart et al. [16] the proposed mapping framework employs a bottom-up hierarchical map reconstruction, where a high-fidelity map $\mathbf{\Theta}^0$ is first fitted to the input data without any iterative optimization like EM. This high-fidelity map representation is then operated over to obtain a lower fidelity map representation and so on. This enables us to compute the model complexity of the world at different hierarchical levels from the raw data instead of using an arbitrary fixed value. Similar to an OctoMap structure, our map representation is a set of tree like hierarchical structures, where each tree is independent of the other, thus retaining the computational benefits of a surfel based SLAM approach, and is able to fit high level features like planes at the highest hierarchical level, similar to Planar SLAM.

Further, the model fitting technique described in this thesis exploits the understandings of the working of a depth sensor as described in Sec 2.1.1 to fit parametric models to sensor data that

attempt to maximize the accuracy of the map representation. Under hard constraints, the model fitting approach described by Tabib et al. [55] converges to the model fitting technique proposed in this thesis. Thus, our approach is a lower bound of EM. We argue that, since the data obtained from depth sensors is indeed not IID, the assumptions made in this work are valid and help reduce the computational cost of fitting arbitrary resolution Gaussian distributions to raw sensor data.

# Chapter 3

# Background

This chapter provides a high-level overview of concepts used in this thesis to enable SLAM using Gaussian distributions as structure primitives.

## 3.1 Gaussian Distribution

A Gaussian distribution is a parametric representation of a set of random $d$-dimensional points ${}^{a}\mathbf{X}$, in a coordinate frame $a$, defined by the sufficient statistics of the data, the mean ${}^{a}\boldsymbol{\mu}$, and the covariance ${}^{a}\boldsymbol{\Sigma}$. Mathematically, we define a Gaussian distribution ${}^{a}\boldsymbol{\theta}$ as,

$$ {}^{a}\mathbf{X} \sim \mathcal{N}\left({}^{a}\boldsymbol{\mu}, {}^{a}\boldsymbol{\Sigma}\right) \tag{3.1} $$

$$ {}^{a}\boldsymbol{\theta} := \{{}^{a}\boldsymbol{\mu}, {}^{a}\boldsymbol{\Sigma}\} \tag{3.2} $$

The mean ${}^{a}\boldsymbol{\mu}$ and the covariance ${}^{a}\boldsymbol{\Sigma}$ are defined as,

$$ {}^{a}\boldsymbol{\mu} = \mathbb{E}[{}^{a}\mathbf{X}] = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N} \tag{3.3} $$

$$ {}^{a}\boldsymbol{\Sigma} = \mathbb{E}[({}^{a}\mathbf{X} - {}^{a}\boldsymbol{\mu})({}^{a}\mathbf{X} - {}^{a}\boldsymbol{\mu})^{T}] = \frac{\sum_{i=1}^{N} ({}^{a}\mathbf{x}_i - {}^{a}\boldsymbol{\mu})({}^{a}\mathbf{x}_i - {}^{a}\boldsymbol{\mu})^{T}}{N} \tag{3.4} $$

A covariance matrix ${}^{a}\mathbf{\Sigma}$, by definition is symmetric and Positive Semi-Definite (PSD), (See Eq. 3.4). The Singular Value Decomposition of a symmetric square matrix $\mathbf{M}$ is

$$\mathbf{M} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{T} \tag{3.5}$$

where, $\mathbf{U}$ is an orthogonal matrix whose columns are the eigenvectors if $\mathbf{M}$, and $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the eigenvalues of $\mathbf{M}$. For a PSD matrix, all the eigenvalues are non-negative.

The probability of sampling a point ${}^{a}\mathbf{x}_i$ from a Gaussian distribution ${}^{a}\boldsymbol{\theta} := \{{}^{a}\boldsymbol{\mu}, {}^{a}\mathbf{\Sigma}\}$ is

$$p\left({}^{a}\mathbf{x}_i \mid {}^{a}\mathbf{\Theta}\right) = \sqrt{(2\pi)^{d}|{}^{a}\mathbf{\Sigma}|}^{-1} \exp\left(-0.5\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)^{T} {}^{a}\mathbf{\Sigma}^{-1}\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)\right) \tag{3.6}$$

and the log-likelihood of ${}^{a}\mathbf{x}_i$ is,

$$\begin{aligned} l\left({}^{a}\mathbf{x}_i \mid {}^{a}\boldsymbol{\theta}\right) &= \log\left(p\left({}^{a}\mathbf{x}_i \mid {}^{a}\mathbf{\Theta}\right)\right) && (3.7)\\ &= \log\left(\sqrt{(2\pi)^{d}|{}^{a}\mathbf{\Sigma}|}^{-1}\right) - 0.5\left(\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)^{T} {}^{a}\mathbf{\Sigma}^{-1}\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)\right) && (3.8)\\ &= -\frac{1}{2}d\log(2\pi) - \frac{1}{2}\log|{}^{a}\mathbf{\Sigma}| - \frac{1}{2}\left(\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)^{T} {}^{a}\mathbf{\Sigma}^{-1}\left({}^{a}\boldsymbol{\mu} - {}^{a}\mathbf{x}_i\right)\right) && (3.9) \end{aligned}$$

## 3.2 Transformation Parameterization

A rigid body transformation in $\mathbb{R}^3$ space can be represented in various ways. It has $3$ degrees of freedom in rotation and $3$ degrees of freedom in translation. The translational part of the transformation is linear and is simple to use for optimizations. However, there are multiple representations of the rotational component: Euler angles, quaternions and rotation matrices representing rotation in Lie group $\mathbb{SO}(3)$. Each rotation matrix represents a unique rotation and the representation does not have any singularities [25]. In this work, we represent the rigid body transformation as an

element of the Lie group $\mathbb{SE}(3)$.

## 3.3 Lie Groups

Seminal documentation by Eade [15] provides detailed description about Lie groups their corresponding Lie algebra for 2D and 3D rigid body transformations. In this section, we provide a brief summary on $\mathbb{SE}(3)$ and $\mathbb{SO}(3)$ Lie groups.

Lie groups and lie algebra are extensively used in robotics and computer vision because of their special geometric properties. Lie algebra provides a coherent and a robust framework for working with 3D rigid body transformations.

### 3.3.1 Rigid Body Rotations

Every Lie Group has a corresponding Lie algebra that we denote by the corresponding lowercase representation. For the rotation Lie group $\mathbb{SO}(3)$, its corresponding Lie algebra is $\mathbf{so}(3)$. The elements of $\mathbf{so}(3)$ are a set of $3 \times 3$ skew-symmetric matrices and the elements of $\mathbb{SO}(3)$ are represented by $3 \times 3$ matrices and define rotation of rigid bodies. Elements of the group can be added by matrix multiplication and an inverse rotation can be applied by inverting the rotation matrix. A rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$ is orthogonal. Therefore,

$$\mathbf{R}^{-1} = \mathbf{R}^T, \mathbf{R} \in \mathbb{SO}(3) \tag{3.10}$$

Given a random vector $\boldsymbol{\omega} \in \mathbb{R}^3$, an element of $\mathbf{so}(3)$ can be written as a linear combination of the generators:

$$\omega_1 \mathbf{G}_1 + \omega_2 \mathbf{G}_2 + \omega_3 \mathbf{G}_3 \in \mathbf{so}(3) \tag{3.11}$$

where $\mathbf{G}_i$ are the generators of $\mathbb{SO}(3)$ [15]. For the sake of convenience we write $\boldsymbol{\omega} \in \mathbf{so}(3)$ and define $\boldsymbol{\omega}_\times$ to represent an element of $\mathbf{so}(3)$ that is the skew-symmetric matrix of $\boldsymbol{\omega}$ similar to [15].

### 3.3.2 Rigid Body Transformations

A group of rigid body transformations in 3D space belonging to $\mathbb{SE}(3)$ Lie group, is represented by rotation matrices $\mathbf{R} \in \mathbb{SO}(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$, as a homogeneous transformation:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}(3) \tag{3.12}$$

An element of $\mathbf{se}(3)$ can be represented as:

$$(\boldsymbol{\omega}, \mathbf{u})^T = \boldsymbol{\phi} \in \mathbb{R}^6 \tag{3.13}$$

$$\omega_1\mathbf{G}_1 + \omega_2\mathbf{G}_2 + \omega_3\mathbf{G}_3 + u_1\mathbf{G}_4 + u_2\mathbf{G}_5 + u_3\mathbf{G}_6 \in \mathbf{se}(3) \tag{3.14}$$

For brevity, we write $(\boldsymbol{\omega}, \mathbf{u})^T = \boldsymbol{\phi} \in \mathbf{se}(3)$.

### 3.3.3 Exponential Maps

There exists a one-to-one mapping between the elements of the Lie algebra $\mathbf{se}(3)$ ($\mathbf{so}(3)$) and the elements of the Lie group $\mathbb{SE}(3)$ ($\mathbb{SO}(3)$). This closed form mapping is defined by an exponential map:

$$\boldsymbol{\phi} = (\boldsymbol{\omega}, \mathbf{u}) \in \mathbf{se}(3) \tag{3.15}$$

$$\exp(\boldsymbol{\phi}) = \exp \begin{bmatrix} \boldsymbol{\omega}_\times & \mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}(3) \tag{3.16}$$

$$= \begin{bmatrix} \exp(\boldsymbol{\omega}_\times) & \mathbf{Vu} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.17}$$

where

$$\exp(\boldsymbol{\omega}_\times) = \mathbf{I}_{3\times 3} + \frac{\sin\theta}{\theta}\boldsymbol{\omega}_\times + \frac{1-\cos\theta}{\theta^2}\boldsymbol{\omega}_\times^2 \in \mathbb{SO}(3) \tag{3.18}$$

$$\theta^2 = \boldsymbol{\omega}^T\boldsymbol{\omega}$$

is the exponential map of $\mathbf{so}(3)$ and

$$\mathbf{V} = \mathbf{I}_{3\times 3} + \frac{\sin\theta}{\theta}\boldsymbol{\omega}_\times + \frac{\theta-\sin\theta}{\theta^3}\boldsymbol{\omega}_\times^2 \tag{3.19}$$

Therefore, the mapping from elements of the Lie group $\mathbb{SE}(3)$ $(\mathbb{SO}(3))$ to elements of the Lie algebra $\mathbf{se}(3)$ $(\mathbf{so}(3))$ is given by a log map:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{SE}(3), \mathbf{R} \in \mathbb{SO}(3) \tag{3.20}$$

$$\ln(\mathbf{T}) = \begin{bmatrix} \ln(\mathbf{R}) & \mathbf{V}^{-1}\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.21}$$

where $\ln(\mathbf{R})$ is the log map of $\mathbb{SO}(3)$

$$\ln(\mathbf{R}) = \frac{\theta}{2\sin\theta}\left(\mathbf{R} - \mathbf{R}^T\right) = \boldsymbol{\omega} \tag{3.22}$$

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right) \tag{3.23}$$

$$\mathbf{V}^{-1} = \mathbf{I}_{3\times 3} - \frac{1}{2}\ln(\mathbf{R}) + \frac{1}{\theta^2}\left(1 - \frac{a}{b}\right)\ln(\mathbf{R})^2 \tag{3.24}$$

$$a = \frac{\sin\theta}{\theta}, b = 2\frac{1-\cos\theta}{\theta^2} \tag{3.25}$$

The vector $\boldsymbol{\phi} \in \mathbb{R}^6$ can be extracted as $(\boldsymbol{\omega}, \mathbf{V}^{-1}\mathbf{t}) = (\boldsymbol{\omega}, \mathbf{u})$.

When the transformation between two elements in $\mathbb{SE}(3)$ is extremely small, i.e. $\theta^2 \approx 0$, we can write $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. The exponential map defined in Eq. 3.16, Eq. 3.18 and Eq. 3.19

can be approximated as

$$\exp\left(\boldsymbol{\omega}_{\times}\right) \approx \mathbf{I}_{3\times 3} + \boldsymbol{\omega}_{\times} + 0 \tag{3.26}$$

$$\mathbf{V} \approx \mathbf{I}_{3\times 3} + \boldsymbol{\omega}_{\times} + 0 \tag{3.27}$$

$$\exp(\boldsymbol{\phi}) = \begin{bmatrix} \mathbf{I}_{3\times 3} + \boldsymbol{\omega}_{\times} & \left(\mathbf{I}_{3\times 3} + \boldsymbol{\omega}_{\times}\right)\mathbf{u} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \tag{3.28}$$

The linear relationship in Eq. 3.28 of the $\mathfrak{se}(3)$ parameters with the exponential map provides mathematical ease when optimizing a cost function with respect to transformation parameters.

## 3.4 Non Linear Least Squares

Least squares optimization is a family of techniques that attempts to find a set of parameters that minimize the sum of the squares of a quadratic cost function. Each element of the overall quadratic cost is known as a residual. When the squares of residuals are non-linear with respect to the parameters, the family of techniques used to minimize the cost function is known as non-linear least squares. In other words, the goal of non-linear least squares optimization is to maximize the likelihood of a set of observations with respect to its parameters. A detailed overview of standard approaches for solving non-linear least square optimization problems is provided by Alismail et al. [2]. In this work, to enable localization of a moving camera in an unknown environment, we employ a commonly used "Levenberg-Marquardt" optimization technique.

Given a set of $N$ observations from the world, $\mathbf{X}$, in a $d$-dimensional space, a real valued function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ parameterized by a set of vector of parameters $\boldsymbol{\theta}$, a standard least squares optimization attempts to find a local minima of the cost function:

$$\mathbf{C}\left(\boldsymbol{\theta}\right) = \frac{1}{2}\sum_{\mathbf{x}\in\mathbf{X}} f^2(\mathbf{x} \mid \boldsymbol{\theta}) \tag{3.29}$$

The function $f(\mathbf{x})$ is the residual of the optimization at point $\mathbf{x}$. Two standard techniques used to minimize the value of the cost function $\mathbf{C}(\boldsymbol{\theta})$ and compute the values of $\boldsymbol{\theta}$ at which the minima of $\mathbf{C}$ is achieved are described in Sec. 3.4.1. These approaches only compute the Jacobian of $\mathbf{C}$ and approximate the Hessian to be $\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$. This approximation is valid when the residual values of $f(\mathbf{x})$ are small at the estimated $\boldsymbol{\theta}$.

### 3.4.1 Solving Non-Linear Least Squares

Gauss-Newton and Levenberg-Marquardt are two standard approaches of solving non-linear least squares problems. Both approaches iteratively estimate the value of $\boldsymbol{\theta}$ by computing increments to the parameters at each iteration and adding the updates to the parameter vector. At each iteration, first, the least squares optimization is linearized by computing the first-order partial derivatives of $\mathbf{C}$ at the current estimate of the parameters. Second, this approximately linear system of equations is solved using an appropriate linear solver (Cholesky decomposition, QR decomposition, etc.). Third, incremental updates to the parameters are computed and added to the current estimate of the parameters. These three steps are repeated until the cost function converges or the parameters updates are infinitesimal.

Gauss-Newton and Levenberg-Marquardt only differ in the way the parameter updates are computed. At each iteration, Gauss-Newton computes the parameter updates $\delta\boldsymbol{\theta}_j$ as:

$$\mathbf{J}_j^T\mathbf{J}_j\delta\boldsymbol{\theta}_{j,GN} = -\mathbf{J}_j^T f(\mathbf{X}, \boldsymbol{\theta})_j \tag{3.30}$$

Gauss-Newton provides a very elegant iterative approach to solve a non-linear least squares problem by simply computing the first-order derivatives of the cost function $\mathbf{C}$. However, it often leads to mathematical inconsistencies when insufficient information is available for some elements of $\boldsymbol{\theta}$ making the Jacobian matrix $\mathbf{J}$ rank deficient. Thus, the incremental update $\boldsymbol{\theta}_{GN}$ lies in the null space of $\mathbf{J}$ and does not have a unique solution. Levenberg-Marquardt attempts to fix this problem

by artificially forcing the pseudo-Hessian ($\mathbf{J}^T\mathbf{J}$) to be full rank by:

$$\left(\mathbf{J}_j^T\mathbf{J}_j + \lambda\left(\text{diag}(\mathbf{J}_j^T\mathbf{J}_j)\right)\right)\delta\boldsymbol{\theta}_{j,GN} = -\mathbf{J}_j^T f(\mathbf{X},\boldsymbol{\theta})_j \tag{3.31}$$

where, $\lambda$ is a scalar parameter $> 0$. Typically, the value of $\lambda$ is adjusted at each iteration to ensure that the incremental steps $\delta\boldsymbol{\theta}$ reduce the value of $\mathbf{C}$.

# Chapter 4

# Hierarchical Gaussian Distributions for 3D Reconstruction

## 4.1 Introduction

In this chapter, we describe in detail our proposed Gaussian distribution based hierarchical map representation that leverages the sensor characteristics of a COTS depth sensor and explicitly incorporates the uncertainty of the input data.

Standard GMM fitting approaches employ a computationally expensive EM routine to estimate the parameters of a GMM ${}^{w}\bar{\mathbf{\Theta}}$ that best approximates the distribution of points ${}^{w}\mathbf{X}$ by maximizing the log-likelihood of ${}^{w}\mathbf{X}$ with respect to ${}^{w}\bar{\mathbf{\Theta}}$. As described in Sec. 2.1.4, this process is computationally expensive and the performance varies highly dependent upon the parameter initialization. In this chapter, we propose a pixel space search-based "Region Growing" technique to fit Gaussian distributions to planar surfaces in the scene thus providing a more accurate measure of reconstruction accuracy. It also enables us to achieve orders of magnitude of computational savings, thus making it feasible to fit high fidelity generative models over the input sensor point cloud data in real-time.

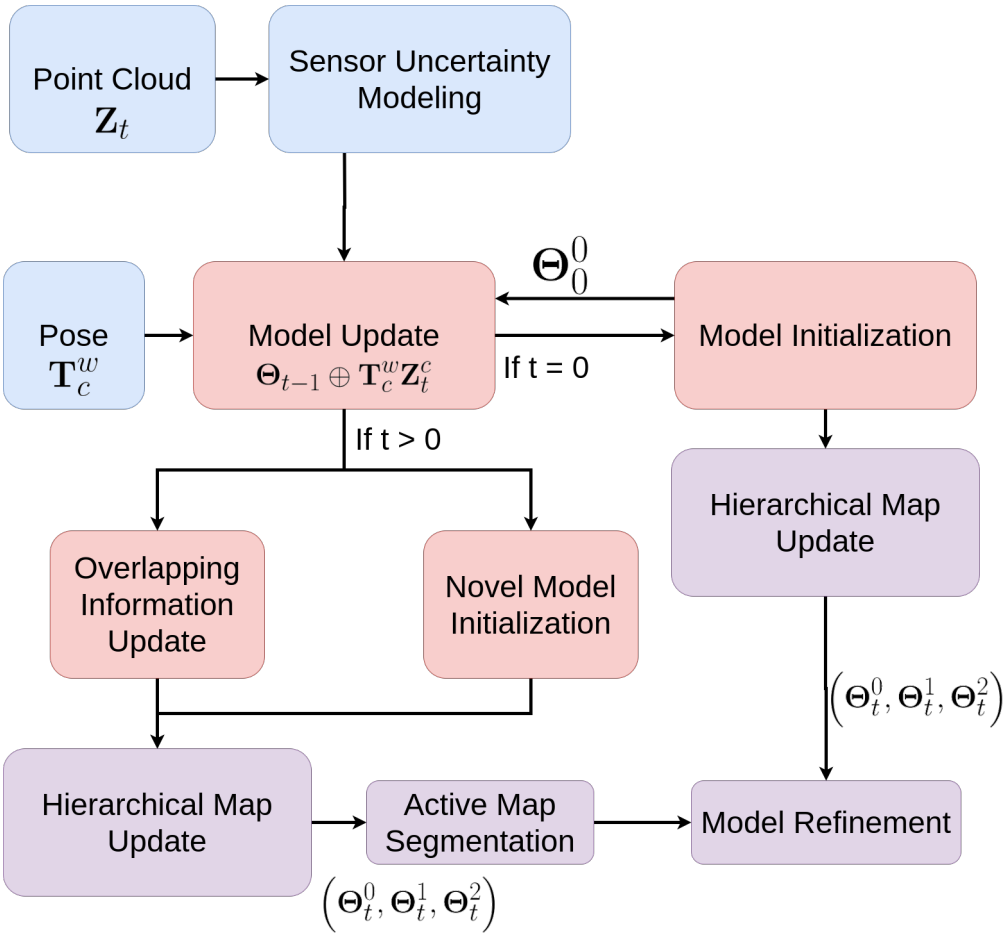
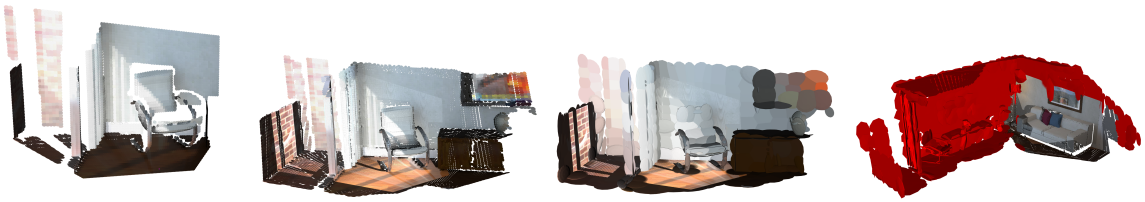



*Figure 4.1: System Overview: The proposed mapping algorithm operates on depth image streams and corresponding sensor poses. As sequential sensor measurements are obtained, the hierarchical map ${}^w\Theta_t^i$ is updated. We use projective correspondences to compute the correspondence between a point $\mathbf{x}_i$ and Gaussian distributions ${}^w\boldsymbol{\theta}_{\mathcal{C}(i)}$. Each Gaussian distribution is then updated with new points that represent the sections of the map that are already observed. Novel distributions ${}^w\boldsymbol{\theta}_i^t$ are then generated using novel information in each depth measurement. As more measurements are obtained the model is further refined by discarding noisy distributions from the global model.*

A detailed system overview of our proposed mapping approach is illustrated in Fig. 4.1. The mapping process can be summarized into the following steps:

1. Initialize a high fidelity surface representation of the environment (Sec. 4.1.1)
2. Compute a correspondence map that signifies the point-to-distribution correspondence for each point in a new depth measurement (Sec 4.1.3)
3. Fuse incrementally observed noisy sensor data into the map representation using correspon-



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

*Figure 4.2: Illustrative example of our proposed mapping framework: (a): A hierarchical Gaussian distribution based map ${}^{w}\boldsymbol{\Theta}_0^0$ is initialized at high fidelity at the given location at time $t = 0$ and hierarchical maps at level 1 and 2 are updated ${}^{w}\boldsymbol{\Theta}_0^1$, ${}^{w}\boldsymbol{\Theta}_0^2$, (b): As the camera moves in the scene, the map estimate ${}^{w}\boldsymbol{\Theta}_0^0$ is updated to ${}^{w}\boldsymbol{\Theta}_t^0$ with the novel information and overlapping evidence observed by the sensor, (c): These updates are propagated through the hierarchical structure to update ${}^{w}\boldsymbol{\Theta}_t^1$ as shown in the figure, (d): As the sensor moves further away, the distributions of ${}^{w}\boldsymbol{\Theta}_t^2$ outside the sensor FOV are labeled inactive (red) and only the active Gaussian distributions are used for further updates until inactive map sections are observed again.*

dence map and Gaussian merging (Sec. 4.1.2)

4. Update the higher hierarchical levels of the map (Sec. 4.1.7)

5. Refine the map by removing outlier Gaussian distributions (Sec. 4.1.6)

6. Segment map as active/inactive for efficient map update (Sec. 4.1.9)

### 4.1.1 Surface Model Initialization

We initialize a Gaussian distribution based representation of the environment at hierarchical level 0 as the first sensor measurement is observed at time $t = 0$. This representation is formulated as an ordered set of Gaussian distributions in a global coordinate frame $w$, ${}^{w}\boldsymbol{\Theta}_0^0 = \left\{ {}^{w}\boldsymbol{\theta}_0^0, {}^{w}\boldsymbol{\theta}_1^0, \ldots {}^{w}\boldsymbol{\theta}_n^0 \right\}$. The covariance $\boldsymbol{\Sigma}_i$ of each ${}^{w}\boldsymbol{\theta}_i^0$ signifies the spread of points that it is fit on. Depth measurements obtained from COTS structured light sensors are spread along sub-manifolds (surfaces of the objects in the scene) in the ambient $\mathbb{R}^3$ space (see Sec. 2.1.1). A ${}^{w}\boldsymbol{\theta}_i^0$ that best represents a subset of this observed data corresponds to a planar patch of the spread of points along a surface. The smallest eigenvalue $\lambda_{i,0}$ of $\boldsymbol{\Sigma}_i$ represents the variance of points along the direction with the least data variation i.e., the normal to the surface that ${}^{w}\boldsymbol{\theta}_i^0$ is fit on. The corresponding eigenvector $\mathbf{u}_{i,0}$ represents the direction of this normal. Given an image $\mathcal{I}_0$ of size $V \times U$, we use this understanding

to fit the best Gaussian distributions ${}^{w}\boldsymbol{\theta}_i^0$ to small $b_0 \times b_0$ pixel patches of $\mathcal{I}_0$ that represent planar spread of points.

