Localization, Mapping, and Planning in 3D Environments

Nathaniel Fairfield

CMU-RI-TR-05-09

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January, 2009

Thesis Committee:
David Wettergreen, Chair
George Kantor
Tony Stentz
Paul Newman, University of Oxford
Hanumant Singh, Woods Hole Oceanographic Institute

Abstract

Building a map, localizing within the map, and planning using the map are fundamental problems for mobile robotics. Every mobile robotic system must incorporate some type of solution to all three problems.

While the interdependency between mapping and localization is well known as the Simultaneous Localization and Mapping (SLAM) problem, there is a growing understanding in the research community that planning how the robot goes about mapping and exploring an environment (and operating in the environment afterwards) can avoid degenerate conditions and significantly reduce complexity of SLAM. Thus the task of exploring a new environment combines all three problems, since the robot must plan to find actions that reduce uncertainty in both mapping and localization. This combined problem is known as Active SLAM.

Independently, SLAM and planning have been solved in small, two dimensional, structured domains. Robots need to move beyond these simple environments. The challenge is to develop real-time Active SLAM methods that allow robots to explore large, three dimensional, unstructured environments, and allow subsequent operation in these environments over long periods of time.

But scaling up to truly large environments requires a second key insight beyond Active SLAM: to circumvent the scale limitations inherent in SLAM, the world can be divided up into more manageable pieces, or submaps. The SLAM problem then becomes a segmented SLAM problem, which represents the world with a combined metric and topological map, building metric submaps as necessary and refining the topological relationships between submaps.

The contribution of this thesis is a real-time Active SLAM approach that combines a novel evidence grid