Assuming pinhole camera geometry, a depth measurement at pixel location $(v, u)$, $\mathcal{I}_0(v, u)$ can be back projected in to $\mathbb{R}^3$ space using inverse projective function

$$\mathbf{x}_{v,u} = \mathbf{\Pi}^{-1}(\mathcal{I}_0(v, u), v, u) \tag{4.1}$$

Since, this can be viewed as a linear transformation, the points that are proximate in the 3D space, are also proximate in pixel space. We exploit this property of projective pinhole geometry and implement a version of the "Region Growing" algorithm proposed by Poppinga et al. [41] that operates in the image space using uncertain data. The "Region Growing" algorithm can be described as follows:

In each image patch of size $b_0 \times b_0$ pixels, we

1. Select a random seed pixel $(v_s, u_s)$ and compute the 3D point $\mathbf{x}_{v_s,u_s} = \mathbf{\Pi}^{-1}(\mathcal{I}(v_s, u_s), v_s, u_s)$
2. Search for candidate 3D points $\mathbf{x}_{v_c,u_c}$ in the pixel neighborhood of $\mathbf{x}_{v_s,u_s}$ that lie within $\alpha_n$ distance of $\mathbf{x}_{v_s,u_s}$ and initialize a Gaussian distribution ${}^{w}\boldsymbol{\theta}_i^0$ using this set of points as defined by Eq. 3.3 and Eq. 3.4
3. For candidate points $\mathbf{x}_{v_c,u_c}$ that are inside $\alpha_n$ distance, search for its neighbor points $\mathbf{x}_{v_k,u_k}$ within $\alpha_n$ distance, such that the smallest eigenvalue $\lambda_0$ of the covariance $\mathbf{\Sigma}_i$ of a Gaussian distribution ${}^{w}\boldsymbol{\theta}_i^0$ updated with point $\mathbf{x}_{v_k,u_k}$ is less than $\alpha_\lambda^2$ and the largest eigenvalue is less than $\alpha_{\text{len}}$
4. Continue until all points in the image patch are processed

The mean and covariance of a Gaussian distribution completely represent sufficient information about the points that it was fit on. Therefore, for incrementally updating a Gaussian distribution

${}^{w}\boldsymbol{\theta}_j^0$ with a point ${}^{w}\mathbf{x}_i$ we do not need to preserve the history of points used to fit ${}^{w}\boldsymbol{\theta}_j^0$:

$$ {}^{w}\boldsymbol{\theta}_j^{0\prime} := \left\{ \boldsymbol{\mu}_j^{0\prime}, \boldsymbol{\Sigma}_j^{0\prime}, N_j' \right\} \tag{4.2} $$

$$ N_j' = N_j + 1 \tag{4.3} $$

$$ \boldsymbol{\mu}_j^{0\prime} = \frac{N_j \boldsymbol{\mu}_j^0 + {}^{w}\mathbf{x}_i}{N_j'} \tag{4.4} $$

$$ \boldsymbol{\Sigma}_j^{0\prime} = \frac{N_j \left( \boldsymbol{\Sigma}_j^0 + \boldsymbol{\mu}_j^0 \boldsymbol{\mu}_j^{0T} \right) + {}^{w}\mathbf{x}\,{}^{w}\mathbf{x}^T}{N_j'} - \boldsymbol{\mu}_j^{0\prime} \boldsymbol{\mu}_j^{0\prime T} \tag{4.5} $$

For each $b_0 \times b_0$ image patch, we select the Gaussian distribution that represents the largest planar region in that patch. Thus, our initial model at hierarchical level $0$, ${}^{w}\boldsymbol{\Theta}_0^0$, consists of $\frac{V}{b_0} \times \frac{U}{b_0}$ Gaussian distributions that represent planar patches in the observed scene. As a sequence of sensor measurements is observed, a globally consistent model is updated as shown in Fig. 4.2.

### 4.1.2 Incremental Model Updates

As sequential sensor measurements are obtained at time $t$, we can refine the current model estimate at the hierarchical level $0$, ${}^{w}\boldsymbol{\Theta}_{t-1}^0$, of the scene using this incoming stream of structure information. Sequential sensor measurements obtained from a sensor moving in the world, contain some novel information about the scene and some redundant information that has already been observed. However, sensor measurements obtained from a depth sensor are often noisy and therefore unreliable. We use the redundant structure data to improve the map estimate of the map that is already observed using a weighted update and fit new Gaussian distributions to novel information. An additional uncertainty covariance $\boldsymbol{\Sigma}_k^{unc}$ is added to the Gaussian distributions to represent the average uncertainty of the points used to fit each Gaussian distribution, ${}^{w}\boldsymbol{\theta}_{i-1}^k := \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\Sigma}_k^{\text{unc}}, N_k\}$.

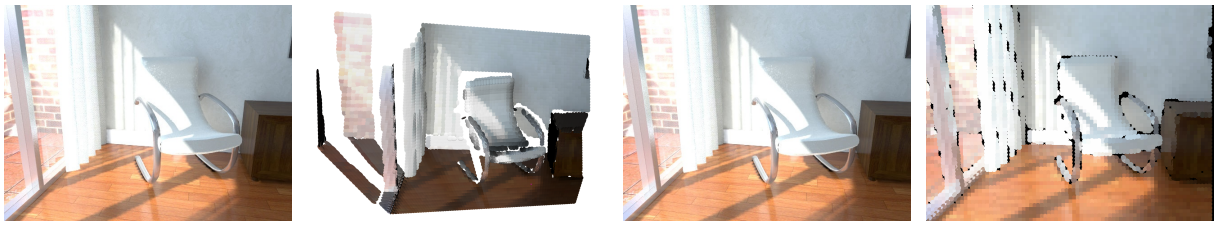
Given a sensor measurement ${}^{w}\mathbf{x}_{v,u}$ at a pixel location $(v, u)$ in image $\mathcal{I}_t$ with Gaussian uncertainty ${}^{w}\boldsymbol{\Sigma}_{v,u}^{\text{unc}}$, we check if this point has already been represented by a distribution in ${}^{w}\boldsymbol{\Theta}_{t-1}^0$ if ${}^{w}\mathbf{x}_{v,u}$ lies within a $\alpha_{\text{conf}}$ probability bound of a distribution ${}^{w}\hat{\boldsymbol{\theta}}_k^0 := \left\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k + \boldsymbol{\Sigma}_{v,u}^{\text{unc}} + \boldsymbol{\Sigma}_k^{\text{unc}}\right\}$. If

${}^{w}\mathbf{x}_{v,u}$ lies within this confidence interval then it is considered to be partially represented by ${}^{w}\boldsymbol{\theta}_{k}^{0}$ and therefore, is classified as non-novel. Points ${}^{w}\mathbf{x}_{k,l}$ that do not lie within the confidence interval of any of the distributions in ${}^{w}\boldsymbol{\Theta}_{i-1}^{0}$ are labeled as novel points.

The novel information obtained in $\mathcal{I}_t$ is used to initialize the new Gaussian distributions $\boldsymbol{\theta}_{k}^{0}$ at hierarchical level $0$ that represent the newly observed surfaces in the scene (see Sec. 4.1.1).

### 4.1.3 Correspondence Map

In order to update the global map of the world ${}^{w}\boldsymbol{\Theta}_{t-1}^{0}$ with the correct overlapping 3D point ${}^{w}\mathbf{x}_{v,u}$, we must check if the uncertainty distribution of ${}^{w}\mathbf{x}_{v,u}$ overlaps with each uncertain distribution in ${}^{w}\boldsymbol{\Theta}_{t-1}^{0}$. As the number of components in ${}^{w}\boldsymbol{\Theta}_{t-1}^{0}$ increases, the search space for finding the correspondence between a point $\mathbf{x}_{v,u} \in \mathcal{I}_t$ and distribution ${}^{w}\boldsymbol{\theta}_{k}^{t-1} \in {}^{w}\boldsymbol{\Theta}_{t-1}^{0}$ also increases. This search becomes computationally intractable in real-time for each pixel in $\mathcal{I}_t$ for a large map. To reduce this search space, we exploit the geometric properties of a Gaussian distribution and the linearity of pinhole projection. Gaussian distributions have a infinite support space in 3D. However, the probability distribution taper off exponentially farther away from the mean. Equipotential contours of a Gaussian distribution can geometrically be represented as ellipsoids in $\mathbb{R}^3$ space. The pinhole projection of a 3D ellipsoid is an ellipse in a 2D image plane [12]. The Gaussian distributions corresponding to ellipses that project at a given pixel location $(v, u)$ on the image plane are potentially proximate to the 3D point $\mathbf{x}_{v,u}$ and the Gaussian distributions that do not project in the proximity of pixel $(v, u)$ are not in the vicinity of point $\mathbf{x}_{v,u}$ and therefore have a negligible contribution to the log-likelihood of $\mathbf{x}_{v,u}$ and can therefore be ignored. For each pixel in $\mathcal{I}_t$ we project the 3D Gaussian distributions in ${}^{w}\boldsymbol{\Theta}_{t-1}^{0}$ at hierarchical level $0$ in to image space, where each pixel stores the index $k$ of ${}^{w}\boldsymbol{\theta}_{k}^{0}$ that projects at the corresponding pixel, using pinhole geometry and a depth buffer implemented in OpenGL. We detail the procedure for projection of a Gaussian distribution ${}^{w}\boldsymbol{\theta}_{k}^{0}$ on to an image plane in Sec. 4.1.4. We extract an ellipsoid for each ${}^{w}\boldsymbol{\theta}_{k}^{0}$



(a) Input sensor observation, $\mathcal{I}_t$ (b) Fitted model $\boldsymbol{\Theta}_t^0$ (c) Sensor observation $\mathcal{I}_{t+1}$ (d) Correspondence map between $\boldsymbol{\Theta}_t^0$ and $\mathcal{I}_{t+1}$

*Figure 4.3: Illustrative example of projective correspondence framework. (a): RGB image of the scene observed at time t, (b): The colored model ${}^w\boldsymbol{\Theta}_t^0$ learned on $\mathcal{I}_t$, (c): RGB image of the scene observed at time $t+1$, (d): RGB colored correspondence map created using model ${}^w\boldsymbol{\Theta}_t^0$. As sequential sensor measurements are observed, individual Gaussian distributions ${}^w\boldsymbol{\theta}_t^0 \in {}^w\boldsymbol{\Theta}_t^0$ are projected in to the current image frame using pinhole projective geometry as described in Sec. 4.1.3 to compute point-to-model correspondence. Each pixel value in the correspondence image refers to the index k of the distribution ${}^w\boldsymbol{\theta}_t^0$ that projects at that location.*

representing $99.97\%$ Chi square probability for each distribution, and project it in an image plane as shown in Fig. 4.3. Finally, to verify if the projective correspondence is correct, we perform the same check as described in Sec. 4.1.2 to ensure that the point $\mathbf{x}_{v,u} \in \mathcal{I}_t$ is within an $\alpha_{\text{conf}}$ bounds of ${}^w\boldsymbol{\theta}_k^0$ that was projected at the same pixel location.

### 4.1.4 Projecting a Gaussian Distribution on an Image Plane

A Gaussian distribution ${}^w\boldsymbol{\theta}_k^0$,

$$(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) = 1$$

represents an ellipsoid in $\mathbb{R}^3$ space. The Eigen decomposition of $\boldsymbol{\Sigma}_k^{-1}$ yields:

$$(\mathbf{x} - \boldsymbol{\mu}_k)^T \left( \mathbf{U}_k \begin{bmatrix} \boldsymbol{\sigma}_1^2 & 0 & 0 \\ 0 & \boldsymbol{\sigma}_2^2 & 0 \\ 0 & 0 & \boldsymbol{\sigma}_3^2 \end{bmatrix} \mathbf{U}_k^T \right)^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) = 1 \tag{4.6}$$

$$(\mathbf{x}-\boldsymbol{\mu}_k)^T\,\mathbf{U}_k\begin{bmatrix}\frac{1}{\boldsymbol{\sigma}_1^2} & 0 & 0\\ 0 & \frac{1}{\boldsymbol{\sigma}_2^2} & 0\\ 0 & 0 & \frac{1}{\boldsymbol{\sigma}_3^2}\end{bmatrix}\mathbf{U}_k^T\,(\mathbf{x}-\boldsymbol{\mu}_k)=1 \tag{4.7}$$

$$\tag{4.8}$$

For a Chi squared bound of $99.97\%$ when the diagonal matrix is scaled by a factor of $3$, i.e.

$$(\mathbf{x}-\boldsymbol{\mu}_k)^T\,\mathbf{U}_k\begin{bmatrix}\frac{1}{9\boldsymbol{\sigma}_1^2} & 0 & 0\\ 0 & \frac{1}{9\boldsymbol{\sigma}_2^2} & 0\\ 0 & 0 & \frac{1}{9\boldsymbol{\sigma}_3^2}\end{bmatrix}\mathbf{U}_k^T\,(\mathbf{x}-\boldsymbol{\mu}_k)-1=0 \tag{4.9}$$

$$\tag{4.10}$$

The general equation of an ellipsoid in $\mathbb{R}^3$ is [46]:

$$\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}^T\mathbb{T}^{-T}\mathbf{D}\mathbb{T}^{-1}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}=0 \tag{4.11}$$

where

$$\mathbf{D}=\text{Diag}\begin{bmatrix}1 & 1 & 1 & -1\end{bmatrix}$$

Comparing Eq. 4.10 and Eq. 4.11,

$$\mathbb{T}=\begin{bmatrix}\mathbb{I}_{3\times3} & -\boldsymbol{\mu}_k\\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{U}_k & 0\\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}3\sqrt{\sigma_1^2} & 0 & 0 & 0\\ 0 & 3\sqrt{\sigma_2^2} & 0 & 0\\ 0 & 0 & 3\sqrt{\sigma_3^2} & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4.12}$$

We use this parameterization to splat the Gaussian component to image space as presented in [46]. An OpenGL vertex shader determines the axis-aligned bounds of the projected component in clip space exploiting tangent planes. A fragment shader is then used to determine if a ray from the camera space intersects the transformed ellipsoid within these billboard locations. This leads to a much more accurate projection of the ellipsoid than an approach that generates a polygonal approximation of the ellipsoid in a vertex shader and then paints it in. Since the components are rendered from back to front, visibility is implicitly taken into account to ensure that the closest distribution is used for point correspondence. We use an inverse depth buffer to increase the effective depth resolution of projected components and thus prevent z-fighting of components that are far away from the sensor.

### 4.1.5 Gaussian Distribution Update

If an uncertain point $\left\{ {}^{w}\mathbf{x}_{v,u}, {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} \right\}$ has been previously partially observed by a distribution ${}^{w}\boldsymbol{\theta}^{0}_{k}$, ${}^{w}\boldsymbol{\theta}^{0}_{k}$ can be interpreted as the prior probability of a 3D point being sampled around ${}^{w}\mathbf{x}_{v,u}$. We can therefore compute the posterior probability as a product of two Gaussian distributions, given by

$$
\begin{aligned}
{}^{w}\hat{\mathbf{x}}_{v,u} &= \boldsymbol{\Sigma}_{k} \left( \boldsymbol{\Sigma}_{k} + {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} \right)^{-1} {}^{w}\mathbf{x}_{v,u} && (4.13) \\
&= {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} \left( \boldsymbol{\Sigma}_{k} + {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} \right)^{-1} \boldsymbol{\mu}^{k}_{i-1} && (4.14) \\
{}^{w}\hat{\boldsymbol{\Sigma}}^{\text{unc}}_{v,u} &= \boldsymbol{\Sigma}_{k} \left( \boldsymbol{\Sigma}_{k} + {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} \right)^{-1} {}^{w}\boldsymbol{\Sigma}^{\text{unc}}_{v,u} && (4.15)
\end{aligned}
$$

The posterior distribution ${}^{w}\hat{\mathbf{X}}_{v,u} := \left\{ {}^{w}\hat{\mathbf{x}}_{v,u}, {}^{w}\hat{\boldsymbol{\Sigma}}^{\text{unc}}_{v,u} \right\}$ defines the most likely estimate of the partially observed noisy point. We can add this point to the current estimate of ${}^{w}\boldsymbol{\theta}^{k}_{i-1}$ and refine its parameters as

$$
\hat{\boldsymbol{\mu}}_{k} = \frac{\boldsymbol{\mu}_{k} N_{k} + {}^{w}\hat{\mathbf{x}}_{v,u}}{N_{k} + 1} \tag{4.16}
$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\left(\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T\right) N_k + {}^w\hat{\mathbf{x}}_{v,u}\, {}^w\hat{\mathbf{x}}_j^T}{N_k + 1} - \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T \tag{4.17}$$

$$N_k = N_k + 1 \tag{4.18}$$

$$\boldsymbol{\Sigma}_k^{\text{unc}} = \boldsymbol{\Sigma}_k^{\text{unc}} + {}^w\hat{\boldsymbol{\Sigma}}_{v,u}^{\text{unc}} \tag{4.19}$$

If the sensor uncertainty model is perfectly known, the incremental update defined in Eq. 4.16 − Eq. 4.19 converges to an accurate reconstruction of the structure in the scene with less sensor information. However, in practice, if the uncertainty model is not well known, we perform an additive update instead defined in Eq. 4.20. The map representation requires more data to converge to a correct representation in this case, however, it does not converge to an incorrect estimate.

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_k &= \frac{\boldsymbol{\mu}_k N_k + {}^w\mathbf{x}_{v,u}}{N_k + 1} \\
\hat{\boldsymbol{\Sigma}}_k &= \frac{\left(\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T\right) N_k + {}^w\mathbf{x}_{v,u}\, {}^w\mathbf{x}_{v,u}^T}{N_k + 1} - \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^T \\
N_k &= N_k + 1 \\
\boldsymbol{\Sigma}_k^{\text{unc}} &= \boldsymbol{\Sigma}_k^{\text{unc}} + {}^w\boldsymbol{\Sigma}_{v,u}^{\text{unc}}
\end{aligned} \tag{4.20}$$

### 4.1.6 Model Refinement

Sensor measurements obtained from RGBD sensors often contain spurious depth measurements due to reflective objects in the scene, noisy particles or just random noise. If a distribution ${}^w\boldsymbol{\theta}_k^0$ is fit to such noisy data, the proposed projective correspondence computation pipeline fails as these components occlude the true map distributions. However, due to the spurious nature of such measurements, sequential sensor measurements do not provide overlapping evidence for the noisy distributions, ${}^w\boldsymbol{\theta}_k^0$. The pixel locations where ${}^w\boldsymbol{\theta}_k^0$ is projected are rejected as candidate correspondences by our correspondence refinement step. If the rendered depth measurements at these pixel locations are less than the observed depth measurements, the number of points represented by ${}^w\boldsymbol{\theta}_k^0$,

$N_k$ is set to $N_k - 1$. Distributions ${}^w\boldsymbol{\theta}_k^0$ that do not have sufficient evidence $N_k < \alpha_N$ are eventually discarded thus eliminating spurious and noisy distributions from the hierarchical level 0 map ${}^w\boldsymbol{\Theta}_t^0$.

### 4.1.7 Hierarchical Mapping

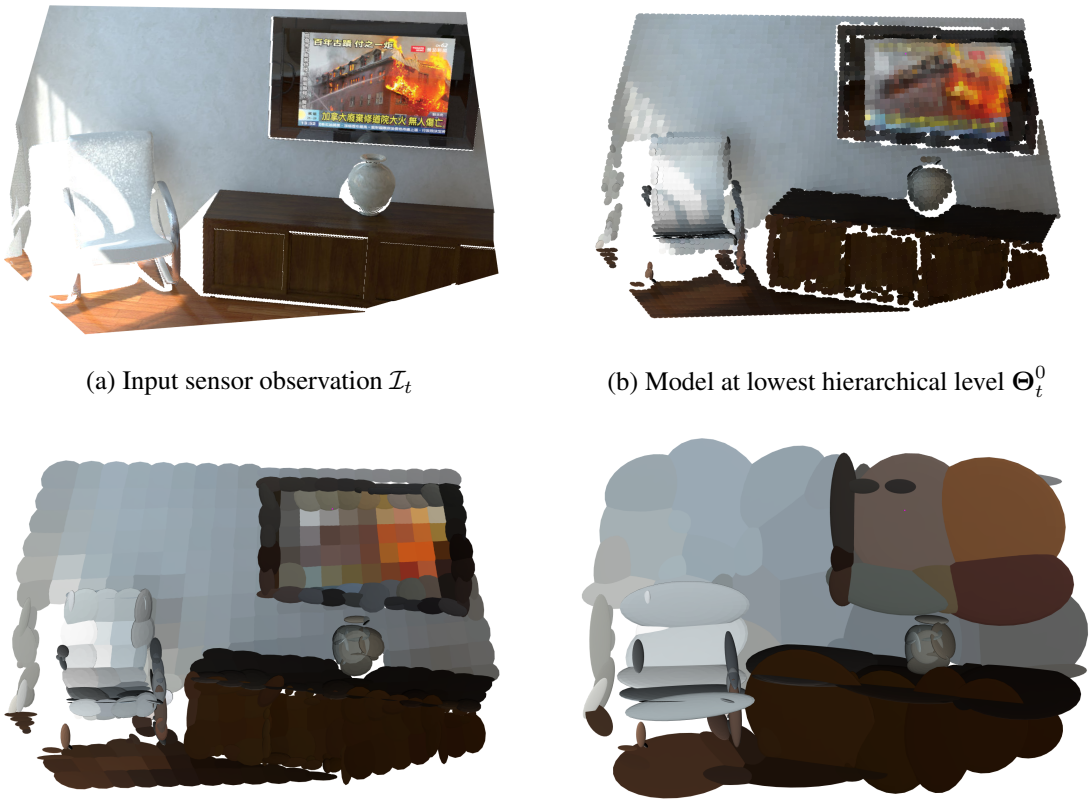
The model reconstruction strategy described in Sec. 4.1.1 is inherently independent for each image patch and therefore, easily extendible to a hierarchical model reconstruction strategy. Similar to the level 0 pixel space region-growing, we employ a region-growing approach modified to use Gaussian distributions, ${}^w\boldsymbol{\theta}_t$ as the primitives instead of points ${}^w\mathbf{x}$. A 3-level hierarchy is defined in image space by dividing the image into larger image patches recursively. We divide the image into image patches of size $b_1 \times b_1$ at hierarchical level 1 and $b_2 \times b_2$ at hierarchical level 2 such that $b_2 \geq b_1 \geq b_0$. Given the map fitted at hierarchical level 0, ${}^w\boldsymbol{\Theta}_t^0$, a Gaussian distribution based "Region Growing" is performed in a image patch of size $b_1 \times b_1$, using "Bhattacharyya Coefficient"[1] as a similarity measure with thresholds on the maximum thickness of a Gaussian distribution $\alpha_{\lambda,1}$ and the largest spread of a Gaussian distribution $\alpha_{\text{len},1}$.

Once all ${}^w\boldsymbol{\Theta}_t^1$ distributions are computed, the same process is repeated in a larger image patch of size $b_2 \times b_2$ with thresholds $\alpha_{\lambda,2}$ and $\alpha_{\text{len},2}$. Finally, every distribution ${}^w\boldsymbol{\theta}_k^2$ absorbs one or more distributions $\boldsymbol{\theta}_k^1$ which absorbs one or more $\boldsymbol{\theta}_t^0$. Due to the distributive nature of our image patch splitting, each distribution in hierarchical level $l-1$ can only have one parent distribution in hierarchical level $l$ that it is encompassed by. An illustrative example of a hierarchical map learned on a single depth measurement is shown in Fig. 4.4.

As the model ${}^w\boldsymbol{\Theta}_t^0$ is incrementally updated, these updates are propagated up the hierarchical tree structure at each time step and therefore the entire hierarchical map is updated with the most recent information. Novel components observed in the live sensor frame that get added to ${}^w\boldsymbol{\Theta}_t^0$ also get added to ${}^w\boldsymbol{\Theta}_t^1$ and ${}^w\boldsymbol{\Theta}_t^2$. A Gaussian distribution based "Region Growing" is performed at level 1 in an image patch of size $b_1 \times b_1$ to incorporate these new distributions in to ${}^w\boldsymbol{\Theta}_t^1$. Similarly,

[1] https://en.wikipedia.org/wiki/Bhattacharyya_distance

novel distributions in ${}^{w}\mathbf{\Theta}_t^1$ get incorporated into ${}^{w}\mathbf{\Theta}_t^2$ by performing a Gaussian distribution based "Region Growing" on distributions in ${}^{w}\mathbf{\Theta}_t^1$ in an image patch of size $b_2 \times b_2$.



(a) Input sensor observation $\mathcal{I}_t$

(b) Model at lowest hierarchical level $\mathbf{\Theta}_t^0$

(c) Model at hierarchical level 1 $\mathbf{\Theta}_t^1$

(d) Model at hierarchical level 2 $\mathbf{\Theta}_t^2$

*Figure 4.4: Illustrative example of the proposed hierarchical mapping strategy. Given an image $\mathcal{I}_t$ at time $t$ our hierarchical approach fits a highest resolution map ${}^{w}\mathbf{\Theta}_t^0$ according to Sec. 4.1.1 on image patches of size $b \times b$. Distributions in this model that lie on a large image patch of size $b_1 \times b_1$ are combined to create ${}^{w}\mathbf{\Theta}_t^1$ and similarly distributions in ${}^{w}\mathbf{\Theta}_t^1$ that lie on an even larger image patch of size $b_2 \times b_2$ are further combined to create ${}^{w}\mathbf{\Theta}_t^2$*

### 4.1.8 Sensor Uncertainty Model

The uncertainty associated with a sensor measurement ${}^{w}\mathbf{x}_{v,u}$ observed at a pixel location $(v, u)$ is assumed to be normally distributed with zero mean and ${}^{w}\mathbf{\Sigma}_{v,u}^{\text{unc}}$ covariance in Sec. 4.1.2. Given a noisy point $\mathbf{x}$ with normally distributed noise characteristics $\boldsymbol{\theta}^{\text{unc}} := \{{}^{w}\mathbf{b}, {}^{w}\mathbf{\Sigma}^{\text{unc}}\}$ the true world

point $\bar{\mathbf{x}}$ can be written as:

$$^{w}\bar{\mathbf{x}} = {}^{w}\mathbf{x} + \mathcal{N}\left({}^{w}\mathbf{b}, {}^{w}\mathbf{\Sigma}^{\text{unc}}\right) \tag{4.21}$$

$$^{w}\bar{\mathbf{x}} = \left({}^{w}\mathbf{x} + {}^{w}\mathbf{b}\right) + \mathcal{N}\left(\mathbf{0}, {}^{w}\mathbf{\Sigma}^{\text{unc}}\right) \tag{4.22}$$

Therefore, we can interpret a Gaussian uncertainty with a non-zero mean, as a zero mean Gaussian distribution with the same covariance and bias adjusted measurement (see Eq. 4.22).

A raw measurement $^{w}\mathbf{x}_{v,u}$ obtained from a depth sensor has three principal directions of uncertainty: one along the direction of the ray, $\sigma_z$ in $\mathbb{R}^3$ and two along the pixel location width and height in pixel space (pixel quantization error), $\sigma_p$, as shown in Fig. 4.5. As presented by Proença and Gao [42], the uncertainties in the pixel space can be propagated to $\mathbb{R}^3$ space and transformed into the camera coordinate frame:

$$^{w}\mathbf{\Sigma}^{\text{unc}}_{v,u} = \begin{bmatrix} f_x^{-1} & 0 & (u-c_x)\,f_x^{-1} \\ 0 & f_y^{-1} & (v-c_y)\,f_y^{-1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_p^2 & 0 & 0 \\ 0 & \sigma_p^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \begin{bmatrix} f_x^{-1} & 0 & (u-c_x)\,f_x^{-1} \\ 0 & f_y^{-1} & (v-c_y)\,f_y^{-1} \\ 0 & 0 & 1 \end{bmatrix}^T \tag{4.23}$$

where, $f_x, f_y, c_x, c_y$ are the camera intrinsic parameters.

The value of $\sigma_z$ is computed for each pixel in the sensor measurement from an empirically fitted model presented in [36] for a Kinect RGBD sensor, where $\sigma_z$ is shown to predominantly vary with the $z$ coordinate of a point $\mathbf{x}_{v,u}$,

$$\sigma_z = 0.0012 + 0.0019\left(x_z - 0.4\right)^2 + \frac{0.0001}{\sqrt{x_z}} \tag{4.24}$$

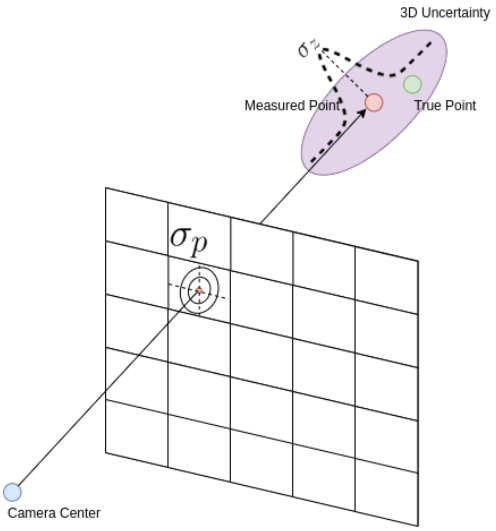



*Figure 4.5: Illustrative example of the Gaussian uncertainty associated with a raw sensor measurement. The measurement obtained at a pixel location may not lie at the center of the pixel. $\sigma_p$ denotes the standard deviation associated with the measurement being obtained from the center of the pixel along the $u$ and $v$ directions. $\sigma_z$ denotes the standard deviation associated with the depth measurement along the direction of the ray passing through the center of the camera and the observed 3D point. The 3D ellipsoid (shown in purple) represents the 3D Gaussian uncertainty obtained by propagating the standard deviations through projective constraints.*

### 4.1.9 Active-Inactive Mapping

As the size of the observed map increases, the number of Gaussian distributions in the global map ${}^{w}\mathbf{\Theta}_t^0$ increases proportionally. This increases the cost of projecting the map in to the image frame and computing point-to-model correspondences (see Sec. 4.1.3). RGBD sensors like the Kinect have a limited FOV. When a stream of sensor measurements is obtained, only a part of the observed map ${}^{w}\mathbf{\Theta}_t^0$ is currently in the sensor FOV. The raw sensor measurements in the live depth frame can only affect the part of the map inside its FOV. Exploiting these sensor FOV constraints, we can reduce the size of the map used for correspondence computation to only the subsection that is within the FOV of the sensor. This enables us to trim the number of distributions that will be affected by the current sensor measurement to only the distributions that lie within the FOV of the sensor.

Every Gaussian distribution in ${}^{w}\mathbf{\Theta}_t^0$ that has a finite projection in the current image frame must

therefore be used for correspondence computation. However, the computational complexity of this operation also scales with the number of Gaussian distributions in ${}^{w}\boldsymbol{\Theta}_t^0$. Instead, we exploit the hierarchical structure of the map representation to efficiently down-sample the map size by checking the projection of distributions at hierarchical level 2, ${}^{w}\boldsymbol{\Theta}_t^2$. ${}^{w}\boldsymbol{\Theta}_t^2$ has a very small number of distributions as compared to ${}^{w}\boldsymbol{\Theta}_t^0$ (experimentally shown in Fig. 5.8) and therefore, can be used to quickly check if the individual distributions ${}^{w}\boldsymbol{\theta}_t^2$ lie within the FOV of the sensor. All ${}^{w}\boldsymbol{\Theta}_t^0$ distributions whose parent distribution ${}^{w}\boldsymbol{\Theta}_t^2$, lies outside the current FOV of the sensor, are labeled inactive and the rest of the map is active.

This perspective based component sub-selection enables us to compute the point-to-distribution correspondences efficiently and create a globally accurate map representation.

## 4.2 Implementation

For the remainder of this thesis we use $b_0 = 8, b_1 = 32, b_2 = 160$. This provides us a very high fidelity map representation at level $0$, a smoother map representation at level $1$ and a low fidelity but highly succinct map representation at level $2$.

## 4.3 Evaluation

In this section, we demonstrate the reconstruction accuracy of the proposed map representation and also highlight the compression capabilities. We evaluate the accuracy and performance of our proposed mapping strategy on multiple datasets quantitatively and qualitatively. First, we demonstrate the correctness of our incremental mapping strategy on noisy input data. Second, we compare the metric accuracy of our mapping approach to various state-of-the-art mapping algorithms and demonstrate the superior reconstruction performance with lower memory requirements. Third, we demonstrate the mapping strategy's compression capabilities while incurring marginal loss of fidelity. Finally, we compare the qualitative performance of our proposed approach on publicly

available real world datasets in terms of quality and error of reconstruction. In all the following experiments, $b$, the patch size is set to $8$.

We only compare the memory footprint utilized by NDTMap and OctoMap for storing occupied cells to maintain a fair comparison of compression capabilities.

We use the following datasets for our evaluations:

1. "Living Room": ICL-NUIM Living room Dataset [8, 20]
2. "Lounge": Lounge Dataset, "Copyroom": Copyroom Dataset, "Stonewall": Stonewall Dataset [62]

### 4.3.1 Comparison Metrics

The quantitative evaluation of our proposed 3D mapping approach is performed using the following metrics:

1. Reconstruction Error (m) : Gaussian distributions used as structure primitives in this work are generative models. We sample 3D points from each Gaussian distribution in ${}^{w}\mathbf{\Theta}^{l}$ at hierarchical level $l$, within $3\sigma$ confidence bounds to reconstruct a global 3D point cloud of the environment. The reconstruction error is then computed as the mean distance of each 3D point in the sampled point cloud to its closest surface in the ground truth mesh of the environment. Larger error suggests that the sampled points are farther away from the ground truth map and smaller error suggests high similarity between the ground truth map and the reconstructed point cloud.
2. Precision of Reconstruction : Precision is defined as the fraction of sampled points whose distance to the ground truth mesh is less than $\alpha_{l,\lambda}$ for our proposed approach. For NDTMap and OctoMap precision is defined as the fraction of points whose distance to the ground truth mesh is less than the resolution of the map.
3. Recall of Reconstruction : Recall of the reconstruction is computed by sampling 3D points uniformly over the ground truth mesh and querying whether the points are observed by

the 3D reconstructed map. For our proposed approach and NDTMap, a point is defined as observed if it lies inside the $3\sigma$ bounds of any Gaussian distribution in the map. For OctoMap, a point is defined as observed if it lies inside an observed voxel of the map.

4. Memory Consumption (MB) : The memory consumption of a representation is defined in MegaBytes (MB) as the required storage space for each map representation at the end of a dataset.

### 4.3.2 Accuracy of Representation

In this section, we demonstrate the accuracy of our incremental mapping strategy on perfect and noisy sensor data on "Living Room" dataset. We compare the performance of Gaussian distributions at hierarchical level 0 learned on individual sensor scans to Gaussian distributions learned incrementally on perfect sensor data and incrementally on noisy sensor data. Table 4.1 shows that the proposed incremental strategy reconstructs the world with low error and high precision and recall of input information obtained from every scan. Further, actively incorporating the noise model of the sensor while mapping enables the pipeline to fit a more compressed representation thus reducing the memory requirement of the map representation while increasing the precision of reconstruction over time.

### 4.3.3 Surface Reconstruction Accuracy

We evaluate the surface reconstruction accuracy of our approach and compare it to state-of-the-art algorithms, OctoMap [24] and NDTMap [5], on the entire "Living Room" dataset with added Gaussian noise to each sensor measurement according to a sensor model described in Sec. 4.1.8. This dataset provides ground truth poses for a camera following a trajectory in a simulated environment. This environment mesh is used to compute the reconstruction accuracy of the fitted model. We also compare the memory footprint of these representations and demonstrate the superior compression and representation capability of the proposed method. Figure 4.7 shows the

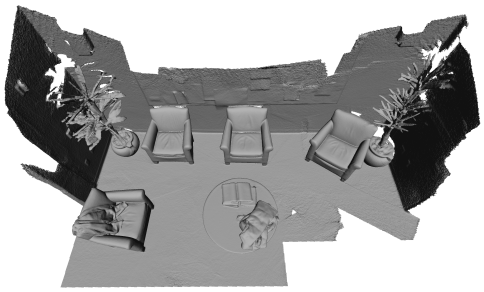
| Map Type | Error (m) | Precision | Recall | Memory (MB) |
|---|---|---|---|---|
| Perfect input data | | | | |
| Individual | 0.0019 | 0.890 | **0.997** | 0.913 |
| Incremental | **0.0006** | **0.987** | 0.996 | **0.0372** |
| Noisy input data | | | | |
| Incremental | 0.0031 | 0.790 | 0.985 | 0.2604 |
| Noise Compensated | **0.0017** | **0.939** | **0.992** | **0.041** |

*Table 4.1: Quantitative performance comparison of ${}^{w}\boldsymbol{\Theta}_t^0$ fit to each individual scan on "Living Room" dataset with a model fit incrementally, with and without explicitly incorporating the sensor noise in incremental updates. By performing incremental map updates, we are able to exploit the overlapping structure information observed in multiple sensor scans and thus achieve an order of magnitude better compression performance. Further, by explicitly incorporating the sensor uncertainty into the map reconstruction pipeline, we are able to capture the underlying distribution of structure points with higher accuracy and also reduce the memory footprint of the map.*

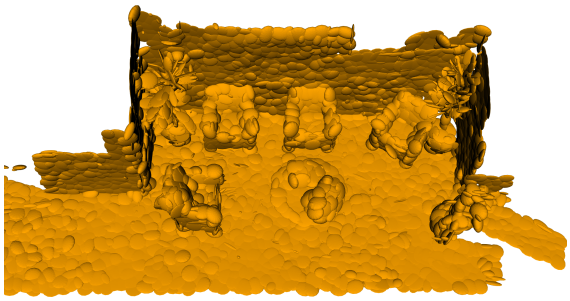
qualitative reconstruction of "Living Room" dataset using noisy input data as compared to the true mesh. Model fitted using our proposed approach represents the environment with high accuracy by representing input points as uncertain distributions and probabilistically updating the map representation. Table 4.2 shows that given current sensor pose estimates our proposed approach can represent the scene with higher precision, reconstruction accuracy and lower memory footprint than OctoMap and NDTMap.

| Approach | Error (m) | Precision | Recall | Memory (MB) |
|---|---|---|---|---|
| **Proposed** | **0.0019** | **0.890** | **0.985** | **0.128** |
| NDTMap | 0.0042 | 0.691 | 0.653 | 0.731 |
| OctoMap | 0.0236 | 0.251 | 0.636 | 0.5901 |

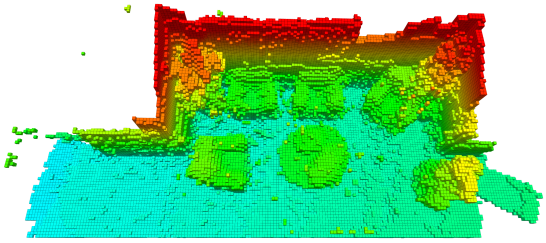
*Table 4.2: Quantitative comparison of state-of-the-art mapping approaches with the proposed approach on noisy "Living Room" dataset. OctoMap and NDTMap are being fit at 5 cm resolution. For our proposed approach we use maximum thickness parameter $\alpha_{0,\lambda} = 5$ cm. The proposed approach can explicitly incorporate the uncertainty of the sensor measurements into the model. Further, the GMM map is data driven and does not make assumptions about the distribution of data in fixed grids like OctoMap and NDTMap. Therefore, our proposed approach can achieve higher accuracy, precision, recall while being more memory efficient that other compression mapping techniques.*



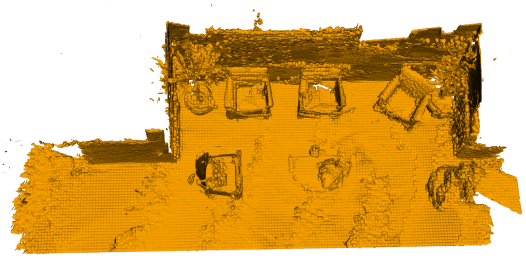
(a) Ground truth mesh of "Lounge" dataset



(b) Reconstruction ${}^{w}\boldsymbol{\Theta}_{t}^{0}$ from proposed approach



(c) Reconstructed OctoMap for "Lounge" dataset

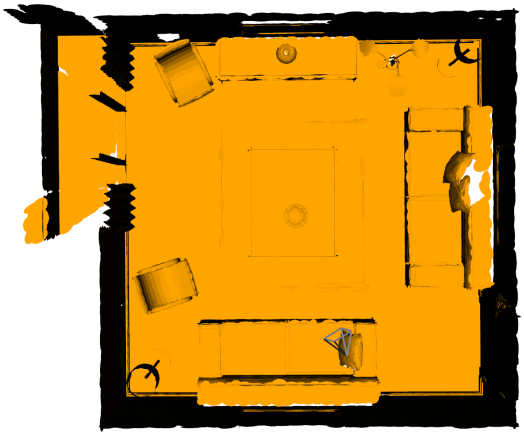


(d) Reconstruced NDTMap for "Lounge" dataset

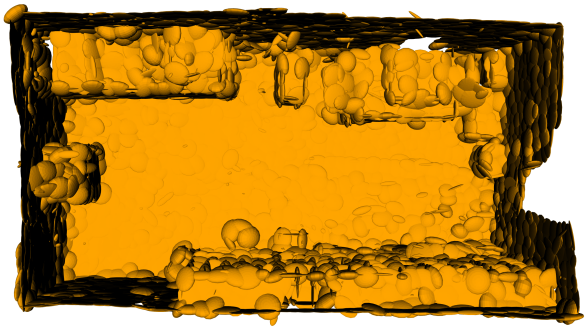
*Figure 4.6: Qualitative comparison on "Lounge" dataset of Left to right: Mesh estimated from [62]; proposed reconstruction framework; OctoMap, NDTMap. Unlike, OctoMap and NDTMap our map representation does not make assumptions about the distribution of the structure points and fits volumetric planar primitives to the data. Further, due to the explicit incorporation of sensor uncertainty in the representation, the proposed representation looks less cluttered and more structured than NDTMap and OctoMap while achieving orders of magnitude higher compression than either of those representations.*

### 4.3.4 Map Compression

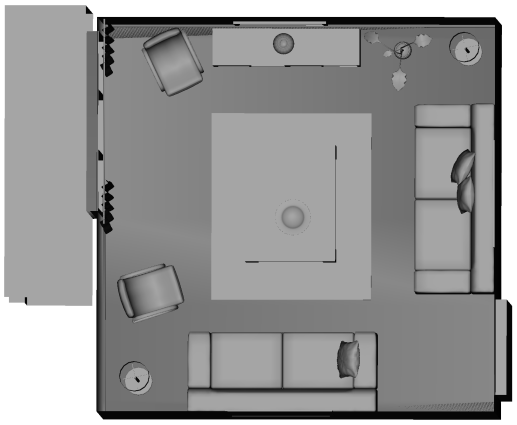
The proposed mapping strategy can be used to solve various problems in robotics like state estimation [54], trajectory planning and collision avoidance [12]. However, for real-time performance, these applications require a map representation with low memory footprint and high computational efficiency. In this section, we demonstrate the capability of the proposed mapping strategy to obtain an order of magnitude better compression of input data from "Living Room" dataset with minor loss of representation fidelity. This ensures complete map coverage and enables the use of the proposed mapping approach to enable other robotic applications. Table 4.3 demonstrates that by reducing the required surface reconstruction accuracy from 1 mm to 10 cm, the memory



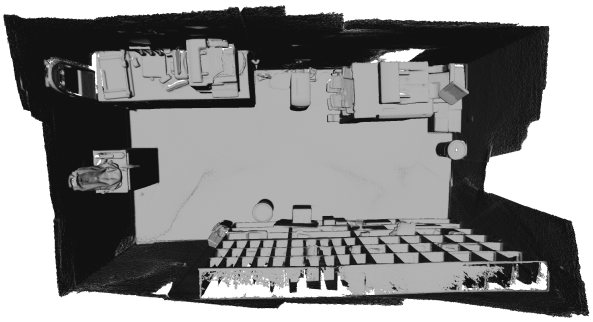
(a) “Living Room” reconstruction



(b) “Copyroom” reconstruction



(c) “Living Room” ground truth mesh



(d) “Copyroom” ground truth mesh

*Figure 4.7: Qualitative comparison of the top down view of reconstruction obtained from the proposed framework on “Living Room” dataset (a) and “Copyroom” dataset(b), ground truth map of “Living Room” dataset (c) and the mesh constructed from [62] for “Copyroom” dataset(d).*

requirement of the map is reduced by a factor of approximately 20 and the mean reconstruction error is still less than 1 cm.

### 4.3.5 Hyper Parameter Selection

The trade-off between accuracy and map compression can be tuned by proper selection of the hyper parameters $b_i$, $\alpha_{i,\text{len}}$ and $\alpha_{i,\lambda}$. Varying the patch size at level 0, $b_0$, changes the size of image patches used for model fitting. Larger the value of $b_0$ lesser the number of Gaussian distributions fit to the map. However, that also reduces the recall of the map representation as only one Gaussian

| $\alpha_{0,\lambda}$ (m) | $\alpha_{0,\text{len}}$ (m) | Error (m) | Precision | Recall | Memory (MB) |
|---|---|---|---|---|---|
| 0.01 | 0.05 | 0.0008 | 0.969 | **0.989** | 0.314 |
| 0.001 | 0.05 | **0.0006** | **0.987** | **0.989** | 0.349 |
| 0.1 | 0.1 | 0.0029 | 0.877 | 0.985 | 0.075 |
| 0.01 | 0.1 | 0.0016 | 0.934 | 0.985 | 0.195 |
| 0.001 | 0.1 | 0.0009 | 0.966 | **0.989** | 0.143 |
| 0.1 | 0.2 | 0.0098 | 0.701 | 0.970 | **0.020** |
| 0.01 | 0.2 | 0.0034 | 0.883 | 0.982 | 0.0308 |
| 0.001 | 0.2 | 0.0033 | 0.930 | **0.989** | 0.084 |

*Table 4.3: Quantitative comparison of various performance metrics at different map resolutions on "Living Room" dataset. At high resolutions, we achieve extremely accurate metric performance, however, have a large memory footprint. Even after increasing the thickness and length thresholds, our proposed approach is able to represent the structure information in the scene at high accuracy with error less than* $1$ *cm and gain substantial decrease in memory footprint. Larger thresholds enable* ${}^{w}\boldsymbol{\Theta}_{t}^{0}$ *to represent large planar regions in structured environments with less model complexity.*

distribution is fit to each $b_0 \times b_0$ image patch. The parameters $\alpha_{0,\lambda}$ and $\alpha_{0,\text{len}}$ dictate the maximum spread of a Gaussian distribution at level $0$. Lower $\alpha_{0,\lambda}$ enables distributions to represent extremely planar regions with high accuracy while a higher value of $\alpha_{0,\lambda}$ represents volumetric regions in the scene. Similarly, a smaller value of $\alpha_{0,\text{len}}$ ensures that the law of large number is not violated by representing a large amount of data with a single Gaussian distribution. However, a large value of $\alpha_{\text{len}}$ still enables the algorithm to represent the map with high accuracy and with a low memory footprint. Table 4.4 highlights this trade-off on the highest resolution model ${}^{w}\boldsymbol{\Theta}_{T}^{0}$ on the "Living Room" dataset.

### 4.3.6 Real World Datasets

The objective of using these datasets is to demonstrate the performance of our pipeline on real-world datasets where the provided ground truth poses are noisy and a correct sensor model is not available. "Lounge", "Copyroom" and "Stonewall" datasets are captured using a hand-held sensor and sensor poses are estimated as described by Zhou and Koltun [62]. These datasets provide an estimate of the map reconstructed using a dense mapping framework. We compare the performance

| $\alpha_{0,\lambda}$ (m) | $\alpha_{0,\text{len}}$ (m) | $b_0$ | Error (m) | Precision | Recall | Number of ${}^{w}\boldsymbol{\theta}^{0}$ |
|---|---|---|---|---|---|---|
| 0.01 | 0.05 | 8 | 0.0008 | 0.969 | 0.989 | 7858 |
| 0.001 | 0.05 | 8 | **0.0006** | **0.987** | **0.989** | 8743 |
| 0.1 | 0.1 | 8 | 0.0029 | 0.877 | 0.985 | 1877 |
| 0.01 | 0.1 | 8 | 0.0016 | 0.934 | 0.985 | 4873 |
| 0.001 | 0.1 | 8 | 0.0009 | 0.966 | **0.989** | 3576 |
| 0.1 | 0.2 | 8 | 0.0098 | 0.701 | 0.970 | **518** |
| 0.01 | 0.2 | 8 | 0.0034 | 0.883 | 0.982 | 772 |
| 0.001 | 0.2 | 8 | 0.0033 | 0.930 | **0.989** | 2116 |
| 0.01 | 0.05 | 16 | 0.0014 | 0.932 | 0.987 | 7397 |
| 0.001 | 0.05 | 16 | 0.0009 | 0.969 | 0.988 | 9079 |
| 0.1 | 0.1 | 16 | 0.0027 | 0.885 | 0.985 | 1940 |
| 0.01 | 0.1 | 16 | 0.0019 | 0.926 | 0.985 | 2200 |
| 0.001 | 0.1 | 16 | 0.0011 | 0.959 | **0.989** | 4118 |
| 0.1 | 0.2 | 16 | 0.007 | 0.783 | 0.983 | 565 |
| 0.01 | 0.2 | 16 | 0.003 | 0.885 | 0.982 | 805 |
| 0.001 | 0.2 | 16 | 0.0024 | 0.934 | **0.989** | 2673 |

*Table 4.4: Comparison of reconstruction error, precision, recall and memory consumption of the proposed mapping approach at different resolutions by varying the mapping hyper-parameters.*

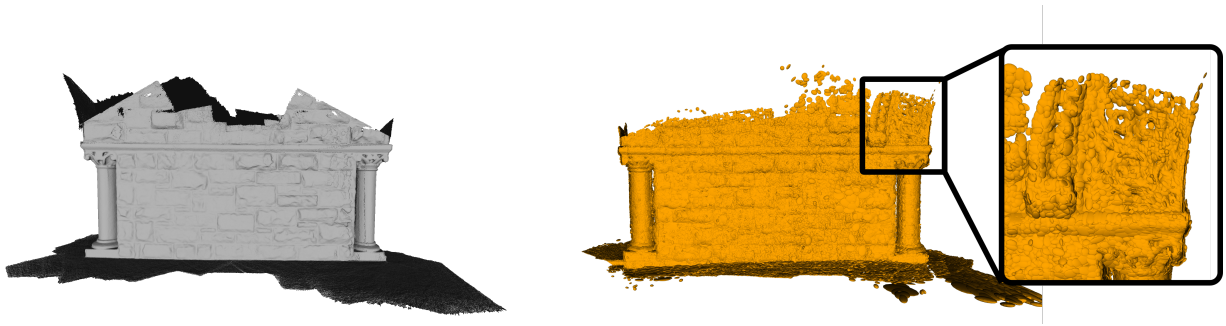


*Figure 4.8: Qualitative comparison of the model reconstructed by our approach at high information fidelity on "Stonewall" dataset: Left: Mesh provided by [62]; Right: proposed reconstruction with $\alpha_{0,\lambda} = 0.1$ cm. The zoomed in view shows that the high fidelity model is able to retain minute texture details in the scene and create a high quality scene reconstruction using succinct Gaussian distributions as surface primitives.*

of our mapping strategy with state-of-the-art mapping frameworks using this model. In order to compare the performance of OctoMap and NDTMap with the proposed hierarchical approach, we vary the voxel grid size for these map representations and compare the memory consumption and reconstruction error of the representations at various configurations. However, since our approach does not use 3D voxels to change the resolution, we compare the performance of our approach at all three hierarchical levels with OctoMap and NDTMap at multiple resolutions. Table 4.5 and Table 4.6 demonstrate that for "Lounge" and "Copyroom" datasets, the proposed hierarchical map representation outperforms NDTMap and OctoMap in both compactness of the representation and the accuracy of reconstruction at all three hierarchical levels. Even with noisy pose estimates and unknown sensor noise model the proposed algorithm creates a qualitatively accurate representation of the environment and is orders of magnitude more memory efficient than NDTMaps and OctoMaps on these datasets. NDTMap fits volumetric distributions to fixed sized voxels, while OctoMap represents the world as voxels. Our approach on the other hand, can more accurately capture the spread of the structure data along planar regions and shown in Fig. 4.6. Therefore, we can achieve very high compression values while retaining high reconstruction accuracy. The proposed map refinement step eliminates the spurious distributions learnt on noisy sensor observations. Therefore, our approach as a low reconstruction error and a small standard deviation,

whereas NDTMap has a large amount of noise incorporated in the final map representation as shown in Fig. 4.9 and Fig. 4.10. These figures show that our approach achieves the best trade-off between memory consumption and reconstruction accuracy and is capable of reconstructing the world with high accuracy. We also qualitatively compare a high fidelity model reconstruction of "Stonewall" dataset at $\alpha_{0,\lambda} = 0.1$ cm in Fig. 4.8. The higher fidelity model at hierarchical level $0$ retains minute details in the scene but requires a magnitude more memory than the lower fidelity model at hierarchical level $2$. The lower fidelity model loses the detailed texture, but retains all the geometric details of the scene.

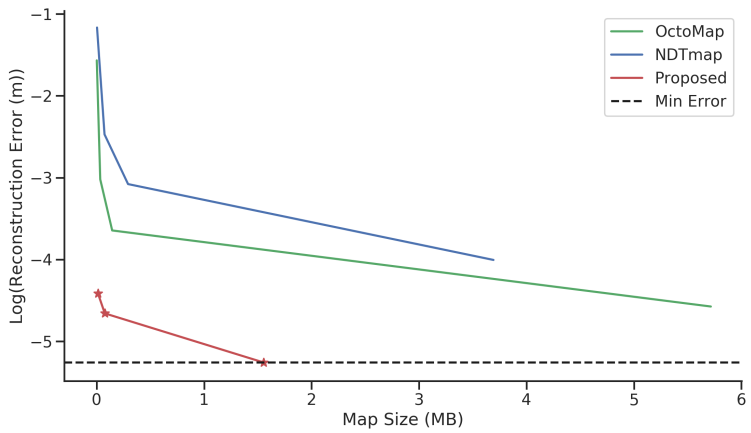



*Figure 4.9: Quantitative comparison of the memory vs accuracy trade-off of state-of-the-art mapping approaches with the proposed approach on "Copyroom" dataset.*

### 4.3.7 Runtime Analysis

The proposed map initialization and update framework proposed in this chapter operates at an average rate of $27$ Hz and the active-inactive map segmentation component operates at $\approx 60$ Hz. A detailed runtime analysis of the mapping algorithm is presented in Chapter 5 along with runtime analysis of the localization component proposed in this thesis.

| Approach | Resolution | Error (m) | Memory (MB) |
|---|---|---|---|
| NDTMap | 0.01 | $0.0182 \pm 0.0835$ | 3.69 |
| | 0.05 | $0.0460 \pm 0.2056$ | 0.29 |
| | 0.1 | $0.0844 \pm 0.3001$ | 0.07 |
| | 0.5 | $0.3116 \pm 0.5760$ | 0.01 |
| OctoMap | 0.01 | $0.0103 \pm 0.0197$ | 5.71 |
| | 0.05 | $0.0261 \pm 0.0551$ | 1.45 |
| | 0.1 | $0.0487 \pm 0.0897$ | 0.03 |
| | 0.5 | $0.2086 \pm 0.1904$ | **0.002** |
| Proposed | Level 0 | $\mathbf{0.0052 \pm 0.0086}$ | 1.55 |
| | Level 1 | $\mathbf{0.0094 \pm 0.0140}$ | 0.07 |
| | Level 2 | $\mathbf{0.0121 \pm 0.0193}$ | 0.01 |

*Table 4.5: Quantitative comparison of state-of-the-art mapping approaches with proposed approach on "Copyroom" dataset using reconstruction accuracy and memory footprint metrics. Since our method does not set a specific constraint over the resolution such as the voxel size, we compare the trade-off between accuracy and memory consumption in Fig. 4.9. Our map representation has lower reconstruction error than NDTMap and OctoMap at different resolutions. The reconstruction error of the lowest fidelity map of the proposed approach is comparable to OctoMap and outperforms NDTMap at* $0.01$ *m resolution. NDTMap does not incorporate noisy measurements correctly in the map and therefore has a large variance in reconstruction error.*

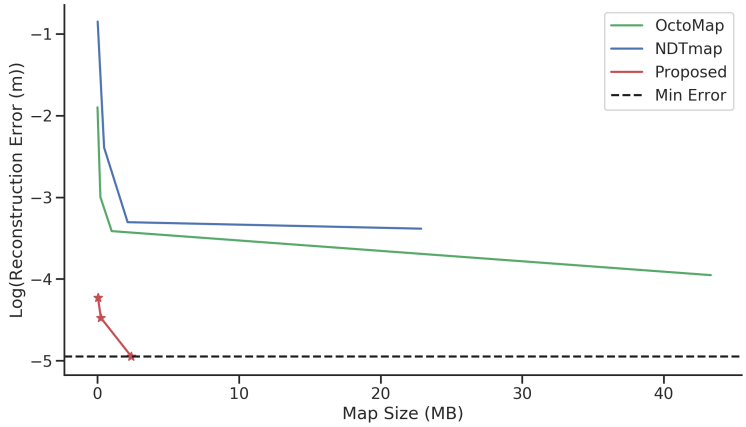



*Figure 4.10: Quantitative comparison of memory vs accuracy trade-off of state-of-the-art mapping approaches with the proposed approach on "Lounge" dataset.*

| Approach | Resolution | Error (m) | Memory (MB) |
|---|---|---|---|
| NDTMap | 0.01 | $0.0339 \pm 0.0801$ | 22.8664 |
| | 0.05 | $0.0367 \pm 0.2067$ | 2.1333 |
| | 0.1 | $0.0914 \pm 0.3709$ | 0.4732 |
| | 0.5 | $0.4273 \pm 0.9783$ | 0.0188 |
| OctoMap | 0.01 | $0.0192 \pm 0.0185$ | 43.3226 |
| | 0.05 | $0.0329 \pm 0.0283$ | 1.0096 |
| | 0.1 | $0.0499 \pm 0.0546$ | 0.2104 |
| | 0.5 | $0.1496 \pm 0.2059$ | **0.0066** |
| Proposed | Level 0 | $\mathbf{0.0071 \pm 0.0100}$ | 2.3918 |
| | Level 1 | $\mathbf{0.0114 \pm 0.0139}$ | 0.2433 |
| | Level 2 | $\mathbf{0.0146 \pm 0.0199}$ | 0.0379 |

*Table 4.6: Quantitative comparison of memory vs accuraacy trade-off of state-of-the-art mapping approaches with the proposed approach on "Lounge" dataset. Since our method does not set a specific constraint over the resolution such as the voxel size, we compare the trade-off between accuracy and memory consumption in Fig. 4.10. Our map representation has lower reconstruction error than NDTMap and OctoMap at different resolutions. The reconstruction error of the lowest fidelity map of the proposed approach is still lower than the reconstruction error of NDTMap and OctoMap constructed at* $0.01$ *m resolution and consumes orders of magnitude less memory.*

# Chapter 5

# Iterative Closest Distribution

In the previous chapter, the pose of the sensor is assumed to be known perfectly at each time step to reconstruct a dense map of the environment as shown in Fig. 4.1. However, in real-world scenarios, this information is not always available. Other sources of odometry may exist, but errors in the extrinsic calibration of various sensors get propagated in the map representation, leading to an inaccurate reconstruction. In this chapter, this limitation is addressed by proposing a frame-to-model localization technique that computes a locally optimum transformation between the live sensor frame and a global map estimate using only depth sensor information. The localization pipeline is more robust to noisy sensor information than state-of-the-art approaches due to the explicit incorporation of uncertainty associated with each sensor measurement into the localization cost function. This localization framework is then integrated with the mapping framework to propose a SLAM framework that is able to generate a consistent map of the scene observed from a COTS depth sensor. We demonstrate quantitatively and quantitatively the improvement in performance achieved by using the proposed localization framework with the map representation introduced in Chapter 4 over state-of-the-art SLAM techniques.

## 5.1 Introduction

This work bridges the gap between existing state-of-the-art dense depth tracking and mapping algorithms for depth cameras and the benefits of using Gaussian distributions as a succinct representation.

Reliable model based SLAM using depth images from COTS depth sensors necessitates accounting for their significant noise characteristics Nguyen et al. [36]. Conventional state-of-the-art algorithms described in Sec. 2.2, pre-process input data, smoothen out high frequency information and further incorporate approximations to sensor depth uncertainty while building the map model. Instead, we propose that this noise should be completely accounted for by the representation, with localisation framed as a problem of maximising the likelihood of raw sensor data having been sampled from it. The chosen map representation (Gaussian distributions) provides a smooth and continuous data likelihood function (see Sec. 3.1) with well defined gradients. Further, sensor noise characteristics can be well modeled by a Gaussian distribution with a finite bias and variance which can be empirically modeled as demonstrated by Nguyen et al. [36]. This representation thus enables us to formulate a robust likelihood based localization framework that can explicitly incorporate the uncertainty of a sensor measurement into the map representation. An illustrative example of a 3D Gaussian distribution map constructed using pose estimates computed with respect to this global map is shown in Fig. 5.1.

## 5.2 Localization Using Gaussian Distributions

This section describes the proposed frame-to-model localization strategy that is similar in spirit to Iterative Closest Point (ICP) (see Sec. 2.2), however, our approach minimizes the distance between uncertain point cloud measurements in the raw depth observations and their corresponding Gaussian distributions and computes a more robust and smooth cost function by explicitly incorporating

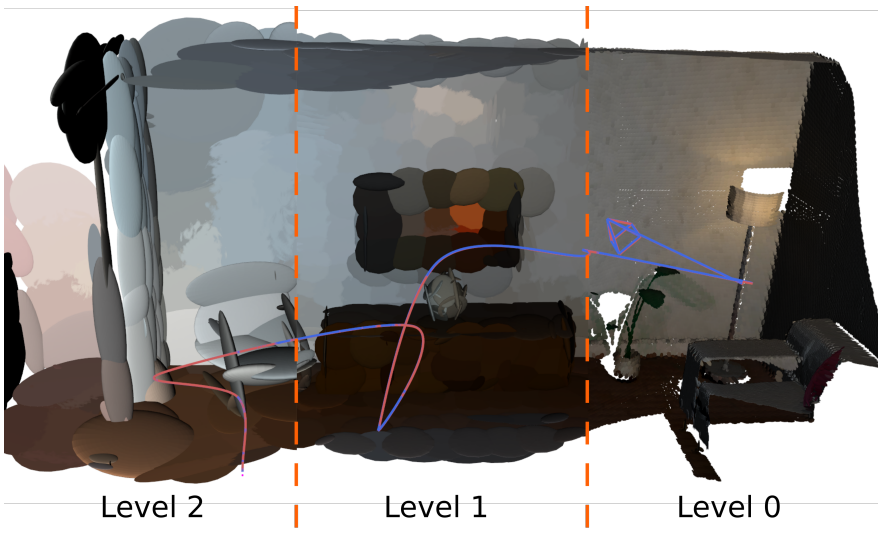



*Figure 5.1: Our approach continuously performs data-driven Gaussian distributions growth with spatial regularity imposed by the hierarchical structure. Shown here are map components in projected colour at the three hierarchical fidelities we maintain while building the map. Note how the higher level components intuitively fit the gross structure of the environment. The tracked trajectory in red along with the ground truth in blue are also superimposed. Segment from the "Living Room" dataset.*

the measurement uncertainty in the cost function formulation. Given an estimate of the global map ${}^{w}\boldsymbol{\Theta}^{l}$ at hierarchical level $l$, in a coordinate from $w$ and a point cloud ${}^{c}\mathbf{X}$ in the live sensor coordinate frame $c$, the objective of this section is to compute the transformation parameters ${}^{w}\mathbf{T}_{c} \in \mathbb{SE}(3)$ that can transform ${}^{c}\mathbf{X}$ to coordinate frame $w$, ${}^{w}\mathbf{X}$. We first describe the proposed cost function and then the cost minimization framework.

### 5.2.1 Point To Distribution Distance

The log-likelihood of a point ${}^{a}\mathbf{x}_{i}$ given a Gaussian distribution ${}^{a}\boldsymbol{\theta}$ defined in the same coordinate frame $a$ is given in Eq. 3.9. For a point cloud ${}^{a}\mathbf{X} := \{{}^{a}\mathbf{x}_{0}, {}^{a}\mathbf{x}_{1}, {}^{a}\mathbf{x}_{2}, \ldots, {}^{a}\mathbf{x}_{i}, \ldots, {}^{a}\mathbf{x}_{N}\}$ the log-likelihood is the sum of the log-likelihood of individual points. However, for a point cloud ${}^{c}\mathbf{X}$ in a coordinate frame $c$, a set of Gaussian distributions ${}^{w}\boldsymbol{\Theta}$ in a different coordinate frame $w$, and the coordinate transformation ${}^{c}\mathbf{T}_{w}$, the log-likelihood computation can be extended by transforming the Gaussian distributions in the coordinate frame $c$ and then compute the log-likelihood of ${}^{c}\mathbf{X}$. We define a function $\mathbb{C}(i) : \mathbb{R} \rightarrow \mathbb{R}$ as a correspondence function that provides the index of the

Gaussian distribution ${}^{c}\boldsymbol{\theta}$ that a 3D point ${}^{c}\mathbf{x}_i$ is most likely to be sampled from.

$$
\begin{aligned}
l\left({}^{c}\mathbf{X} \mid {}^{c}\mathbf{T}_w, {}^{w}\boldsymbol{\Theta}\right) &= \sum_{i}^{N} -\frac{1}{2}\log\left((2\pi)^d \mid {}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T \mid\right) \\
&\quad - \frac{1}{2}({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\mu}_{\mathbb{C}(i)} + {}^{c}\mathbf{t}_w - {}^{c}\mathbf{X}_i)^T \left({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T\right)^{-1} ({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\mu}_{\mathbb{C}(i)} + {}^{c}\mathbf{t}_w - {}^{c}\mathbf{X}_i) \qquad (5.1) \\
&= \sum_{i}^{N} -\frac{1}{2}\left(\log\left((2\pi)^d \mid {}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T \mid\right) + \mathbf{d}_i^T \left({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T\right)^{-1} \mathbf{d}_i\right) , \qquad (5.2)
\end{aligned}
$$

where $\mathbf{d}_i = ({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\mu}_{\mathbb{C}(i)} + {}^{c}\mathbf{t}_w - {}^{c}\mathbf{X}_i)$.

Raw depth measurements obtained from a COTS depth sensor are noisy. This noise can be modeled as a zero mean Gaussian distribution, $\left\{\mathbf{0}, {}^{c}\bar{\boldsymbol{\Sigma}}_i\right\}$ in 3D centered around the observed 3D point measurement ${}^{c}\mathbf{x}_i$ as described in Sec. 4.1.8. Using linearity of addition of normal distributions (see [39][Eq.355]), the uncertainty of the point can be explicitly added to the cost function Eq. 5.2

$$
\begin{aligned}
& l\left({}^{c}\mathbf{X}, {}^{c}\boldsymbol{\Sigma}_i^{\text{unc}} \mid {}^{c}\mathbf{T}_w, {}^{w}\boldsymbol{\Theta}\right) \\
&= \sum_{i}^{N} -0.5\log\left((2\pi)^d \mid {}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T + {}^{c}\boldsymbol{\Sigma}_i^{\text{unc}} \mid\right) \qquad (5.3) \\
&\quad - 0.5\left(\mathbf{d}_i^T \left({}^{c}\mathbf{R}_w \, {}^{w}\boldsymbol{\Sigma}_{\mathbb{C}(i)} \, {}^{c}\mathbf{R}_w^T + {}^{c}\boldsymbol{\Sigma}_i^{\text{unc}}\right)^{-1} \mathbf{d}_i\right)
\end{aligned}
$$

Explicitly adding the uncertainty of the sensor measurements into the cost function as described in Eq. 5.3 provides us the log-likelihood of uncertain points ${}^{c}\mathbf{X}$ with known noise model, having been sampled from a set of distributions ${}^{w}\boldsymbol{\Theta}$. This cost function is robust to noisy sensor measurements and is conservative in terms of the probability distribution. The value of this cost function is maximized at the true ${}^{c}\mathbf{T}_w$ and decays smoothly in the local neighborhood of ${}^{c}\mathbf{T}_w$.

### 5.2.2 Pose Estimation

The aim of this section is to find the transformation parameters ${}^{c}\boldsymbol{\phi}^{*}_{c-1} \in \mathbf{se}(3)$ that maximize the log-likelihood, Eq. 5.3, of an uncertain point cloud ${}^{c}\mathbf{X}$ having been sampled from a active map transformed in the last sensor coordinate frame ${}^{c-1}\boldsymbol{\Theta}$.

$$ {}^{c}\boldsymbol{\phi}^{*}_{c-1} = \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{\operatorname{argmax}}\; l\left({}^{c}\mathbf{X}, {}^{c}\boldsymbol{\Sigma}^{\text{unc}}_{i} \mid {}^{c}\mathbf{T}_{c-1}({}^{c}\boldsymbol{\phi}_{c-1}), {}^{c-1}\boldsymbol{\Theta}\right) \tag{5.4} $$

The noise model described in Sec. 4.1.8, $\boldsymbol{\Sigma}^{\text{unc}}_{v,u}$, is dependent on the view-point of the sensor and is inversely proportional to the squared depth measurement. A 3D point ${}^{c}\mathbf{x}_i$ is in close vicinity of the 3D mean ${}^{c}\boldsymbol{\mu}_{\mathbb{C}(i)}$ of its corresponding distribution ${}^{c}\boldsymbol{\theta}_{\mathbb{C}(i)}$. Further, COTS depth sensors operate at high frame rates and therefore the relative transformation ${}^{c}\mathbf{T}_{c-1}$ is small. Therefore, we can approximately write,

$$ {}^{c}\boldsymbol{\Sigma}^{\text{unc}}_{i}({}^{c}\mathbf{x}_i) \approx {}^{c}\boldsymbol{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}({}^{c}\boldsymbol{\mu}_{\mathbb{C}(i)}) \approx {}^{c-1}\boldsymbol{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}({}^{c-1}\boldsymbol{\mu}_{\mathbb{C}(i)}) \tag{5.5} $$

Combining Eq. 5.4 and Eq. 5.5,

$$ \begin{aligned} {}^{c}\boldsymbol{\phi}^{*}_{c-1} &= \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{\operatorname{argmax}}\; l\left({}^{c}\mathbf{X}, {}^{c}\boldsymbol{\Sigma}^{\text{unc}}_{i} \mid {}^{c}\mathbf{T}_{c-1}({}^{c}\boldsymbol{\phi}_{c-1}), {}^{c-1}\boldsymbol{\Theta}\right) && (5.6) \\ &= \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{\operatorname{argmax}} \sum_{i}^{N} -0.5 \log\left((2\pi)^{d} \mid {}^{c}\mathbf{R}_{c-1}\, {}^{c-1}\left(\boldsymbol{\Sigma}_{\mathbb{C}(i)}{}^{c} + {}^{c-1}\boldsymbol{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}\right)\mathbf{R}^{T}_{c-1} \mid\right) \\ &\quad - 0.5\left(\mathbf{d}^{T}_{i}\left({}^{c}\mathbf{R}_{c-1}\left({}^{c-1}\boldsymbol{\Sigma}_{\mathbb{C}(i)} + {}^{c-1}\boldsymbol{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}\right){}^{c}\mathbf{R}_{c-1}\right)^{-1}\mathbf{d}_i\right) && (5.7) \end{aligned} $$

Since ${}^{c}\mathbf{R}_{c-1} \in \mathbf{SO}(3)$ is an orthogonal matrix (see 3.3), the first term in the summation in Eq. 5.7 is independent on the parameters ${}^{c}\boldsymbol{\phi}_{c-1}$.

$$ {}^{c}\boldsymbol{\phi}^{*}_{c-1} = \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{\operatorname{argmin}} \sum_{i}^{N} \frac{1}{2}\left(\mathbf{d}^{T}_{i}\left({}^{c}\mathbf{R}_{c-1}\left({}^{c-1}\boldsymbol{\Sigma}_{\mathbb{C}(i)} + {}^{c-1}\boldsymbol{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}\right){}^{c}\mathbf{R}_{c-1}\right)^{-1}\mathbf{d}_i\right) \tag{5.8} $$

The covariance matrix ${}^{c-1}\mathbf{\Sigma}_{\mathbb{C}(i)} + {}^{c-1}\mathbf{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}$ is a symmetric positive semi-definite matrix by construction. Its Eigen decomposition provides a diagonal matrix $\Lambda_{\mathbb{C}(i)}$ with non-negative singular values along the diagonal and an orthogonal matrix ${}^{c-1}\mathbf{U}_{\mathbb{C}(i)}$ (see [39][Eq. 459]) whose columns are the eigenvectors of ${}^{c-1}\mathbf{\Sigma}_{\mathbb{C}(i)} + {}^{c-1}\mathbf{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}$.

$$ {}^{c}\boldsymbol{\phi}^{*}_{c-1} = \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{argmin} \frac{1}{2} \sum_{i}^{N} \mathbf{d}_i^T \left( \left( {}^{c}\mathbf{R}_{c-1} {}^{c-1}\mathbf{U}_{\mathbb{C}(i)} \right) \mathbf{\Lambda}^{-1}_{\mathbb{C}(i)} \left( {}^{c}\mathbf{R}_{c-1} {}^{c-1}\mathbf{U}_{\mathbb{C}(i)} \right)^T \right) \mathbf{d}_i \tag{5.9} $$

$\left( {}^{c}\mathbf{R}_{c-1} {}^{c-1}\mathbf{U}_{\mathbb{C}(i)} \right)$ is a square matrix, whose columns represent the eigenvectors of the rotated covariance ${}^{c}\mathbf{\Sigma}_{\mathbb{C}(i)} + {}^{c}\mathbf{\Sigma}^{\text{unc}}_{\mathbb{C}(i)}$. Equation 5.9 can then be reinterpreted as squared distance of a point ${}^{c}\mathbf{x}_i$ to a mean $\left( {}^{c}\mathbf{R}_{c-1} {}^{c-1}\boldsymbol{\mu}_{\mathbb{C}(i)} + {}^{c}\mathbf{t}_{c-1} \right)$ along the rotated eigenvectors $\left( {}^{c}\mathbf{R}_{c-1} {}^{c-1}\mathbf{U}_{\mathbb{C}(i)} \right)$ weighted by the elements of $\mathbf{\Lambda}^{-1}_{\mathbb{C}(i)}$,

$$ \begin{aligned} {}^{c}\boldsymbol{\phi}^{*}_{c-1} &= \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{argmin} \frac{1}{2} \sum_{i}^{N} \sum_{j}^{3} \left( \frac{1}{\sqrt{\lambda_{\mathbb{C}(i),j}}} \mathbf{d}_i^T {}^{c}\mathbf{R}_{c-1} {}^{c-1}\mathbf{u}_{\mathbb{C}(i),j} \right)^2 \\ &= \underset{{}^{c}\boldsymbol{\phi}_{c-1}}{argmin} \frac{1}{2} \sum_{i}^{N} \mathbf{r}_i^2 \end{aligned} \tag{5.10} $$

The cost function described in Eq. 5.10 represents a non-linear least squares optimization problem, where $\mathbf{r}_i$ is the residual of $i^{\text{th}}$ point in ${}^{c}\mathbf{X}$. We can therefore employ a Levenberg-Marquardt optimization routine to minimize the total residual of ${}^{c}\mathbf{X}$ using a local parameterization of motion parameters (see Sec. 3.4) $\Delta {}^{c}\boldsymbol{\phi}_w := [\Delta\omega_x, \Delta\omega_y, \Delta\omega_z, \Delta t_x, \Delta t_y, \Delta t_z]$, where $\boldsymbol{\omega}$ represent the rotation parameters, $\mathbf{t}$ represent the translation parameters, ${}^{c}\boldsymbol{\phi}^{*}_{c-1} \in \mathbf{se}(3)$ and

$$ \exp({}^{c}\boldsymbol{\phi}^{*}_{c-1}) = {}^{c}\mathbf{T}^{*}_{c-1} \in \mathbb{SE}(3) $$

The Jacobian for the residual, $\mathbf{r}_i$ of $i^{\text{th}}$ point in the point cloud with respect to the elements of local parameterization is given by

$$\mathbf{J}_i = \frac{\partial \mathbf{r}_i}{\partial \Delta^c \boldsymbol{\phi}_{c-1}} = \frac{1}{2\mathbf{r}_i} \frac{\sum_1^3 \lambda_{\mathbb{C}(i),j}^{-0.5} \partial \left( \mathbf{d}_i^T {}^c\mathbf{R}_{c-1} {}^{c-1}\mathbf{u}_{\mathbb{C}(i),j} \right)}{\partial^c \boldsymbol{\phi}_{c-1}} , \tag{5.11}$$

$$\frac{\partial \mathbf{d}_i^T}{\partial^c \boldsymbol{\phi}_{c-1}} = \left[ \left[ {}^c\mathbf{R}_{c-1} \boldsymbol{\mu}_{\mathbb{C}(i)} + {}^c\mathbf{t}_{c-1} \right]_\times \mid \mathbf{I}_3 \right] , \text{ and} \tag{5.12}$$

$$\frac{\partial^c \mathbf{R}_{c-1} {}^{c-1}\mathbf{u}_{\mathbb{C}(i),j}}{\partial^c \boldsymbol{\phi}_{c-1}} = \left[ \left[ {}^c\mathbf{R}_{c-1} {}^{c-1}\mathbf{u}_{\mathbb{C}(i),j} \right]_\times \mid \mathbf{0}_3 \right] . \tag{5.13}$$

Using the residuals in Eq. 5.10 and Jacobian vectors in Eq. 5.11 we can solve the following linear system iteratively until convergence:

$$\left( \mathbf{H} + \alpha \text{Diag}(\mathbf{H}) \right) \Delta^c \boldsymbol{\phi}_{c-1} = \mathbf{b} \tag{5.14}$$

$$\mathbf{H} = \sum_i^N \mathbf{J}_i^T \mathbf{J}_i, \quad \mathbf{b} = -\sum_i^N \mathbf{J}_i^T \mathbf{r}_i . \tag{5.15}$$

In real world, the correspondence function $\mathbb{C}(i)$ is not exactly known. The projective correspondence computation described in Sec. 4.1.3 finds the best correspondences given an estimate of ${}^c\mathbf{T}_{c-1}$. Thus, similar to an expectation maximization routine, given a good initialization for the transformation parameters, ${}^c\mathbf{T}_{c-1}$, we iteratively:

- compute a set of projective correspondences (Sec. 4.1.3
- using the fixed set of correspondences compute the best transformation parameter estimate using Eq. 5.15,

until convergence.

## 5.3 Implementation

We assumed the sensor uncertainty model used in Sec. 5.2.2 to be a zero mean Gaussian distribution. Empirically computed noise models may have some bias and variance components. A

Gaussian noise model with bias $\mathbf{b}$ and variance $\mathbf{\Sigma}^{\text{unc}}$ can be reinterpreted as a zero mean Gaussian distribution as shown in Sec. 4.1.8. The sensor noise model described in Chapter 4 is used here, that consists of an empirically derived noise component along the direction of the ray and two components describing the pixel quantization error in the pixel space.

The pose estimation approach described in Sec. 5.2.2 converges to the closest local minima of the log-likelihood cost function. Therefore, the convergence is highly influenced by the map parameters ${}^{w}\mathbf{\Theta}_t^k$. As described in Chapter 4, the map representation at the highest fidelity represents the sensor observations very accurately, and therefore, may contain high frequency noise information that influences the pose estimation cost function negatively. However, the Gaussian distributions at hierarchical level 1 provide a smoother map representation, that eliminates the high frequency map information while representing the structure information in the scene with high accuracy. Therefore, for the pose estimation framework, we compute the log-likelihood cost of ${}^{c}\mathbf{X}$ with respect to the hierarchical level 1 global model ${}^{w}\mathbf{\Theta}_t^1$. Additionally, in real world scenarios, we observe that the non-uniform scaling of the distinct Gaussian distributions given by their eigenvalues ($\mathbf{\Lambda}$) in Eq. 5.9 causes the cost function to diverge in complex scenarios. As described in Sec. 2.1.1, the measurements obtained from depth sensors are spread along the surfaces of objects in the scene. Therefore, for a standard Gaussian distribution,

$$\lambda_0 << \lambda_1 < \lambda_2$$

where $\lambda_0$ represents the eigenvalue along the direction of the normal of the represented surface. Due to noisy sensor information, this structure often causes numerical instabilities often leading to the covariance matrix to become non PSD. To avoid such numerical inconsistencies, we use a uniform scaling that provides a more stable and smoother cost function for the pose estimation

pipeline. In practice, we use a fixed $\mathbf{\Lambda}$:

$$\mathbf{\Lambda} = \text{Diag}\begin{bmatrix} 1 & 0.001 & 0.001 \end{bmatrix}$$

This structure enables us to capture the fact that the cost function changes the most along the direction of the smallest eigenvalue. Additionally, it converts the cost function to an Euclidean distance function along the first eigenvector for every Gaussian distribution. Thus the cost function can weight each point in the live sensor scan equally and get a more robust pose estimate.

We use the similar perspective constraints described in Sec. 4.1.9 to sub-select the number of Gaussian distributions used to compute the point-to-distribution correspondence for pose estimation. This enables us to rapidly sub-select only the components of the map that are updated by the live sensor measurement efficiently and reduce the computational complexity of computing point-to-distribution correspondences.

## 5.4 Evaluation

In this section, we present results of our proposed SLAM strategy on various publicly available datasets:

1. "Living Room"[62]
2. "Lounge"[62]
3. "Plant Scene 1"[43]
4. "SFM Lab 1"[43]
5. "Freiburg2 XYZ"[52]

We compare the performance of the proposed SLAM implementation to well known ElasticFusion and KinectFusion approaches (see Sec. 2.2). First we compare the Relative Position Error (RPE) and Absolute Trajectory Error (ATE) of our algorithm, ElasticFusion and KinectFusion

with ground truth and show the superior performance of our algorithm on all the different datasets. ATE compares the overall performance of a global SLAM approach for trajectory tracking and highlights the reliability and robustness of the SLAM approaches to various sensor measurements. RPE on the other hand compares the frame-to-frame tracking error, thus providing a consistent metric of comparison even when the absolute position may get lost due to an incorrect position estimate. We also show the qualitative 3D reconstruction of these environments generated from our SLAM approach to show the consistency of our mapping and tracking pipelines. ElasticFusion is a surface primitive based SLAM algorithm and KinectFusion is a volumetric TSDF based SLAM approach as described in Sec. 2.2. Comparison with these two approaches highlights the benefits of using our proposed map representation that add volumetric properties to a disconnected set of structure primitives.

### 5.4.1 3D Reconstruction and Trajectory Tracking

We first compare the trajectory tracking performance of our SLAM approach with ElasticFusion and KinectFusion on publicly available datasets. First we evaluate the performance on the simulated "Living Room" dataset. This dataset provides ground truth locations for a camera moving through a synthetically generated real looking environment. Further, the 3D ground truth model is also provided, that enables us to compute the 3D reconstruction accuracy. Reconstruction error is computed as the distance of a reconstructed point to its closest surface in the ground truth model. Table 5.1 and Fig. 5.2 compare the tracking and mapping performance of the SLAM systems. Our trajectory tracking approach outperforms the state-of-the-art SLAM approaches while using a more compressed map representation. Additionally, the 3D reconstruction provided by our approach is very similar to the provided ground truth model (Fig. 5.3, Fig. 5.4) and performs as well as the state-of-the-art approaches. We compute the reconstruction accuracy by sampling points from the Gaussian distributions and computing the error of the sampled points. Gaussian distribu-

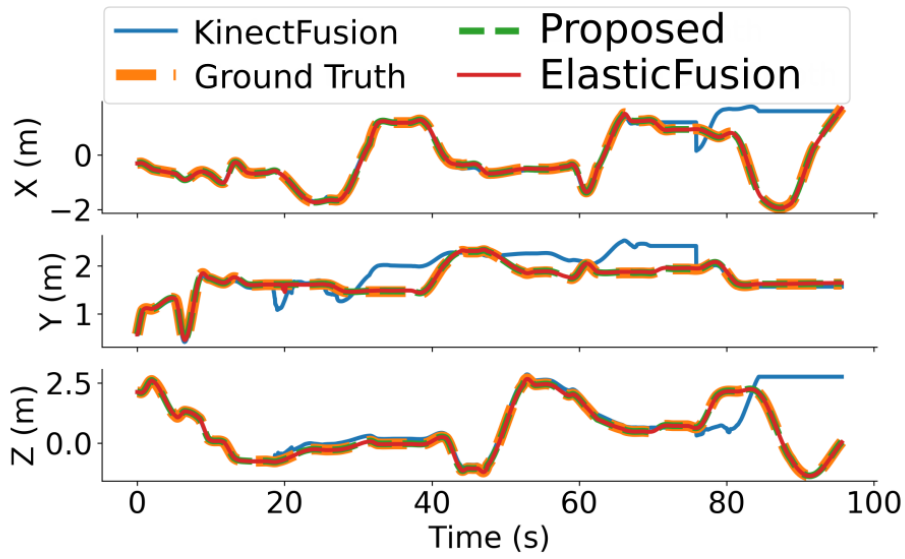



*Figure 5.2: Trajectory tracking comparison on "Living Room" dataset. The proposed approach has the lowest trajectory tracking error compared to ElasticFusion and KinectFusion. KinectFusion estimate drifts off due to incorrect handling of degenerate scenarios where the sensor is not observing textured information.*

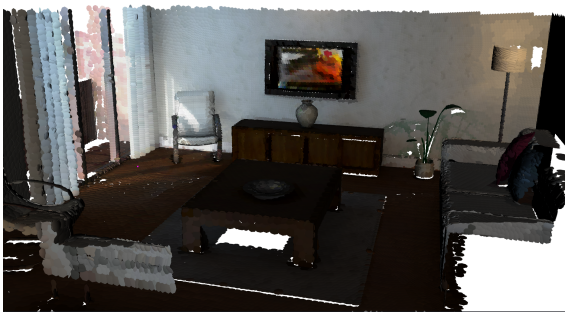
tions have infinite support and therefore resampled points at the edges of the map leads to a larger reconstruction error of the proposed SLAM approach compared to ElasticFusion. However, the proposed map representation consumes orders of magnitudes lesser memory than state-of-the-art SLAM approaches with a marginal accuracy trade-off as shown in Table 5.1.

| Method | ATE (m) | ATE (deg) | RPE (m) | RPE (deg) | Reconstruction Error (m) | Memory (MB) |
|---|---|---|---|---|---|---|
| Proposed | **0.0112** | **0.0042** | **0.0002** | 0.0006 | **0.0062**<br>0.0072<br>0.0084 | 3.5148<br>.5142<br>**0.0645** |
| KinectFusion | 2.6938 | 4.2423 | 0.0150 | 0.0144 | 0.2097 | 67.108 |
| ElasticFusion | 0.0233 | 0.0140 | 0.0048 | **0.0001** | 0.0064 | 69.2774 |

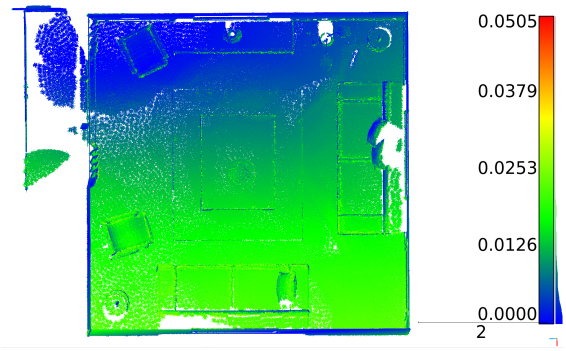
*Table 5.1: Surface Reconstruction and Trajectory Tracking Error for "Living Room" dataset.*

### 5.4.2 Trajectory Tracking Performance

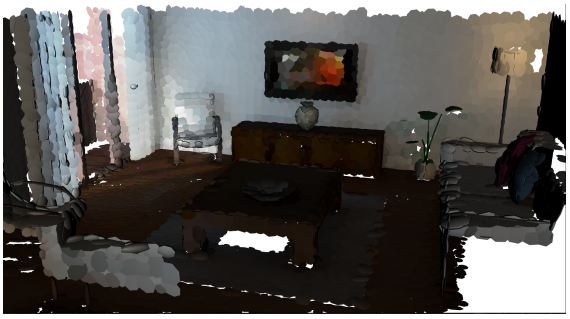
We evaluate the trajectory tracking performance of ElasticFusion, KinectFusion and our proposed approach on a variety of publicly available real-world datasets. The 3D reconstruction obtained from our framework on multiple datasets is shown in Fig. 5.5. The ground truth models are not



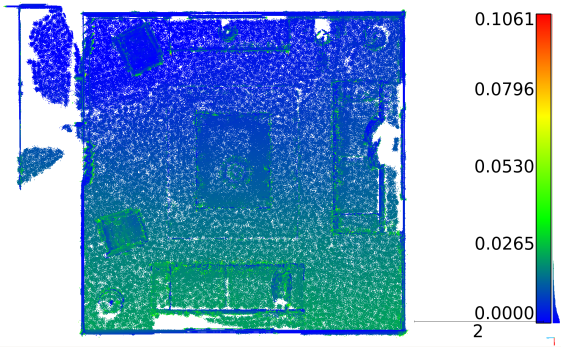
(a) Reconstruction $\mathbf{\Theta}_T^0$



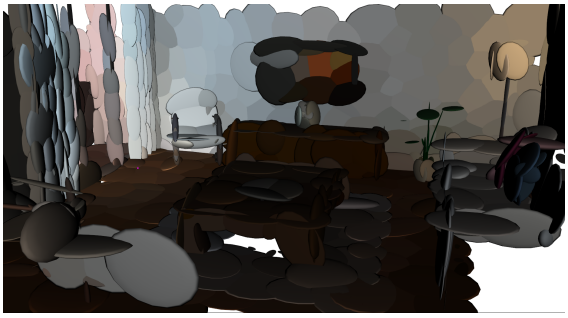

(b) Sampled 3D points from $\mathbf{\Theta}_T^0$



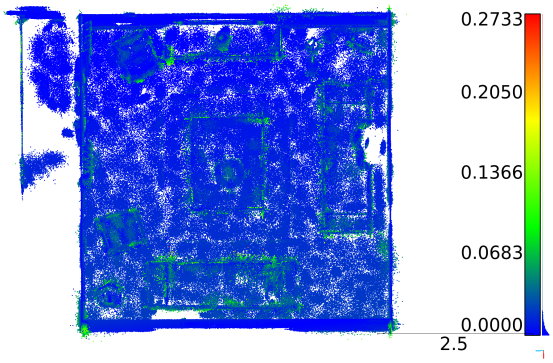
(c) Reconstruction $\mathbf{\Theta}_T^1$




(d) Sampled 3D points from $\mathbf{\Theta}_T^1$



(e) Reconstruction $\mathbf{\Theta}_T^2$




(f) Sampled 3D points from $\mathbf{\Theta}_T^2$

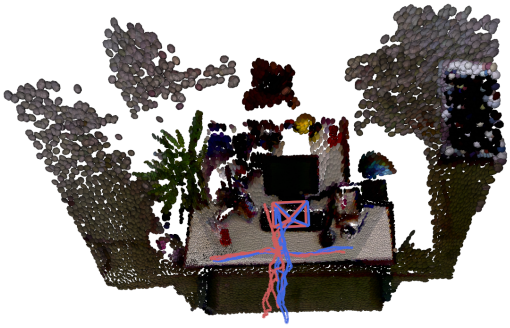
*Figure 5.3: Gaussian distribution based reconstruction of "Living Room" dataset obtained from the proposed SLAM framework. Left Column: Gaussian distributions at the three hierarchical levels obtained from our SLAM framework. Right column: 3D points sampled from Gaussian distributions at the three hierarchical levels colored by the error in reconstruction.*


*Figure 5.4: Surfel based 3D reconstruction of "Living Room" dataset obtained from ElasticFusion SLAM framework. Right: Surfel centers colored according to their reconstruction error.*

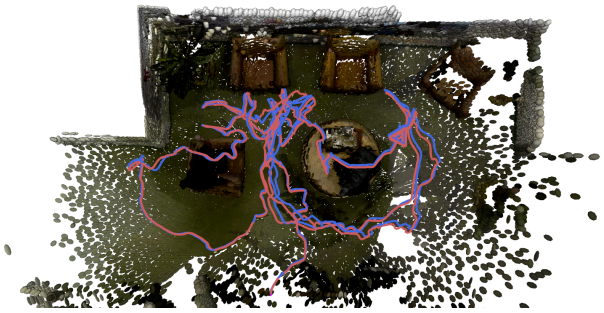
available for these datasets therefore, we cannot compare the 3D reconstruction accuracy. Table 5.2 provides the statistics for frame-to-model trajectory tracking for the SLAM approaches and Fig. 5.6 and Fig. 5.7 show the $X - Y - Z$ plots of the tracked trajectories. We compare both, the Absolute Trajectory Error (ATE) RMSE and the Relative Pose Error (RPE) RMSE. Our SLAM approach outperforms the state-of-the-art SLAM approaches in terms of the ATE. The RPE of the proposed approach is larger than KinectFusion and ElasticFusion for the "Lounge" dataset by $\approx 2$ mm. However, the ground truth trajectory provided for this dataset has loop closure corrections incorporated in the ground truth trajectory. Therefore, the RPE value for this dataset is unreliable. From Fig 5.6 and Fig. 5.7, we observe that since our proposed algorithm incorporates the sensor uncertainty into the localization framework, it provides more robust and reliable localization estimates over a variety of datasets, whereas state-of-the-art SLAM algorithms ElasticFusion and KinectFusion fail in certain datasets to provide an accurate localization estimate.

### 5.4.3 Memory Scaling

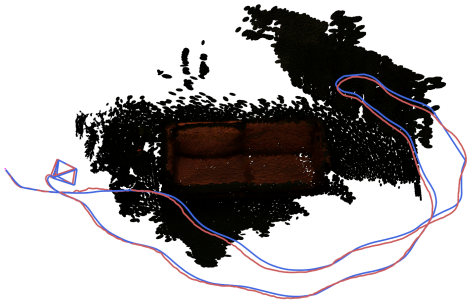
Although the Gaussian components require less elements to represent the same map area than comparable approaches, the memory usage over time naturally increases linearly. Crucially, however, due to working with a hierarchy, only a subset of the entire map needs to be retained in active GPU



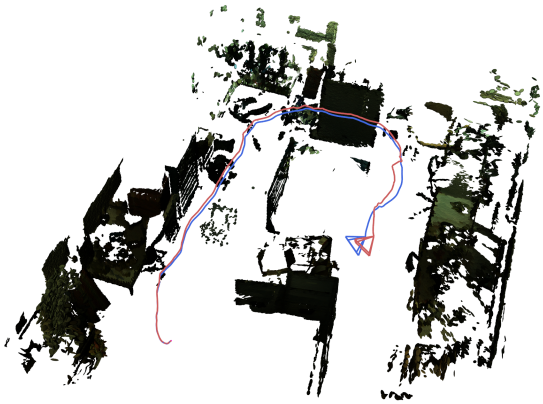
(a) “Freiburg2 XYZ”



(b) “Lounge”



(c) “Plant Scene 1”



(d) “SFM Lab 1”

*Figure 5.5: 3D reconstruction obtained from our SLAM framework on various datasets. The coloured ellipsoids represent 3D Gaussian distributions that are incorporated as new sensor measurements are observed and registered to the global frame. The blue line is the ground truth trajectory and the red line is the estimated trajectory from the proposed pipeline.*

and CPU memory thus for most scenes the number of active components at any given time across hierarchy levels remain constant, as shown in Fig. 5.8

### 5.4.4 Runtime Analysis

The SLAM framework proposed in this thesis is implemented on a Desktop CPU. Due to the iterative nature of the localization section, it is the computational hotspot of the algorithm. However, the residual computation for each point in the depth measurement is independent of the others and therefore this section is highly parallelizable. The pose estimation section was implemented

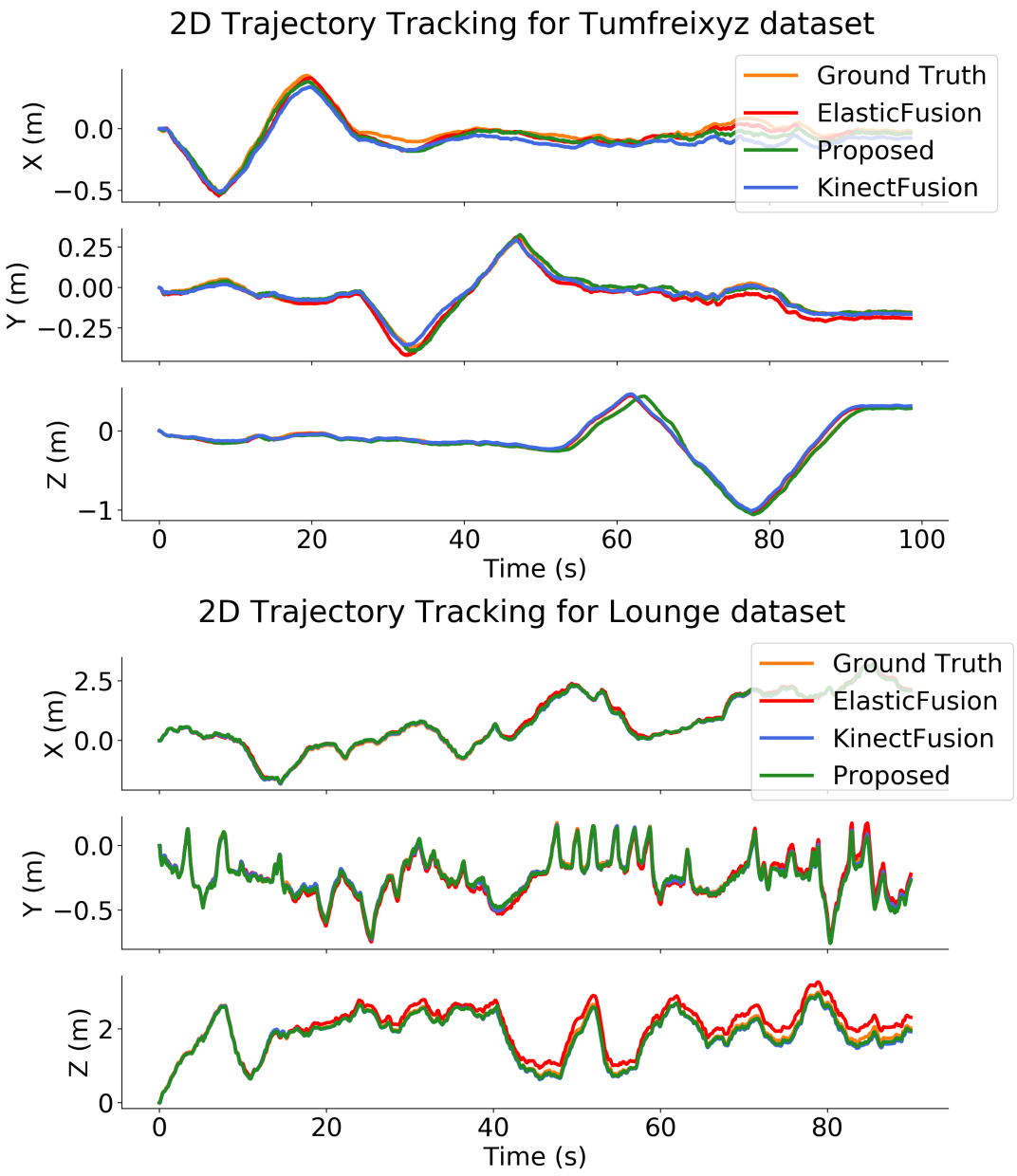



*Figure 5.6: Trajectory tracking performance comparison of the proposed SLAM approach with KinectFusion and ElasticFusion on the real-world datasets: Top: Freiburg2 XYZ dataset, Bottom: Lounge dataset. All the compared approaches track the camera trajectory well. However, the proposed SLAM approach performs most robustly and the tracks the camera motion with highest accuracy in datasets with varying information fidelity and obtained from different sources.*

on a NVidia GTX 980 Ti GPU with the rest of the mapping and map update framework on a i7 CPU. Figure 5.9, highlights the average execution time statistics for the various subcomponents of our proposed SLAM approach on the "Living Room" dataset. The pose estimation subsystem

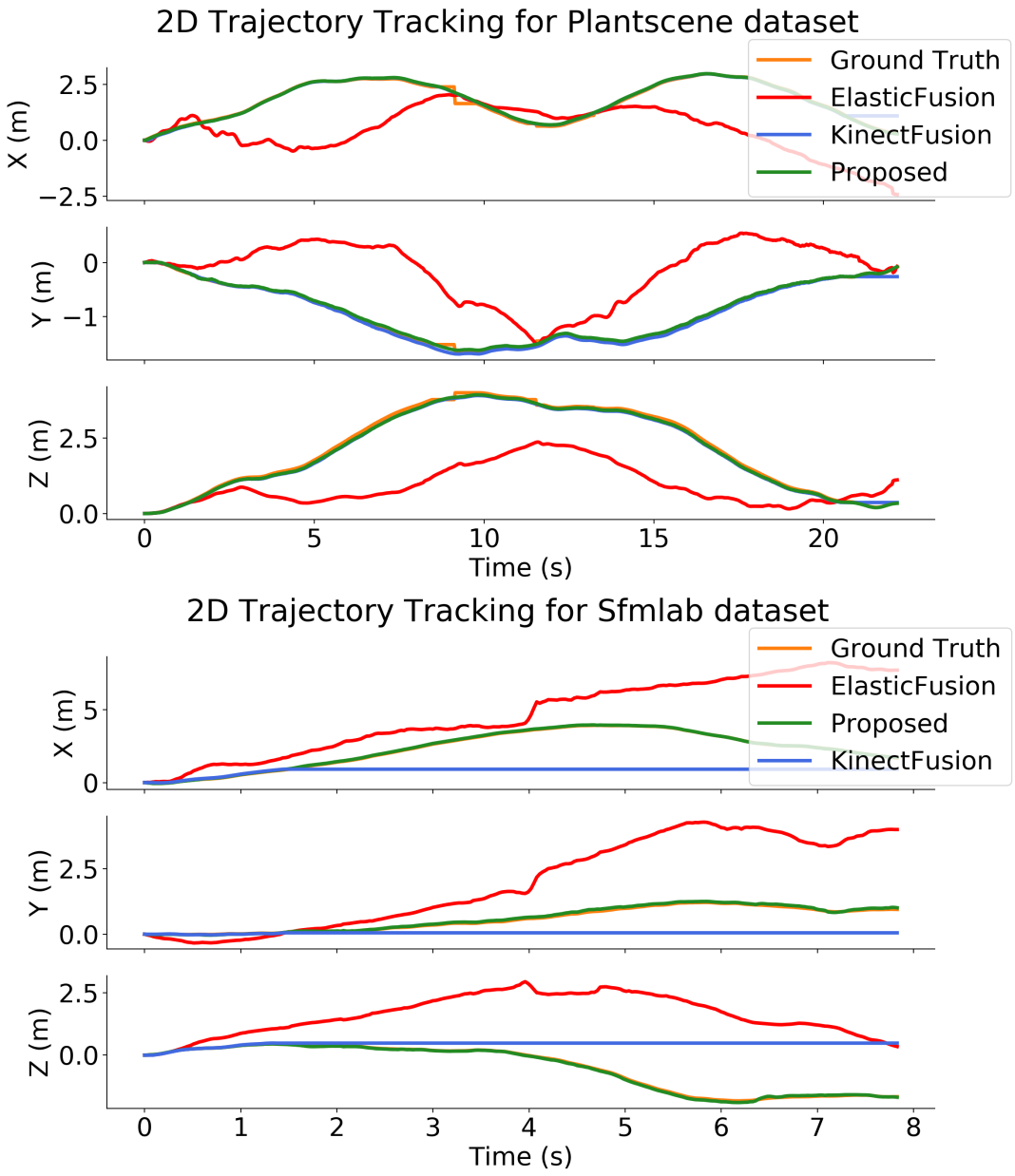



*Figure 5.7: Trajectory tracking performance comparison of the proposed SLAM approach with KinectFusion and ElasticFusion on the real-world datasets: Top: Plant Scene dataset and Bottom: SFM Lab 1 dataset. ElasticFusion fails to track the camera motion for both these datasets due to the sparsity of available in these datasets. KinectFusion and proposed approach track the camera motion reliably with the proposed approach achieving the highest tracking accuracy.*

still requires the largest computational resources and operates at approximately 5 Hz. The bars in red show the run-time of correspondence map computation implemented using OpenGL. Due to the difference in the information rendered for pose estimation and map update, the average

| Dataset | Method | ATE (m) | ATE (deg) | RPE (m) | RPE (deg) |
|---|---|---|---|---|---|
| "Plant Scene 1" | Proposed | **0.1062** | **0.0813** | **0.0305** | **0.01493** |
| | KinectFusion | 0.1829 | 0.0876 | 0.0390 | 0.0152 |
| | ElasticFusion | 2.504 | 1.6330 | 0.0389 | 0.0193 |
| "SFM Lab" | Proposed | **0.0596** | **0.5509** | **0.0104** | **0.0073** |
| | KinectFusion | 2.3631 | 2.6424 | 0.0308 | 0.0361 |
| | ElasticFusion | 4.2693 | 3.3090 | 0.0709 | 0.0358 |
| "Lounge" | Proposed | **0.0784** | **0.4135** | 0.0055 | 0.0047 |
| | KinectFusion | 0.0899 | 0.6450 | **0.0037** | **0.0025** |
| | ElasticFusion | 0.2109 | 0.7119 | 0.0048 | 0.0028 |
| "Freiburg2 XYZ" | Proposed | **0.0481** | **0.0364** | **0.0017** | **0.0042** |
| | KinectFusion | 0.1007 | 0.0677 | 0.0024 | 0.0053 |
| | ElasticFusion | 0.0720 | 0.0455 | 0.0023 | 0.0052 |

*Table 5.2: 3D Trajectory tracking error comparison of our approach with state-of-the-art SLAM approaches on multiple real-world datasets.*

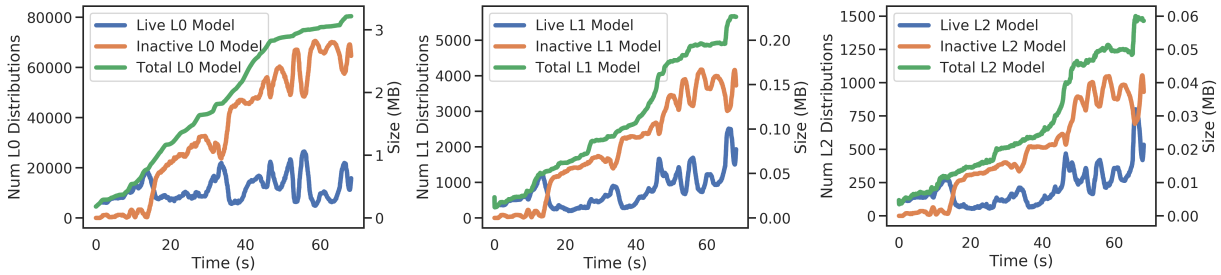



*Figure 5.8: Number of active, inactive and total number of Gaussian distributions over time that are maintained with time during the SLAM framework execution on "Living Room" dataset. The number of total distributions increases with time as more information is observed about the scene time. However, during any time step t, a majority of the scene is not visible in the sensor FOV. Therefore, the number of active components at every hierarchical level stays approximately constant over time as shown by the orange line while the number of inactive components rises.*

runtime is different for the two correspondence map computations. The entire SLAM framework (map updates and localization) operates at an average rate of 4 Hz. However, the reduction in the computational complexity of the proposed SLAM pipeline as compared to the state-of-the-art approaches leaves room for improvement in terms of the algorithmic runtime.

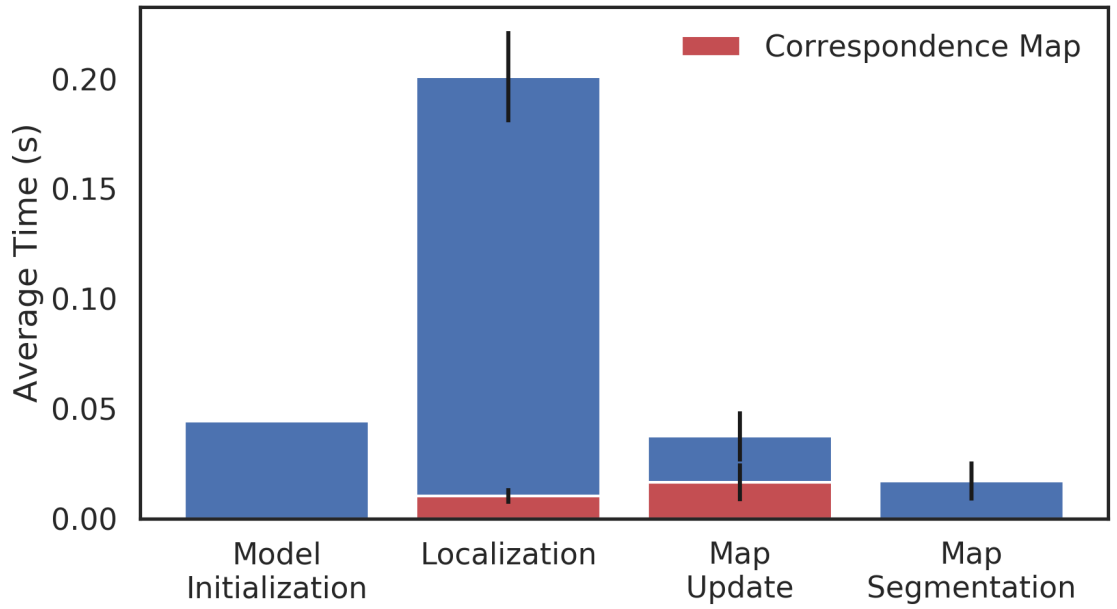



*Figure 5.9: Average runtime statistics of the various subcomponents of our proposed SLAM algorithm on the "Living Room" dataset. The statistics are collected on an Intel i7 Desktop grade CPU with the localization component running on a NVidia GTX 980 Ti GPU. On an average, the algorithm operates at $\approx 4$ Hz with the localization component proposed in this chapter, operating at $5$ Hz.*

# Chapter 6

# Applications

In Chapter 4, we proposed a hierarchical model reconstruction pipeline using independent Gaussian distributions as structure primitives. We represent the scene at varying information fidelity at different hierarchical levels with the hierarchical level $2$ map being the lowest fidelity and level $0$ being the highest. In Chapter 5, a local frame-to-model localization framework is proposed that utilizes the hierarchical map to efficiently track a moving camera in an unknown scene and generate a consistent $3$D reconstruction of that scene. The log-likelihood of each point $\mathbf{x}_i$ is computed using a single Gaussian distribution ${}^{w}\boldsymbol{\theta}_j$ that the point ${}^{w}\mathbf{x}_i$ corresponds to. However Gaussian distributions have infinite support space. Therefore, the probability densities of the Gaussian distributions used to reconstruct the scene overlap with each other. A more accurate cost function must therefore, compute the log-likelihood of a point $\mathbf{x}_i$ with respect to all the Gaussian distributions in the scene.

In this chapter, we extend the localization framework to a global scope and propose a multi-hypothesis localization framework [12] enabled on a SWaP constrained system, when the $3$D reconstruction of the world is known a-priori. We fit a low fidelity GMM ${}^{w}\bar{\boldsymbol{\Theta}}$ to the global point cloud data of the world. Global localization is then formulated as a Monte-Carlo log-likelihood maximization framework with respect to a GMM with fixed model complexity. Additionally, we describe a collision avoidance framework [13] that uses Gaussian distributions to enable safe nav-

igation for an autonomous system in an unknown environment.

## 6.1 Gaussian Mixture Model (GMM)

A GMM, $\bar{\Theta}$ is defined as a set of Gaussian distributions $\bar{\theta}_i := \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \pi_i\}$ where $\pi_i$ is the corresponding importance weight such that $\sum_i^M \pi_i = 1$. The likelihood of a point $\mathbf{x}$ in $d$ dimensional space having been sampled from a GMM $\bar{\Theta}$ is:

$$p\left(\mathbf{x} \mid \bar{\Theta}\right) = \sum_i^M \pi_i p\left(\mathbf{x} \mid \bar{\boldsymbol{\theta}}_i\right) \tag{6.1}$$

## 6.2 Pose Estimation and Localization

For an agent lost in a known environment, much of the cost of localization can be offset by pre-computing measures of what the sensor is expected to see; localization can then be cast as the much simpler problem of a search through these pre-existing "hallucinated" views. However, exhaustively considering all possible views incurs a prohibitive cost that increases exponentially with both the dimensionality of the state space and the size of the environment. Further, naïve pre-rendering approaches can be susceptible to errors caused by perceptual aliasing due to slight variations in the environment appearance or by regions that are not feature rich, such as blank walls [3].

In this section, we present a framework enabling rapid computation of sensor data likelihood via a GMM. The framework models a depth camera observation as being sampled from the GMM with a likelihood measure that varies smoothly about the true camera pose. We thus exploit the dramatically reduced storage complexity of the representation and the local spatial regularity of a fast-to-compute likelihood function to re-cast the pose estimation problem as that of Monte-Carlo localization [56].

Our framework solves the problem of 6 Degree-of-Freedom pose estimation for Size, Weight,

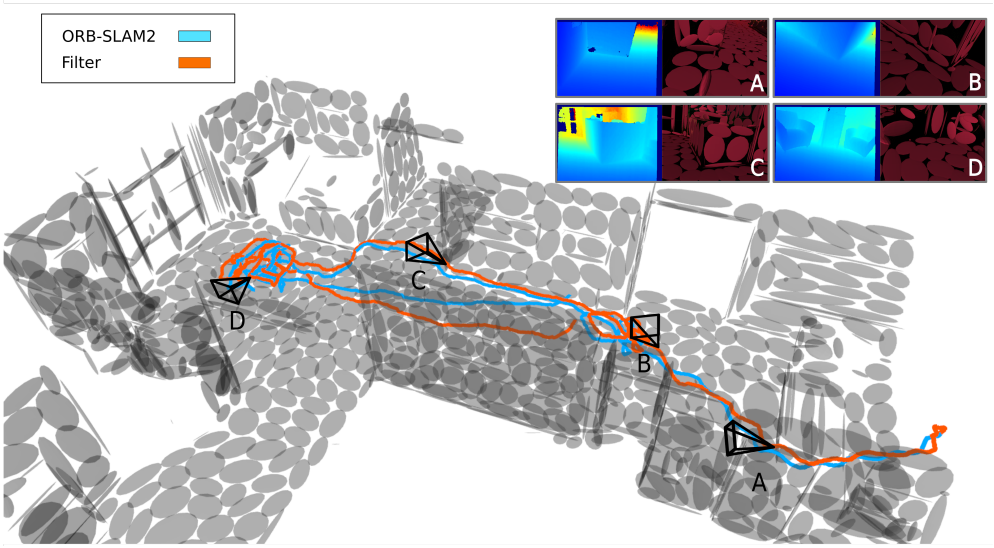



*Figure 6.1: Comparison of the mean particle filter pose (orange) with that of the integrated process model trajectory (cyan) from a representative (office) dataset. The filter estimate is initialized with a uniform distribution away from the true location of the vehicle. As the camera observes more informative data the filter quickly converges to the correct pose. Top Right: Four views of the raw point cloud sensor data and the corresponding view of the GMM map from the mean of the particle filter estimates. The GMM components are superimposed on top of the source point cloud with their* $1.5\sigma$ *bounds visualized as gray ellipsoids.*

and Power (SWaP) constrained micro air vehicles operating in a known dense 3D point cloud environment with an on-board monocular depth camera and Inertial Measurement Unit (IMU). We assume that the vehicle pitch and roll are obtained from an attitude estimation algorithm using the IMU in order to constrain the search space to just heading and position. Our main contributions are:

- A particle filter-based localization strategy based on a high fidelity, memory efficient environment representation enabled by a fast likelihood computation approximation; and
- Experimental evaluation of the approach on a desktop and an off-the-shelf mobile GPU system.

A detailed overview of our global localization framework is shown in Fig. 6.3.

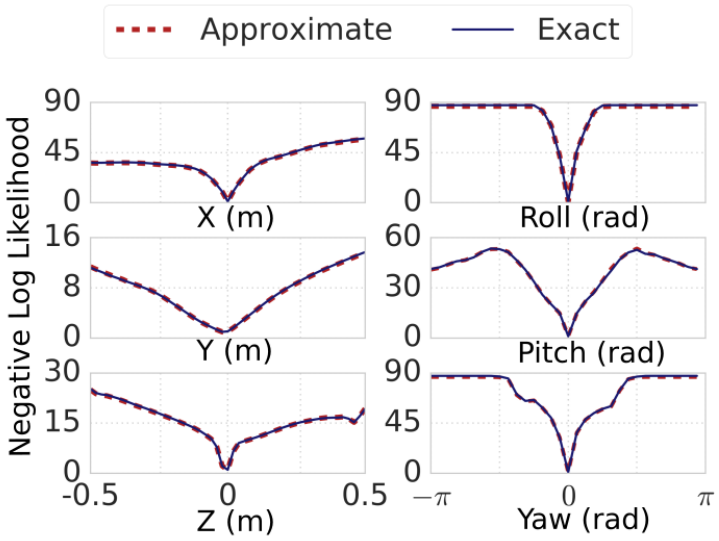



*Figure 6.2: Negative log-likelihood plots of sensor data acquired from camera poses offset from a randomly chosen true pose in dataset D1 by incremental linear and rotational displacements. Utilizing only the relevant components using the approximation discussed in Sec. 6.2.3 leads to almost identical likelihoods as when utilizing all the Gaussian components present in the model.*

### 6.2.1 Estimating the Likelihood of a Camera Pose Hypothesis

As discussed in Sec. 4.1.3, each 3D Gaussian distribution, $\bar{\theta}$ in a GMM ${}^{w}\bar{\Theta}$ can be projected into the image plane as a 2D ellipse. We utilize this property to determine relevant components for computing the likelihood of sensor data (Sec. 6.2.3). Given a scan ${}^{c}\mathbf{X}_t$ of depth pixels $\{x_1, x_2, \ldots, x_k\}$ from a sensor scan and a set of 3D GMM parameters ${}^{w}\bar{\Theta}$, the log-likelihood of the scan being sampled from the GMM is defined as

$$
\begin{aligned}
&l\left({}^{c}\mathbf{X}_t | {}^{w}\bar{\Theta}, \mathbf{T}_w^c\right) = \\
&\sum_{i}^{K} \ln \sum_{j}^{M} \mathbf{1}_j \pi_j p\left(\mathbf{\Pi}^{-1}(x_i); {}^{c}\mathbf{T}_w \boldsymbol{\mu}_j, {}^{c}\mathbf{R}_w \boldsymbol{\Sigma}_j {}^{c}\mathbf{R}_w{}^{\mathrm{T}}\right)
\end{aligned}
\tag{6.2}
$$

where $\mathbf{1}_i$ is a binary indicator function that signifies if the $i^{\text{th}}$ component is used to compute the log-likelihood, $\mathbf{\Pi}^{-1}$ is the inverse projection from depth image pixel to 3D points, and $K$ is the number of pixels in the sensor scan. This likelihood should peak at the true sensor pose and decay

smoothly in the local neighbourhood, which is indeed observed as shown in Fig. 6.2.

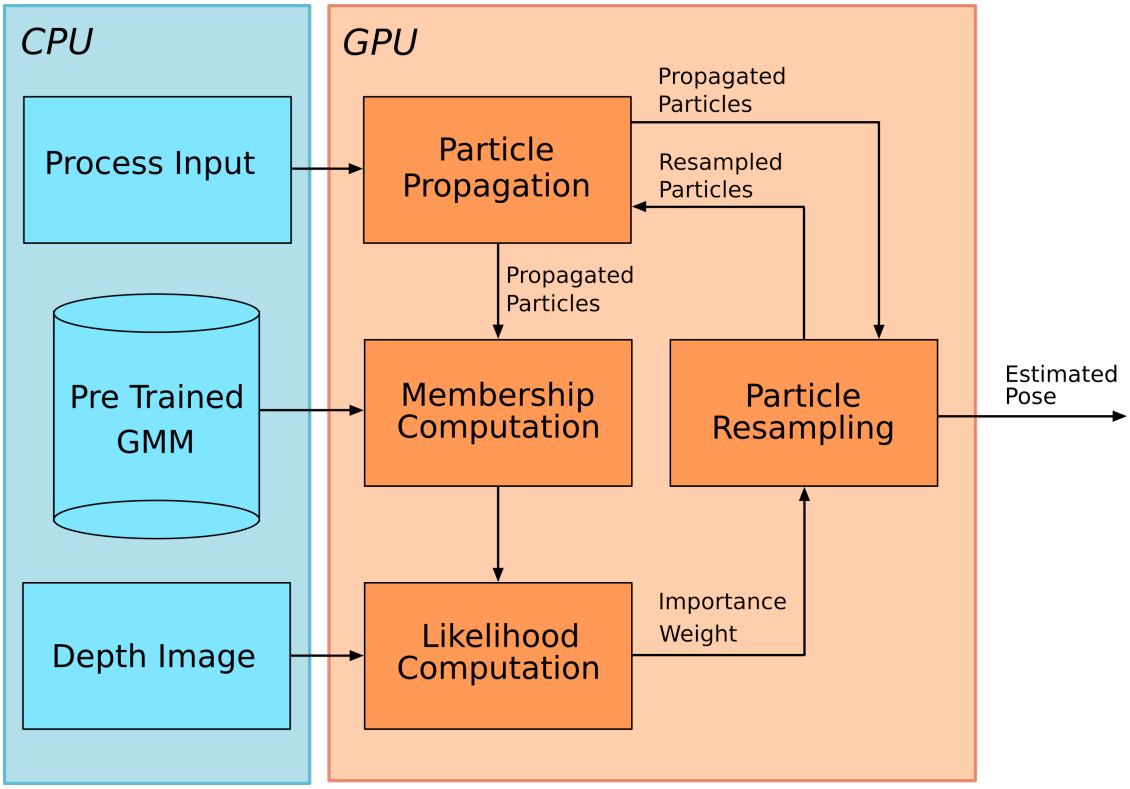



*Figure 6.3: Multi-hypothesis particle filter system overview. The algorithm operates on depth image streams and a source of odometry given a precomputed GMM map of the environment. For each particle, the GMM components are projected into image space using its current pose hypothesis. Relevant components are sub-selected and are then used to compute the likelihood of the depth image. The likelihood values for all the particles are used to resample a new set of particles that are then forward propagated using the process model.*

### 6.2.2 Tracking Multiple Hypotheses

The discussion above only considers the nature of log-likelihood in the vicinity of the true location; in practice it is not reasonable to assume that a single viewpoint suffices to localize the system as perceptual aliasing may arise due to a paucity of data that precludes state observability. Hence, we require a technique that permits tracking of multiple hypotheses and ensures appropriate weighting of equally likely viewpoints given the current sensor observations.

A standard approach to tracking multiple hypotheses is a Monte-Carlo filter (or particle filter). Particle filters operate by continuously sampling candidate particle poses and measure the likeli-

hood of the current sensor data having originated at the sampled pose. Based on the relative scores of the samples the particles are resampled and propagated subject to a process model (often a noisy source of odometry). Convergence is generally achieved as soon as the sequence of observations made over time render alternate hypotheses inadmissible. Note that due to their inherent structure particle filters are extremely parallelizable and we exploit this in our implementation.

### State Propagation

We assume the presence of some odometry to drive the first order Markov process model and inject Gaussian noise into it. Note that we assume that we know the pitch and roll that can be obtained from the attitude and heading reference system onboard a robotic system to a high level of accuracy.

### Importance Weight

The importance weight of a particle in the filter represents a score of how well the sensor scan matches the GMM map at its location. Since the negative log-likelihood of the current scan ${}^{c}\mathbf{X}_t$ being drawn from the GMM map is a minimum at the true location, as shown in Fig. 6.2, in practice we use the inverse of the negative log-likelihood. Thus, given the current state estimate ${}^{c^{(i)}}\mathbf{T}_w$ of a particle $i$ out of $N$ particles at time step $t$, the corresponding normalized importance weight is

$$w_t^{(i)} = \frac{l\left({}^{c}\mathbf{X}_t | {}^{\mathbf{w}}\bar{\mathbf{\Theta}}, {}^{c^{(i)}}\mathbf{T}_w\right)^{-1}}{\sum_j^N l\left({}^{c}\mathbf{X}_t | {}^{\mathbf{w}}\bar{\mathbf{\Theta}}, {}^{c^{(i)}}\mathbf{T}_w\right)^{-1}} \tag{6.3}$$

### Sampling Strategy

A particle filter should ideally converge to the correct hypothesis after running for a finite amount of iterations with a reduction in the filter variance signifying the confidence of the filter. At the same time, an early reduction in the filter variance may cause the filter to diverge to an incorrect hypothesis and never recover due to low variance. In order to avoid such situations, we implement

the stratified sampling strategy [32] in combination with low variance sampling [56]. The particles are divided into random groups of equal weights and in each group we employ low variance sampling. This approach has low particle variance [56] and works well when the particle filter is tracking multiple hypotheses at once.

**Handling Particle Deprivation**

One of the most common failure modalities of a particle filter is that of particle deprivation [57]. Even with a large number of particles, the stochasticity intrinsic to a particle filter might cause it to diverge from the correct state.

We employ a modified version of Augmented MCL strategy as described by Thrun et al. [56] where instead of adding new particles we reinitialize $N_{\text{modify}}$ number of particles randomly selected from the original set using the parameters $\alpha_{\text{slow}}$ and $\alpha_{\text{fast}}$. This is done since we cannot increase the number of particles once the filter is initialized because of implementation limitations. For our process model we use diagonal covariances for translation, and the final choice of parameters in all our experiments is shown in Table 6.1.

### 6.2.3 Fast localization

In order to perform fast localization using the above approach it is essential to compute the likelihood of the data given a proposed pose as quickly as possible.

Equation 6.2 suggests that computing the likelihood of a scan having been sampled from the GMM map is the summation of the contribution of all the components within the GMM. However, the key insight here is that not all components have a significant contribution to the likelihood.

The point clouds that we use in our experiments have roughly uniform coverage of points across the scene. As a consequence, all Gaussian components fit to these point clouds end up having roughly equivalent mixture weight probabilities. This fact, in addition to the diminishing probability mass of the Gaussian distribution, permits the approximation of using only the pro-

jected components within spatial proximity of a certain pixel location for computing the likelihood of the corresponding 3D point being sampled from the map.

As an added optimization step we perform this membership computation over subdivided patches of the image. These optimizations have negligible effect on the computed likelihood value of the sensor data, as demonstrated in Fig. 6.2.

We follow the following steps (graphically illustrated in Fig. 6.4) to obtain the relevant components for computing the likelihood of a depth image:

- Divide the image into $32 \times 32$ pixel patches;
- Compute the 2D projection of each Gaussian component on to the image plane of the depth sensor;
- Inflate the $3\sigma$-bound ellipse of the projected 2D Gaussian of each component by half the diagonal of the patch along its major and minor axis to generate ellipses $\mathcal{E}_i$; and
- For each patch, check if the center of the image patch $c_p$ lies within or on each of the $\mathcal{E}_i$ and update the indicator variable $\mathbf{1}_{i,p}$ accordingly.

$$\mathbf{1}_{i,p} = \begin{cases} 1, \text{ if } c_p \in \mathcal{E}_i \\ 0, \text{ otherwise} \end{cases} \tag{6.4}$$

Given a set of updated indicator variables $\mathbf{1}_{i,p}$ for all the Gaussian components in ${}^{w}\bar{\mathbf{\Theta}}$ and a depth image, ${}^{w}\mathbf{X}_t$, the likelihood of the image can be computed as the sum of the likelihoods of all the image patches computed according to Eq. 6.2.

### 6.2.4 Evaluation

#### Experiment Design

This section presents performance analysis of our filtering approach on a variety of datasets. First, we conduct a sensitivity analysis to determine the number of particles we use in our implementa-

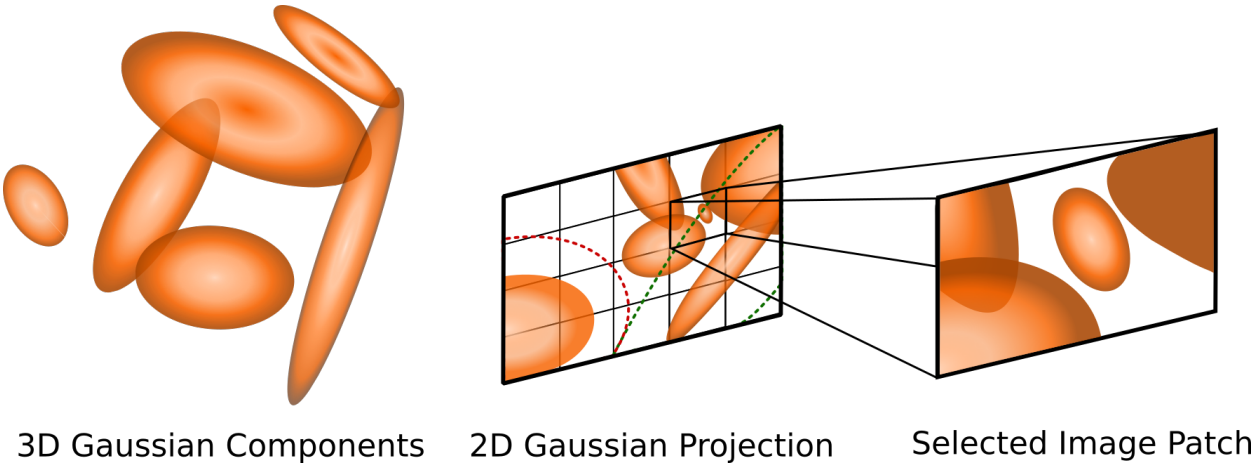



*Figure 6.4: Membership computation process. 3D Gaussian components from the GMM representation of the world are projected to the image plane. The image is subdivided into multiple patches, where for a selected patch the relevant Gaussian members are determined for computing the likelihood. In order to determine the latter, we employ the heuristic described in Sec. 6.2.3. For instance the inflated bounds of the bottom left projected component (red) do not contain the center of the selected patch; in contrast those of the bottom right (green) do, and the component is thus selected for computing the likelihood of data within that particular patch.*

tion. Second, we analyze metric accuracy of the proposed filter on publicly available datasets and show that our filter output is consistent with ground truth. Third, we compare the localization performance of our approach with a state-of-the-art RGBD tracking algorithm (ORB-SLAM2 [35]) on the same sequences and demonstrate superior performance for localization. Fourth, we demonstrate the ability of our approach to incorporate both different odometry algorithms and ground truth map acquisition methodologies. Finally, we analyze runtime performance of our filter and show that its runtime is competitive both on a desktop class system and on an embedded platform, thus enabling SWaP constrained operation.

We evaluate our approach on

- D1: The (a) lounge and (b) copyroom datasets [62];
- D2: The voxblox dataset [37];
- D3: A representative dataset collected in-situ; and
- D4: The TUM Freiburg3 dataset [52] for demonstrating the ability to generalize.

| | Process Noise $\sigma$ | | $\alpha_{slow}$ | $\alpha_{fast}$ |
|---|---|---|---|---|
| | Translation (m) | Yaw (rad) | | |
| Desktop | 0.02 | 0.01 | 0.01 | 0.001 |
| TX2 | 0.025 | 0.1 | 0.05 | 0.005 |

*Table 6.1: Filter hyperparameters*

In all cases we utilize a fixed number of components (Sec. 6.2.4) to first fit a GMM to the point cloud using the scikit-learn[1] toolkit.

We employ two processing systems for evaluation: (1) A desktop with an Intel i7 CPU and an NVIDIA GTX 960 Ti GPU, and (2) An embedded NVIDIA TX2 platform.

**Sensitivity Analysis**

Particle filters can achieve increased performance with large number of particles at the cost of increased computational complexity. Conversely too few particles can lead to divergence from the true location due to an inability to represent the true underlying distribution. In order to find the appropriate number of particles that ensure precision while still being computationally feasible we compare the filter performance with various number of particles against a ground truth filter with $N = 16200$. Assuming the underlying distribution represented by the particle set to be a unimodal Gaussian (a valid assumption after convergence), we compute the variance of the KL-Divergence [23] of multiple runs of the filter output with that of the ground truth filter to determine the empirically optimal parameters to be used in our implementation. A low value of the KL-Divergence variance indicates similar performance to the ground truth filter.

We compute the optimal number of particles to be $N = 1068$ based on D3, the dataset with the largest volumetric span, where the KL-Divergence plot has a knee point as shown in Fig. 6.5 . This specific parameter choice is further motivated by implementation constraints.

[1] http://scikit-learn.org/stable/modules/mixture.html

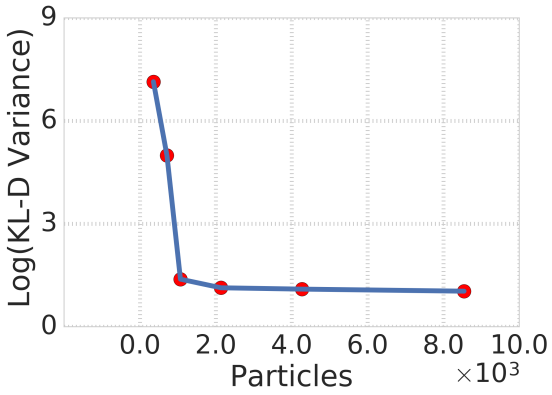



*Figure 6.5: Log of variance of KL-Divergence between the ground truth filter* $(N = 16200)$ *and filters with reduced particle counts. The knee point implies similar performance to the ground truth filter at particles counts* $N > 1000$*. Evaluated on D3.*

### Metric Accuracy Analysis

In this subsection we discuss the localization accuracy of our approach. As mentioned in Sec. 6.2.3 since we do not add new particles when the filter observes particle deprivation and instead randomly reinitialize the particles from the original set, the Root Mean Squared Error (RMSE) of the filter estimate increases when the filter observes particle deprivation. This is highlighted in the plots as vertical shaded regions. For all our evaluations we run the filter 10 times on each dataset and report the average of the mean filter estimate. We do not quantify the sensitivity of the likelihood values to the AHRS pitch and roll estimates as they are accurate enough to not cause any significant difference.

**Evaluation with Ground Truth Datasets (D1, D2):** The objective of using these datasets is to demonstrate the ability of the filter to converge to the ground truth given perfect odometry. We generated a GMM map of the environments using the reconstructed point cloud and used the delta transforms between two consecutive reported sensor poses with added noise as our process model. In all these experiments, we initialized the particles from a uniform distribution over a $4$ m cube and $\pi$ radians yaw orientation around the known initial location. D1(a) and D1(b) contain nominal motion of the sensor, while D2 consists of very aggressive motion in all degrees of freedom.

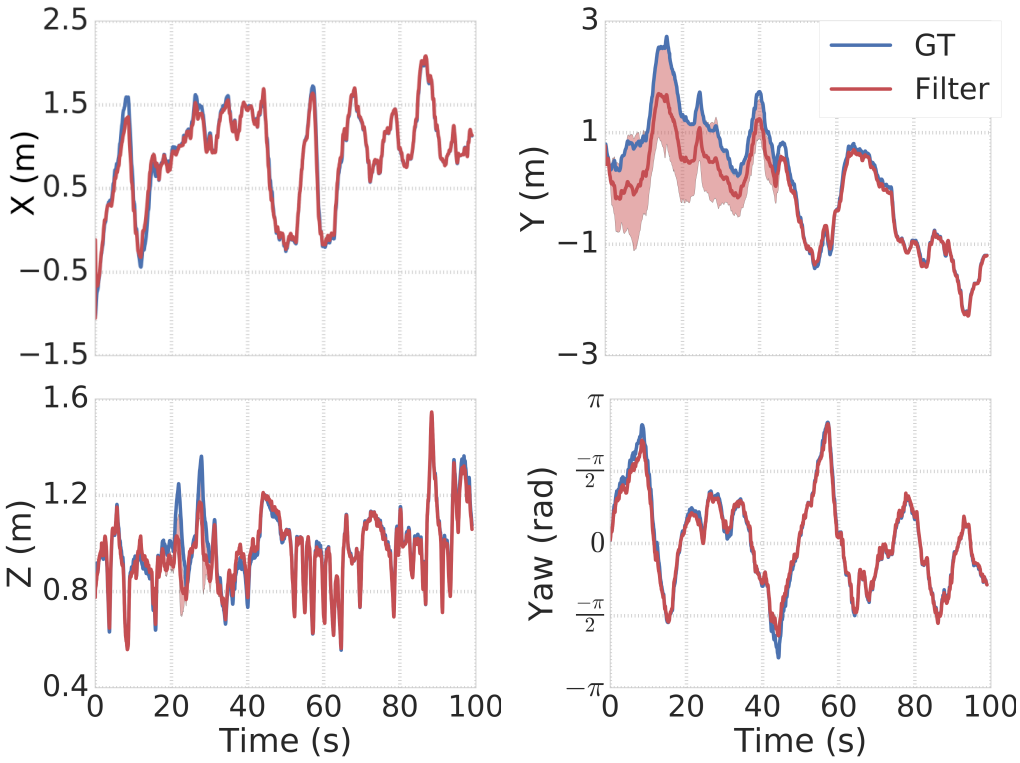



*Figure 6.6: Mean trajectory (red) of the particle filter estimate of 10 trials on the D1(a) dataset compared to the process model trajectory (blue). The shaded region around the mean trajectory shows the variance of the filter estimate over multiple runs. The filter estimates have high variance in the beginning of the trajectories, but soon converge to the correct location and track the ground truth trajectory (blue).*

The filter estimate converged to an incorrect hypothesis for some runs in the initial iterations due to the highly symmetric nature of the environments about the X axis, as can be seen in Fig. 6.6. The RMSE of the filter poses for these datasets is presented in Fig. 6.7.

**Evaluation with Representative Dataset (D3):** The objective of using this dataset is to demonstrate results on a real-world application of the filter. We no longer use ground truth odometry. Additionally, since we don't have a baseline algorithm to directly compare against, we compare the localization performance against ORB-SLAM2 which builds its own succinct map representation. Note that ORB-SLAM2 also utilizes the RGB image data in the dataset whereas we only use the depth. Finally, we also briefly contrast the performance of the filter on the same dataset.

We generate a ground truth point cloud using a FARO Focus 3D Laser scanner [2] and use an

[2] https://www.faro.com/products/construction-bim-cim/faro-focus/

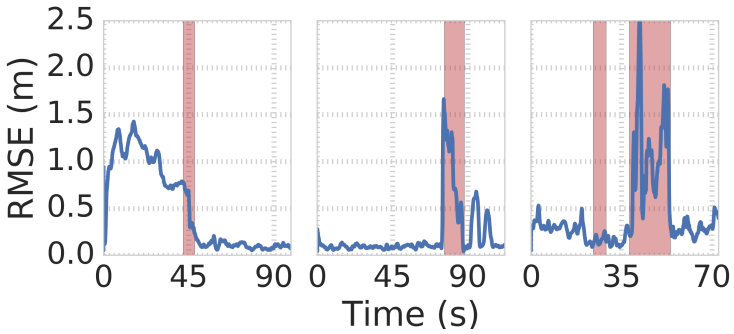



*Figure 6.7: RMSE of 10 trials of the particle filter on the D1(a), D1(b), and D2 datasets respectively. The region in red indicates the time at which the particle filter observes particle deprivation and a consequential RMSE rise.*

ASUS Xtion RGBD camera for acquiring sensor data. An IMU strapped to the camera determines the roll and pitch of the sensor. For odometry, we only use the frame-to-frame relative transform as opposed to the global pose output from ORB-SLAM2 as input to the process model. Note that the global ORB-SLAM2 position we compare to in Fig. 6.8 and Fig. 6.9 is using loop closure to mitigate the drift in its frame-to-frame estimates.

As ground truth is not available for this dataset, we report the negative log-likelihood values at the mean particle filter location and the reported ORB-SLAM2 poses. We show results of two runs in this environment in Fig. 6.8: The first through a nominal path with feature rich data (as shown in detail earlier in Fig. 6.1) where the estimated positions of the sensor for the two approaches are very similar (but with worse likelihood values for ORB-SLAM2). The second run demonstrates the advantage of using particle filters over maximum likelihood estimators in that the former can converge to the correct result even after moving through a region of low observability. We observe that the sensor measurements register at the converged filter location better after snapping back than those for the ORB-SLAM2 estimate, as can be qualitatively seen in Fig. 6.9.

The particles in these experiments are initialized from a uniform distribution over a $4\text{m} \times 8\text{m} \times 3\text{m}$ for position and $\pi$ radians in yaw.

**Evaluation with TUM Dataset (D4):** To demonstrate the ability of our filter to generalize to

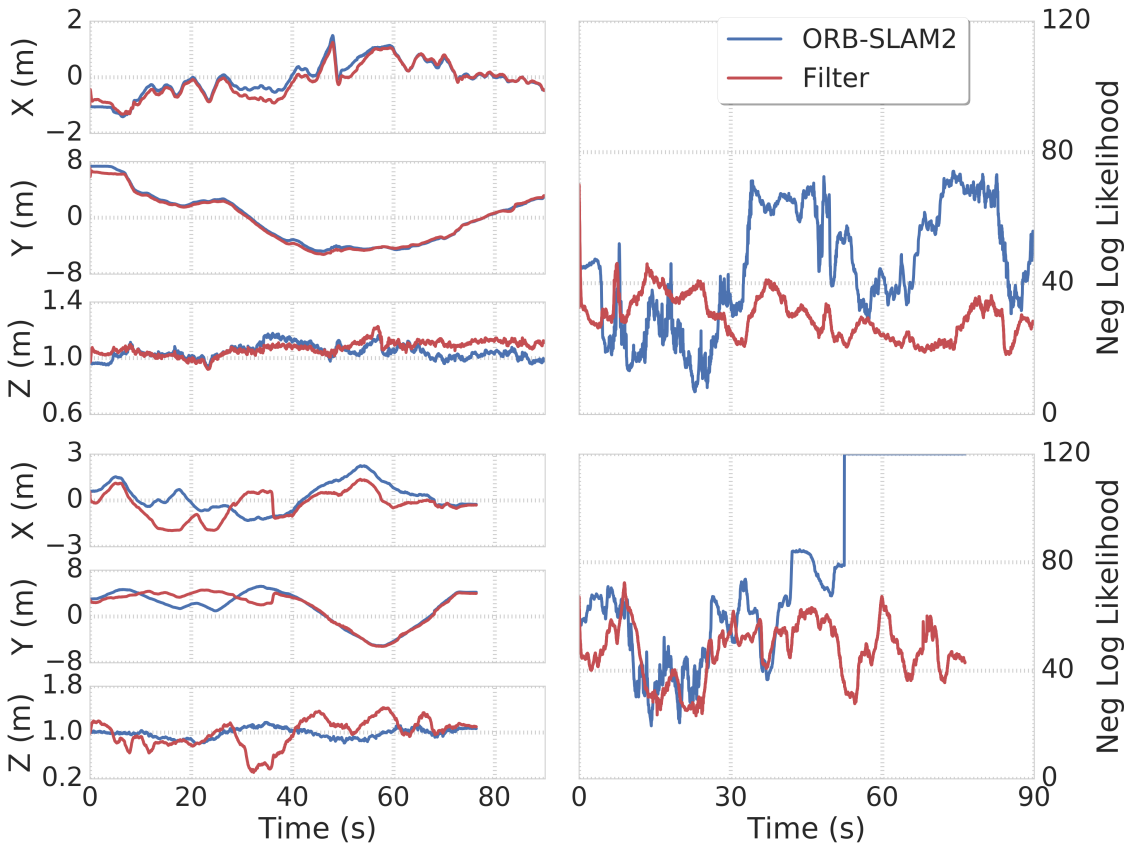



*Figure 6.8: Comparison between the position and corresponding likelihood estimates for two runs from ORB-SLAM2 and our filter, respectively. Top: A nominal path with feature rich data, and Bottom: A path moving through regions of low observability. Contrast the continually increasing divergence (capped in the graph) of the ORB-SLAM2 estimate after moving through the feature poor region with the lower snapped negative likelihood values for the same locations for our filter. The corresponding poses and overlaid depth scan at approximately 55s is shown in Fig. 6.9. Due to a minimal overlap of the depth scan with the map for the ORB-SLAM2 frame, the likelihood value is very low.*

both different odometry algorithms and datasets we compare the performance with three different odometry inputs as process models: The Generalized-ICP algorithm [44], ORB-SLAM2 frame-to-frame relative transform, and ground truth odometry, as shown in Fig. 6.10. The point cloud map of the environment was created by stitching several sensor scans together using their corresponding ground truth poses. In spite of the stitched point cloud not being as well registered as that from a FARO scanner due to sensor and ground truth pose noise, the performance of the filter is similar (Table 6.2).

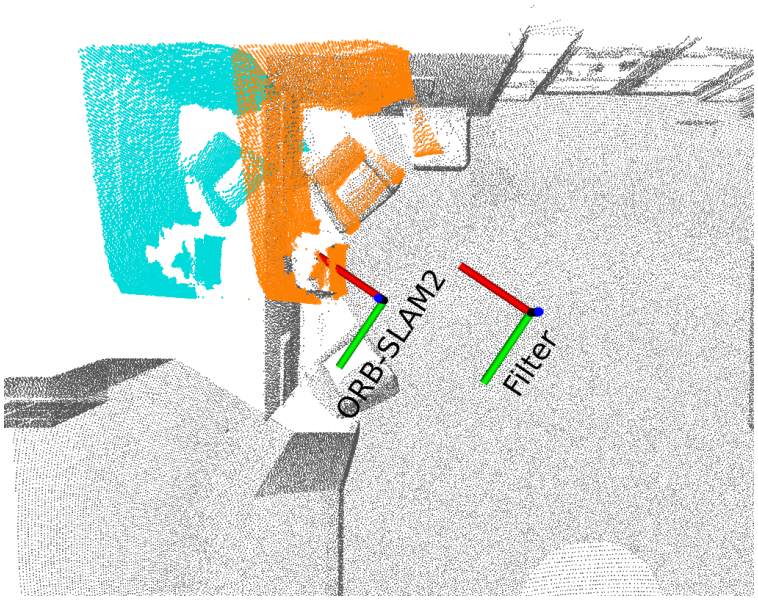



*Figure 6.9: Comparison of registration of current sensor measurement at ground truth point cloud (gray) at ORB-SLAM2 pose estimation (cyan) and at the estimated filter pose (orange). The sensor measurement aligns with the ground truth point cloud in the filter estimate frame while the accumulated drift in the ORB-SLAM2 frame due to transition through a less feature rich region leads to poor alignment.*

| Our Approach | | | ORB-SLAM2 |
|---|---|---|---|
| Process Input | mean | var ($cm^2$) | |
| ORB-SLAM2 Velocity | 7.67 | 0.21 | |
| Ground Truth Velocity | 7.56 | 0.28 | 4.55 |
| G-ICP Velocity | 9.07 | 0.21 | |

*Table 6.2: Performance on D4 (RMSE in cm)*

**Runtime Performance Analysis**

As seen in Fig. 6.11, the likelihood evaluation is the most computationally expensive operation. Execution time for this step varies with the number of Gaussian components used to compute likelihood for each image patch and therefore is dependent on the fidelity of the model.

The filter runs at an average rate of 80 Hz and 9.5 Hz on the Desktop and embedded class systems, respectively. This is comparable to the ORB-SLAM2 rates of 47 Hz and 20 Hz on the respective platforms. Initial convergence on the TX2 is slower due to the implicitly larger odometry steps. However, post convergence the metric performance is not significantly affected. As

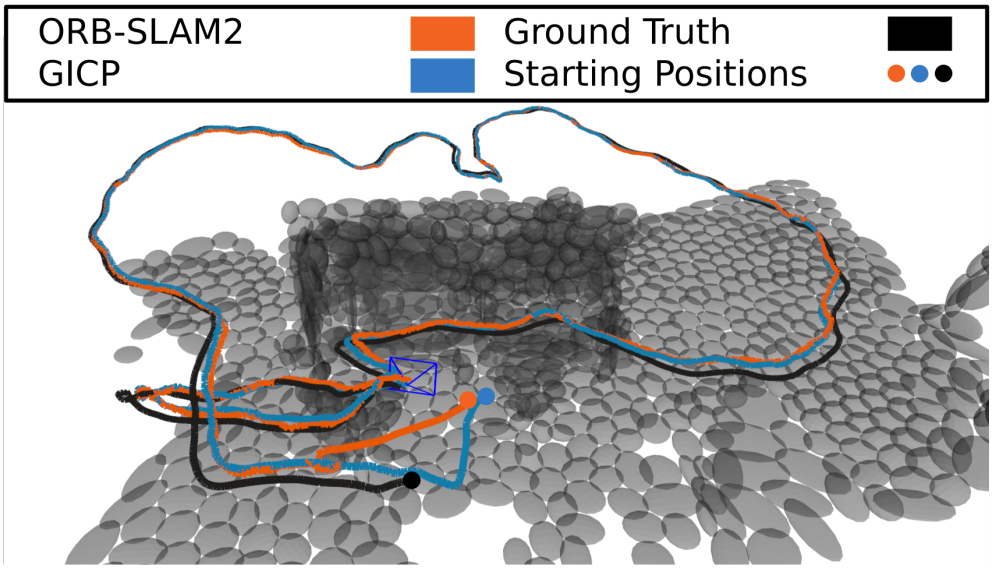



*Figure 6.10: Comparison of our particle filter approach using ORB-SLAM2 frame-to-frame odometry (orange) and Generalized-ICP (cyan) as process models with ground truth pose (black) on TUM's Freiburg 3 Desk Dataset. The GMM representation of the world is created by stitching sensor scans using the ground truth pose estimates. The higher global error of our approach than that of ORB-SLAM2 can be attributed to the noisy reconstruction of the environment point cloud from the accumulated scans.*

an illustrative example, the impact of the slower runtime performance on the TX2 for D1(a) is demonstrated in Fig. 6.12.

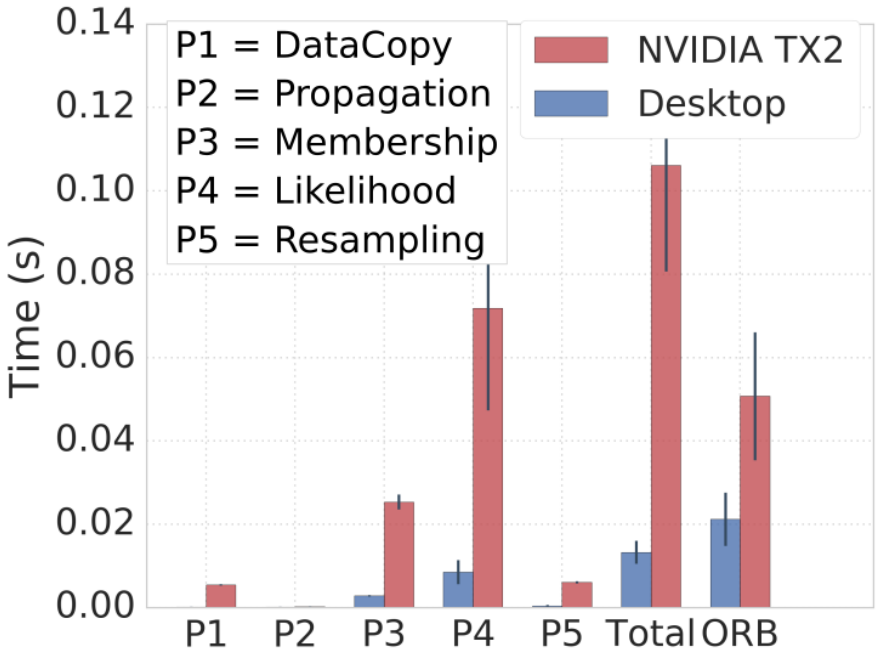



*Figure 6.11: Execution time comparison for subcomponents of the algorithm for the D1(a) dataset on an Intel i7 desktop with an NVIDIA GPU and an embedded NVIDIA TX2 platform. Performance scales linearly with the number of CUDA cores. As a point of comparison ORB-SLAM2 runtime on the same dataset is faster on the embedded platform than on the desktop.*

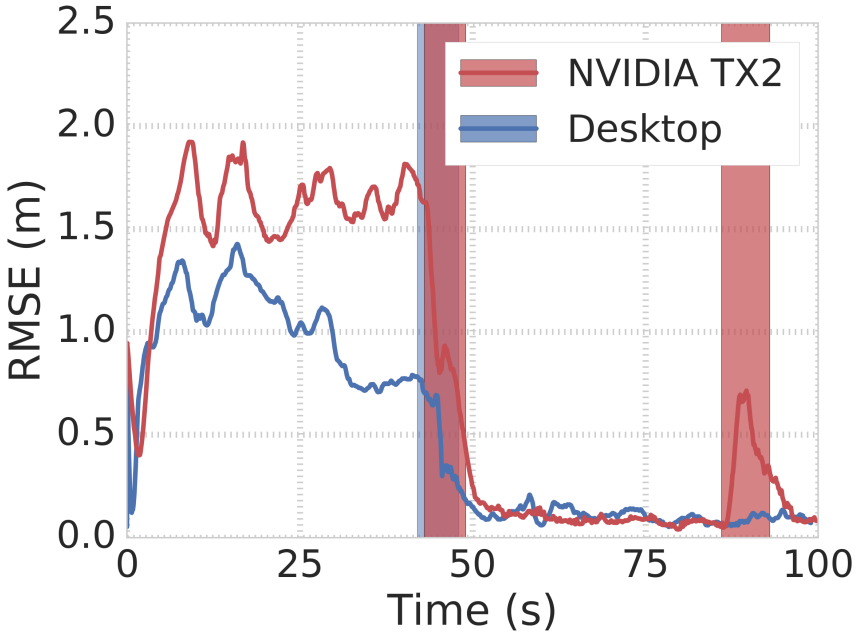



*Figure 6.12: Comparison of the filter performance on the desktop with the NVIDIA TX2 on the D1(a) dataset. As the filter operates at a slower frame rate on the TX2 it initially exhibits a larger error but once the sensor observes a uniquely identifiable location, both trial sets converge to the ground truth location.*

## 6.3 Collision Avoidance

The geometric properties of the Chi squared probability contours of a Gaussian distribution provides various benefits for algorithms that require general information about the presence or absence of matter in specific locations in the world. As described in Sec. 4.1.3, the Chi squared probability contour of a Gaussian distribution is represented by an ellipsoid in $\mathbb{R}^3$ space. We exploit this geometric property of the proposed map representation to create a reactive, online collision checking approach that enables fast obstacle avoidance in unknown, cluttered environments.

The contributions of this section is to present a novel algorithm for generating local maps using Gaussian distributions and show that the proposed method scales efficiently with incremental sensor measurements. We take a geometric approach for computing collisions given the $4\sigma$-probabilistic bound of each Gaussian component and a trajectory.

We illustrate the proposed algorithm with motion primitive based teleoperation proposed by Yang et al. [60] to show real-time collision avoidance. Experimental results of a quadrotor teleoperated through a cluttered environment with Gaussian distribution based local map results in an average collision checking time of $0.150$ to $0.204$ milliseconds per trajectory, which outperforms collision checks against discrete world representations.

### 6.3.1 Local Map

A reactive collision avoidance strategy does not require the information about the global map that was constructed in Chapter 4. For this application, the robot requires information about only its immediate surroundings within and outside its current FOV. To appropriately address this concern, we deviate from the global mapping strategy proposed previously and use a keyframe based local mapping strategy. Further, we diverge from the active-inactive segmentation approach proposed for global mapping and propose a 3D KD-Tree based segmentation approach for component subsection. These algorithmic changes are motivated by the application. Since a robot moving in an

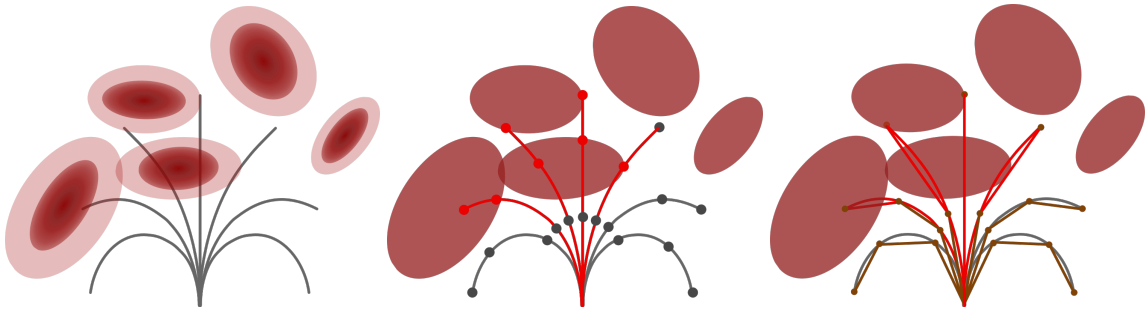


*Figure 6.13: A simplified 2D view of the proposed collision avoidance algorithm, illustrated using forward-arc motion primitives. A set of motion primitives interacting with a local Gaussian distribution based map (in burgundy) with configuration space inflation (light burgundy) is shown in (a). Each Gaussian component is reduced to its $4\sigma$ geometric representation for collision checking, via (b) sampling points along trajectories or (c) creating linear approximations to the trajectory based on curvature and solving for ellipsoid-line intersections. Rejected trajectories are shown in red.*

unknown environment may execute trajectories outside its current FOV, the local active map must be aware of the structure information outside the FOV. The proposed KD-Tree based segmentation approach enables the algorithm to sub-select all the Gaussian distributions within a particular radius of the robot and perform collision avoidance with respect to these components.

The local mapping framework generates a spatially consistent local map and a active local map for collision checking. Since dynamically feasible trajectories often extend past the current FOV of the sensor, it is necessary to create a local map that encloses the vehicle using all recent sensor observations. To achieve this, we dynamically select keyframes and integrate subsequent sensor measurements that provide novel information about the environment to these keyframes. A new map is initialized when a keyframe is observed. This allows us to create spatially consistent maps with minimal information redundancy.

Given a history of sensor measurements as well as their corresponding state estimates, we classify the current sensor frame as a keyframe *KF*, subframe *SF* or a bufferframe *BF*. Novel information is extracted from the current sensor data, to which we fit a hierarchical map. For collision checking, we only use ${}^{w}\Theta^{2}$ that represents the highest hierarchical model at level 2. The active local map contains the Gaussian distributions that represent the vehicle's immediate

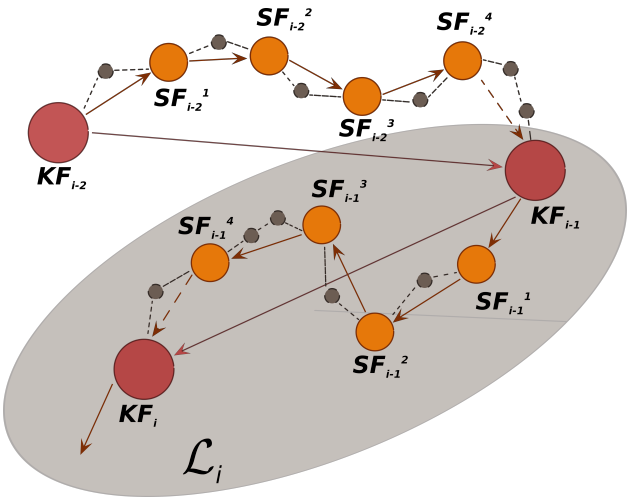



*Figure 6.14: A graphical representation of a local map $\mathcal{L}_i$ . The vehicle poses are classified as KF (red), SF (orange) or a BF (brown) based on the euclidean distance between them. Each SF registers to a KF and a BF is used to be able to represent dynamic obstacles.*

surroundings extracted from the current map ${}^{w}\mathbf{\Theta}^{2}$.

### Frame Classification

Each incoming sensor frame is classified as either a

- *KF*: An anchor frame to which all the subsequent *SFs* and *BFs* in the local map $\mathcal{L}_i$ are registered to,
- *SF*: Sensor frames that provide sufficient novel information unobserved in the current map local map $\mathcal{L}_i$,
- *BF*: Sensor frames that are stored only for a single time step as to accommodate for dynamic obstacles; but do not get stored in $\mathcal{L}_i$

Given the current sensor location in the world frame ${}^{w}\mathbf{T}_{c}$, the latest *KF* location ${}^{w}\mathbf{T}_{k}$, and the latest *SF* location ${}^{w}\mathbf{T}_{s}$, the current frame ${}^{w}\mathcal{F}_{c}$ is classified according to a set of Euclidean distance

thresholds, $\alpha_k$, $\alpha_s$ and $\beta_s$, as

$$
{}^{w}\mathcal{F}_c = \begin{Bmatrix} \textit{KF}, & \text{if } \|\mathbf{t}\left(\nabla\{{}^{w}\mathbf{T}_k, {}^{w}\mathbf{T}_c\}\right)\| \geq \alpha_k \\ \textit{SF}, & \text{if } \|\mathbf{t}\left(\nabla\{{}^{w}\mathbf{T}_s, {}^{w}\mathbf{T}_c\}\right)\| \geq \alpha_s \\ & \text{or } \|\mathbf{R}\left(\nabla\{{}^{w}\mathbf{T}_s, {}^{w}\mathbf{T}_c\}\right)\| \geq \beta_s \\ \textit{BF}, & \text{otherwise} \end{Bmatrix} \tag{6.5}
$$

where $\mathbf{t}(\nabla\{\mathbf{T}_1, \mathbf{T}_2\})$ is the translation between transforms $\mathbf{T}_1$ and $\mathbf{T}_2$, and $\mathbf{R}(\nabla\{\mathbf{T}_1, \mathbf{T}_2\})$ is the change in the heading of the vehicle between transforms $\mathbf{T}_1$ and $\mathbf{T}_2$.

**Local map fusion**

A new local map $\mathcal{L}_i$ is constructed when a new *KF* is spawned. $\mathcal{L}_i$ consists of the GMM components fused to the previous *KF* and the latest *KF*. The local map only contains Gaussian components learned from a *KF* or a *SF*. GMM components learned from a *BF* are only stored for the current time step in order to account for dynamic obstacles.

$$
\mathcal{L}_i = \left\{ {}^{i}\boldsymbol{\theta}^2_{1,i-1}, {}^{i}\boldsymbol{\theta}^2_{2,i-1}, .., {}^{i}\boldsymbol{\theta}^2_{j,i-1}, .. \right\} \bigcup \left\{ {}^{i}\boldsymbol{\theta}^i_1, {}^{i}\boldsymbol{\theta}^i_2, .., {}^{i}\boldsymbol{\theta}^i_j, .. \right\} \tag{6.6}
$$

where ${}^{i}\boldsymbol{\theta}_{j,i-1}$ represents the $j^{\text{th}}$ *SF* Gaussian component in the $(i\text{–}1)^{\text{st}}$ *KF* transformed in the $i^{\text{th}}$ *KF* and ${}^{i}\boldsymbol{\theta}^i_j$ represents the $j^{\text{th}}$ *SF* Gaussian component learned in the $i^{\text{th}}$ *KF*. A graphical representation of this process is shown in Fig. 6.14.

**Active Map Segmentation**

As the local map $\mathcal{L}_i$ may be spatially expansive depending upon the maximum range of the sensor and the velocity of the vehicle, we further reduce the size of the map used for collision checking by segmenting the map as active and inactive local map. We store the means $\boldsymbol{\mu}_i$ of the Gaussian components in the local map in a KD-Tree, and query an $\epsilon$-ball around the current pose of the

vehicle. The query radius $\epsilon$ is determined using the maximum trajectory distance. The Gaussian components $\boldsymbol{\theta}_j$, in the local map $\mathcal{L}_i$ that lie inside the $\boldsymbol{\epsilon}$-ball forms the active local map; i.e., $\mathcal{AL}_i = \{\boldsymbol{\Theta}_j \,:\, |\boldsymbol{\Theta}_j| < \epsilon,\, \boldsymbol{\Theta}_j \in \mathcal{L}_i\}$. This active map $\mathcal{AL}$ is then used to sub-select trajectories that do not collide with the local map and enable safe traversal of a robot via unknown environments.

### 6.3.2 Trajectory Pruning

We propose two ways of computing collisions given a time-parameterized trajectory and a Gaussian distribution based local map. The two proposed ways are graphically illustrated in Fig. 6.13 The ellipsoidal representation of a Gaussian distribution ${}^{w}\boldsymbol{\theta}_i$ in a coordinate frame rotated along the eigenvectors of its covariance can be represented as (see Sec. 4.1.4):

$$f(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C} (\mathbf{x} - \boldsymbol{\mu}) - 1, \tag{6.7}$$

where $\boldsymbol{\mu}$ is the center of the ellipsoid, $\mathbf{C} = \text{Diag}(c_1^{-2}, c_2^{-2}, c_3^{-2})$ and $c_i$ are the major axes of the ellipsoid. We represent the configuration space of the vehicle by a sphere with radius $r$ centered at the robot's geometric center. Then, we transform the local map to incorporate the configuration space by inflating the major axes:

$$f(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{D} (\mathbf{x} - \boldsymbol{\mu}) - 1, \tag{6.8}$$

where $\mathbf{D} = \text{diag}\left((c_1 + r)^{-2}, (c_2 + r)^{-2}, (c_3 + r)^{-2}\right)$. For sufficiency, we take the ellipsoid defined by the $4\boldsymbol{\sigma}$ probability bound of each Gaussian component, which provides approximately $99.95\%$ Chi squared probabilistic coverage of the underlying point density.

Suppose a trajectory is given by $\mathbf{x}(t) = \gamma(t)$, where $\gamma(t)$ is a time-parameterized function with $t$ defined over some interval $t \in [t_0,\, t_f]$, and $\mathbf{x}(t) = [x(t), y(t), z(t)]^T$. Then, the ellipsoid-trajectory

equation becomes:

$$f(t) = f(\mathbf{x}(t)) = (\mathbf{x}(t) - \boldsymbol{\mu})^T \mathbf{R}^T \mathbf{D} \mathbf{R} (\mathbf{x}(t) - \boldsymbol{\mu}) - 1 \tag{6.9}$$

$$f(t) = (\mathbf{x}(t) - \boldsymbol{\mu})^T \mathbf{A} (\mathbf{x}(t) - \boldsymbol{\mu}) - 1, \tag{6.10}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3} \in \mathbb{SO}(3)$ is the rotation matrix to transform the local trajectory into the frame of the eigenvectors of the covariance of the Gaussian distribution. An intersection or collision occurs when $f(t) \leq 0$.

For arbitrary trajectories $\mathbf{x}(t)$, no analytic solutions exist to Eq. 6.10 unless $\mathbf{x}(t)$ is affine. In the following subsections, we present two algorithms for collision checking for arbitrarily complex trajectories and provide a brief discussion on computational complexities.

### Sampling based collision checking

Instead of computing an analytic solution to Eq. 6.10, a simple check would be to sample points along each trajectory. For $M$ Gaussian distributions, $N$ local trajectories, and $S$ samples per trajectory, the computational complexity would be $\mathcal{O}(MNS)$. This approach is delineated in Algorithm 6.1.

**Algorithm 6.1** Collision Checking with Gaussian distribution based Local Map via Sampling

**Given** $M$ Gaussian components, $N$ local trajectories, $S$ samples per trajectory
Discretize time interval $[t_0,\, t_f]$ into $\mathbf{t} = \{t_i\},\ i = 1, \ldots, S$ s.t. $t_i \in [t_0,\, t_f]$
**for** $n = 1 : N$ trajectories **do**
    **for** $m = 1 : M$ Gaussian distributions ${}^w\boldsymbol{\theta}$ **do**
        Obtain the eigenvector matrix $\mathbf{R}_m$, centers $\boldsymbol{\mu}_m$
        compute $\mathbf{A}_m = \mathbf{R}_m^T \mathbf{D}_m \mathbf{R}_m$
        where $\mathbf{D}_m = \text{Diag}\left((c_{m1} + r)^{-2}, (c_{m2} + r)^{-2}, (c_{m3} + r)^{-2}\right)$
        **for** each $t \in \mathbf{t}$ **do**
            Query point at time $t$: $\mathbf{x}_s = \mathbf{x}(t)$
            **if** $f(\mathbf{x}_s) = (\mathbf{x}_s - \boldsymbol{\mu}_m)^T \mathbf{A}_m (\mathbf{x}_s - \boldsymbol{\mu}_m) \leq 1$ **then**
                Reject trajectory and increment

**Piecewise affine trajectory approximation**

If the trajectory is sufficiently smooth, one can generate piecewise affine approximations (PWA) to the trajectory using heuristics. For each trajectory, suppose $s$ segments of affine approximations sufficiently approximate the trajectory. Then, over each segment, the affine approximation $\mathbf{x}_s(t) = \mathbf{a}_s t + \mathbf{b}_s$ with $\mathbf{a}_s, \mathbf{b}_s \in \mathbb{R}^3$ and $t \in [t_{s-1},\ t_s]$ can be analytically solved in the frame of each Gaussian distribution.

The ellipsoid-line equation using Eq. 6.8, in the frame of the Gaussian component, can be written as:

$$f(\mathbf{x}_s) = (\mathbf{x}_s)^T \mathbf{D} (\mathbf{x}_s) - 1 \tag{6.11}$$

$$f(t) = (\mathbf{a}_s t + \mathbf{b}_s)^T \mathbf{D} (\mathbf{a}_s t + \mathbf{b}_s) - 1, \tag{6.12}$$

Without loss of generality, a trajectory can always be transformed into the frame of the mixture component such that $\mathbf{D}$ is diagonal. Collisions are found via solutions to

$$\begin{aligned} 0 = \left( \frac{a_1^2}{c_1^2} + \frac{a_2^2}{c_2^2} + \frac{a_3^2}{c_3^2} \right) t^2 + 2 \left( \frac{a_1 b_1}{c_1^2} + \frac{a_2 b_2}{c_2^2} + \frac{a_3 b_3}{c_3^2} \right) t \\ + \left( \frac{b_1^2}{c_1^2} + \frac{b_2^2}{c_2^2} + \frac{b_3^2}{c_3^2} - 1 \right) \end{aligned} \tag{6.13}$$

With the assumption that the trajectory does not begin inside a Gaussian component.

For $M$ distributions, $N$ trajectories, the number of segments is dependent on the curvature of the trajectory. The computation complexity would be $\mathcal{O}(MNS_n)$, where $S_n \leq S$, $n = 1, \ldots, N$ such that the worst case complexity collapses to that of the sample based approach (with $S$ samples per trajectory). This approach is delineated in Algorithm 6.2. We provide a heuristic for determining number of segments for motion primitives in Sec. 6.3.3.

**Algorithm 6.2** Collision Checking with Gaussian distribution based Local Map via PWA Trajectory Approximation

**Given** $M$ Gaussian components, $N$ local trajectories
**for** $n = 1 : N$ trajectories **do**
    Heuristically discretize trajectory into $S_n$ segments
    Compute $S_n$ affine approximations
    **for** $m = 1 : M$ Gaussian components **do**
        Obtain the eigenvector matrix $R_m$, centers $\boldsymbol{\mu}_m$, and transform $\mathbf{x}$ into the frame of the Gaussian distribution
        **for** each $s = 1 : S_n$ **do**
            Solve Eq. 6.13 and denote solutions as $t^*_{1,2}$
            **if** $t^*_{1,2} \in [t_{s-1}, t_s]$ **then**
                Reject trajectory and increment

### 6.3.3 Local Trajectories: Motion Primitive Library

An example of a family of local trajectories is a motion primitive library. Motion primitives are dynamically feasible local trajectories parameterized by the input space of the dynamics, which have been shown to be amenable to online autonomous exploration [53] and teleoperation [60, 61]. This paper follows [60] and uses *forward-arc motion primitives*. These local trajectories are formed by propagating the dynamics of a unicycle model with a constant linear velocity $v_x$, angular velocity $\omega$, and vertical velocity $v_z$ for a specified amount of time, $T$ [40]. We direct the readers towards the *forward-arc motion primitives* proposed by Yang et al. [60] for a detailed description. In this thesis, we focus on the impact of the proposed map representation on collision avoidance using these *forward-arc motion primitives*.

### 6.3.4 Evaluation

We evaluate our proposed algorithm in simulation in a cluttered environment as shown in Fig. 6.15 and compare it to KD-Tree maps in our local mapping framework approach, in terms of computational complexity of map generation and collision checking, and also show that our method provides safety guarantee of at least the configuration bound."

In the simulation scenario, an operator teleoperates the vehicle using forward-arc motion prim-

[2]Available at: https://github.com/vibhavg/simulation_environments

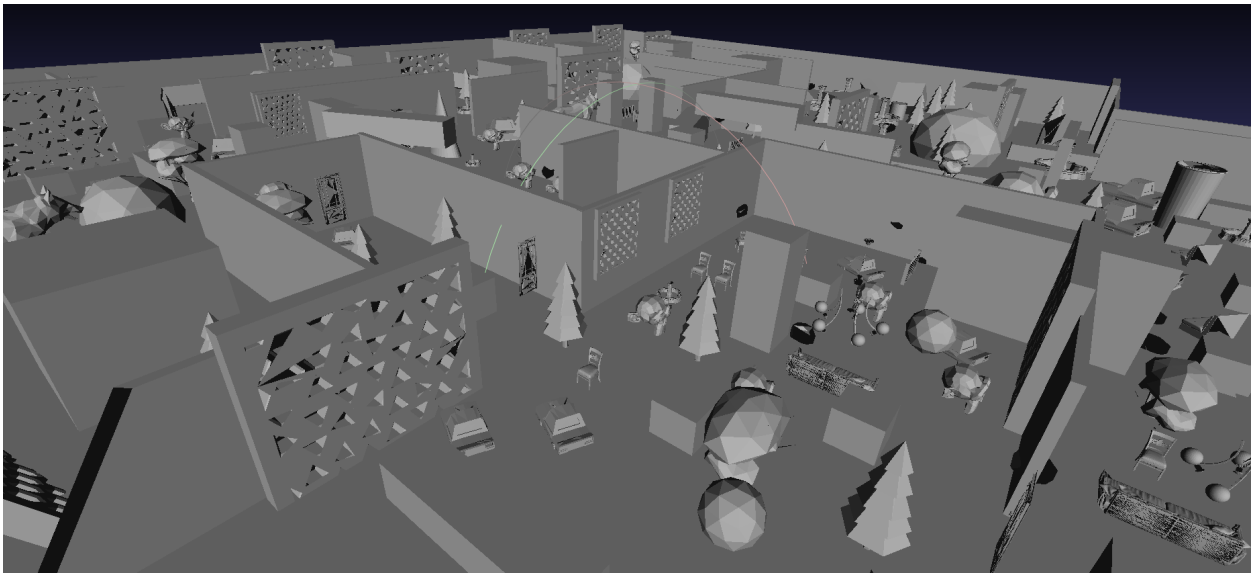


*Figure 6.15: The cluttered environment used to evaluate our collision avoidance strategy.*

itives [60]. We generate a library of 155 motion primitives, using 31 linearly spaced angular velocities $\omega \in \{-3, 3\}$ rad/s, 5 linearly spaced vertical velocities $v_z \in \{-1, 1\}$ m/s, and limit the vehicle to a maximum linear velocity of 2 m/s. We assume a configuration radius of 0.5 m. The heuristics for frame categorization are: $\alpha_k$ = 1.0 m, $\alpha_s$ = 0.2 m, and $\beta_s$ = 0.2 radians.

**Safety**

Safety is evaluated using the minimum distance of the vehicle center to its surroundings. We use a dense point cloud representation as a baseline map and query the radius of free space around the vehicle at each iteration. In Fig. 6.16, two 6-8 minute example trials are shown. Throughout each trial, the vehicle's configuration space, denoted in blue, is contained within the free space around the vehicle, denoted in grey, indicating that the vehicle is safe at all times.

**Efficiency**

Collision checking timing analysis averaged over 10000 trajectories is shown in Fig. 6.17. We observe 0.737 ms for KD-Tree based collision check per trajectory, 0.204 ms for Gaussian distribution based local map with sampling-based collision check per trajectory, and 0.150ms for Gaussian distribution based local map with PWA trajectory approximation. Sufficiently representing the local

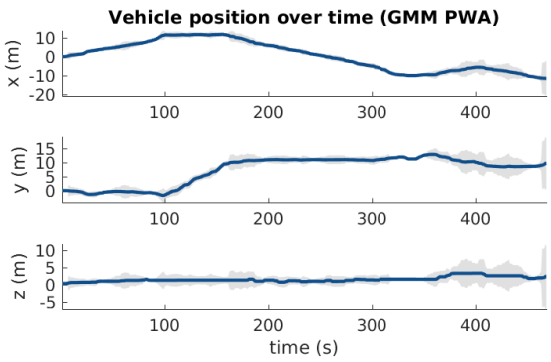



*Figure 6.16: A visualization of free space around each vehicle vs. configuration space for example trials with (a) Gaussian distribution based local map with sampling, and (b) Gaussian distribution based local map with PWA approximations based collision avoidance. The blue line denotes the pose of the vehicle and the light blue shading denotes the configuration space of* $0.5$ *m.*

map using a low number of components contributes to significant speed-ups over KD-Tree queries.

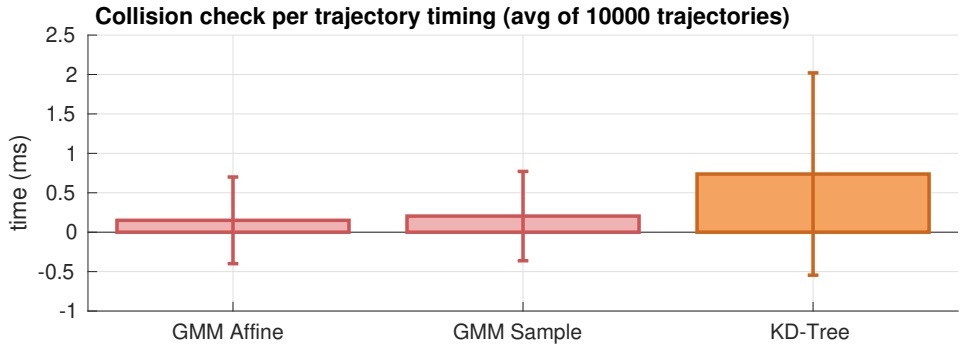



*Figure 6.17: Timing analysis for per trajectory collision checking with samples and PWA trajectory approximations, as compared to using KD-Tree representations. Gaussian distribution local map based methods take* $0.25$ *ms for collision checking per trajectory, whereas KD-Tree takes* $0.75$ *ms per trajectory. Error bars report standard deviation of the mean.*

## 6.4 Summary

In this section, We presented a framework to perform real-time global localization of depth sensors and another framework to perform collision avoidance using the lowest fidelity map representation (Chapter 4). Key to being able to do this is the ability to reinterpret a Gaussian distribution as an ellipsoid in 3D. The projection of an ellipsoid on to the image plane of the sensor to evaluate the

likelihood of the data for a given pose enables us to efficiently compute the likelihood of a particle being the true estimate of the sensor location. By utilizing a fast likelihood computation approximation we can then perform robust particle filter localization in real-time even on an embedded GPU platform. By reinterpreting a Gaussian distribution as strictly a geometric entity, we are able to efficiently subselect a set of trajectories that are safe, and do not collide with the obstacles in the environment. This enables a robot to navigate safely through an unknown environment in real-time using a succinct map representation.

# Chapter 7

# Conclusion

A complex mobile robot system should be able to efficiently process large scale input sensor data and operate in response to it. Size Weight and Power constraints on mobile robots impose restrictions on the available computational and memory resources. The choice of data representation utilized by various perceptual pipelines is of paramount importance for enabling real-time operation of a SWaP constrained system. Often a disconnected system using various independent map representations and data pre-processing pipelines adds additional burden on the available resources and impacts the accuracy and real-time performance of the robot.

This thesis presents a computationally feasible solution to perform SLAM on platforms with computational and memory limitations. A hierarchical Gaussian distribution based map representation is proposed that represents the map information at different fidelities. Additionally, a localization approach is proposed that exploits the hierarchical map representation to enable dense real-time SLAM with low computational requirements.

We first present a high-fidelity 3D reconstruction pipeline using a generative map representation in Chapter 4. Fitting a generative model such as a Gaussian Mixture Model (GMM), [16] is computationally more expensive than commonly used map representations such as voxel grids or surfels [58]. We presented a model fitting technique that, in contrast to the state-of-the-art genera-

tive model based mapping techniques, maximizes the 3D reconstruction accuracy of the map using succinct Gaussian distributions as our underlying map representation. The projective constraints enforced by a commonly available depth sensor enable us to impose a structure on the model fitting algorithm and make the algorithm computationally efficient and feasible. Quantitative and qualitative performance comparison of our mapping approach to state-of-the-art mapping approaches demonstrate superior performance in terms of the reconstruction accuracy and the memory complexity of the map representation. A hierarchical map reconstruction technique is proposed that represents the map at multiple fidelities, reducing the memory complexity of the representation at higher hierarchical levels and improving the reconstruction accuracy at lower hierarchical levels.

Second, we proposed a frame-to-model localization technique that exploits the hierarchical map representation to track a live camera in a global map robustly and accurately. The hierarchical structure of the map representation enables our proposed localization pipeline to utilize adequately informative map for different tasks, such as scan alignment, correspondence computation and active-inactive map segmentation. Quantitative analysis of the proposed SLAM approach demonstrates that our approach outperforms state-of-the-art SLAM approaches and performs robustly and reliably in challenging environments where state-of-the-art approaches fail.

Finally, we discuss some applications of the proposed map representation. We showed that given a global map of the world represented as a GMM, we can use the geometric interpretation of a Gaussian distribution in $3$D to implement a computationally efficient multi-hypothesis particle filter based localization framework on a SWaP constrained system. The compact nature of Gaussian distributions enables us to store adequate information about the structure in the scene that enables a particle filter to operate efficiently in large scale environments. Further, we proposed a safe navigation and collision avoidance technique that exploits the ellipsoidal geometry of a fixed confidence bound of Gaussian distributions. Large structural geometry is efficiently represented as Gaussian distributions and a local teleoperation pipeline uses these Gaussian distributions to

search for safe collision free trajectories in unknown environments.

Overall, we propose the utility of Gaussian distributions as 3D structure primitives to enable real-time autonomy on SWaP constrained systems. Specifically, in this thesis, we demonstrated a SLAM framework that uses hierarchical Gaussian distribution based map representation that is more robust, accurate and memory efficient than state-of-the-art frameworks. Additionally, we demonstrated the applicability of the proposed map representation for common robotics problems such as global localization for kidnapped robots and efficient collision avoidance in unknown environments.

## 7.1 Future Work

The proposed work establishes groundwork for enabling a unified autonomy system that can operate in real-time, generating high fidelity 3D reconstruction of the world while safely exploring or inspecting an unknown environment. Some avenues of future work that could enable this vision include the following:

- Real-time implementation on a SWaP constrained system: The proposed mapping and localization frameworks are highly parallelizable by design. A GPU implementation of the framework would enable a majority of the independent processes that operate sequentially to execute in parallel thus reducing the run-time of the proposed SLAM pipeline.
- Incorporation of loop closures: An accurate frame-to-model localization framework, accumulates drift over time. Enabling an optimization framework in the backend, that can detect loop closures when the sensor revisits an observed section of the map, using either a factor graph like approach or a more suitable deformation graph based approach would enable the pose estimates to recover from large drifts and create more accurate reconstruction of the world.
- Extension to model occupancy: A large variety of active perception algorithms utilize the

information about occupancy around the robot safely navigate in unknown environments and explore environments. The current map representation only stores information about the occupied spaces in the scene. An elegant solution using hierarchical Gaussian distributions to represent free spaces in the world, would enable active perception algorithms to operate on the same map representation that is used for high fidelity 3D reconstruction and mapping, thus reducing the memory and computational redundancy in the data processing pipeline.

# Bibliography


[1] Hirotugu Akaike. A new look at the statistical model identification. In *Selected Papers of Hirotugu Akaike*, pages 215–222. Springer, 1974.

[2] Hatem Alismail, Brett Browning, and Simon Lucey. Direct visual odometry using bit-planes. *arXiv preprint arXiv:1604.00990*, 2016.

[3] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008.

[4] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[5] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[6] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 2019.

[7] Pinyuen Chen, T-J Wu, and J Yang. A comparative study of model selection criteria for the number of signals. *IET Radar, Sonar & Navigation*, 2(3):180–188, 2008.

[8] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015.

[9] Per Christensen. Point-based approximate color bleeding. *Pixar Technical Notes*, 2(5):6, 2008.

[10] Brian Curless and Marc 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, 1996.

[11] Aditya Dhawale and Nathan Michael. Efficient parametric multi-fidelity surface mapping [To Appear]. In *Robotics: Science and Systems*, volume 16, 2020.

[12] Aditya Dhawale, Kumar Shaurya Shankar, and Nathan Michael. Fast Monte-Carlo localization on aerial vehicles using approximate continuous belief representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5851–5859, 2018.

[13] Aditya Dhawale, Xuning Yang, and Nathan Michael. Reactive collision avoidance using real-time local gaussian mixture model maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3545–3550. IEEE, 2018.

[14] Aditya Dhawale, Kumar Shaurya Shankar, and Nathan Michael. Hierarchical Gaussian distributions for real-time SLAM [Submitted]. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[15] Ethan Eade. Lie groups for 2D and 3D transformations. *URL http://ethaneade. com/lie. pdf, revised Dec*, 2013.

[16] Benjamin Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Acceler-

ated generative models for 3D point cloud data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5497–5505, 2016.

[17] Raúl San José Estépar, Anders Brun, and Carl-Fredrik Westin. Robust generalized total least squares iterative closest point registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2004.

[18] Kshitij Goel, Micah Corah, Curtis Boirum, and Nathan Michael. Fast exploration using multirotors: Analysis, planning, and experimentation. *The Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-19-03*, 2019.

[19] Georg Halmetschlager-Funek, Markus Suchi, Martin Kampel, and Markus Vincze. An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments. *IEEE Robotics & Automation Magazine*, 26(1):67–77, 2018.

[20] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531. IEEE, 2014.

[21] Mark H Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[22] Joao F Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8476–8484, 2018.

[23] John R Hershey and Peder A Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, volume 4, 2007.

[24] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard.

OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[25] Du Q Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[26] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[27] Bing Jian and Baba C Vemuri. Robust point set registration using Gaussian Mixture Models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645, 2010.

[28] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[29] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013.

[30] Christine Keribin. Consistent estimation of the order of mixture models. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 49–66, 2000.

[31] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] Genshiro Kitagawa. Monte-Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1), 1996.

[33] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proceed-*

*ings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[34] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 116–121. IEEE, 1985.

[35] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5), 2017.

[36] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*, pages 524–530. IEEE, 2012.

[37] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. IEEE, 2017.

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[39] KB Petersen, MS Pedersen, et al. The Matrix Cookbook, vol. 7. *Technical University of Denmark*, 15, 2008.

[40] Mihail Pivtoraiko, Issa AD Nesnas, and Alonzo Kelly. Autonomous robot navigation using advanced motion primitives. In *Proc. of the IEEE Aerospace Conf.*, pages 1–7, Big Sky, USA, 2009.

[41] Jann Poppinga, Narunas Vaskevicius, Andreas Birk, and Kaustubh Pathak. Fast plane detection and polygonalization in noisy 3D range images. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3378–3383. IEEE, 2008.

[42] Pedro F Proença and Yang Gao. Probabilistic RGB-D Odometry based on Points, Lines and Planes Under Depth Uncertainty. *arXiv preprint arXiv:1706.04034*, 2017.

[43] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 134–144, 2019.

[44] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[45] Jacopo Serafin and Giorgio Grisetti. Using extended measurements and scene merging for efficient and robust point cloud registration. *Robotics and Autonomous Systems*, 92:91–106, 2017.

[46] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus H Gross. GPU-based ray-casting of quadratic surfaces. In *SPBG*, pages 59–65. Citeseer, 2006.

[47] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[48] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019.

[49] Alex Spitzer, Xuning Yang, John Yao, Aditya Dhawale, Kshitij Goel, Mosam Dabhi, Matt Collins, Curtis Boirum, and Nathan Michael. Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor. In *International Symposium on Experimental Robotics*, pages 524–535. Springer, 2018.

[50] Shobhit Srivastava and Nathan Michael. Approximate continuous belief distributions for

precise autonomous inspection. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 74–80. IEEE, 2016.

[51] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The International Journal of Robotics Research*, 31(12):1377–1393, 2012.

[52] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[53] Wennie Tabib, Micah Corah, Nathan Michael, and Red Whittaker. Computationally efficient information-theoretic exploration of pits and caves. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727. IEEE, 2016.

[54] Wennie Tabib, Cormac OMeadhra, and Nathan Michael. On-manifold GMM registration. *IEEE Robotics and Automation Letters*, 3(4):3805–3812, 2018.

[55] Wennie Tabib, Kshitij Goel, John Yao, Mosam Dabhi, Curtis Boirum, and Nathan Michael. Real-time information-theoretic exploration with Gaussian Mixture Model maps. *Proc. Robot.: Sci. and Syst., FreiburgimBreisgau, Germany*, 2019.

[56] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[57] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The unscented particle filter. In *Advances in neural information processing systems*, 2001.

[58] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM without a pose graph. Robotics: Science and Systems, 2015.

[59] Cong-Hua Xie, Jin-Yi Chang, and Yong-Jun Liu. Estimating the number of components in Gaussian mixture models adaptively for medical image. *Optik*, 124(23):6216–6221, 2013.

[60] Xuning Yang, Koushil Sreenath, and Nathan Michael. A framework for efficient teleoperation

via online adaptation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5948–5953. IEEE, 2017.

[61] Xuning Yang, Ayush Agrawal, Koushil Sreenath, and Nathan Michael. Online adaptive teleoperation via motion primitives for mobile robots. *Autonomous Robots*, 43(6):1357–1373, 2019.

[62] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (ToG)*, 32(4):112, 2013.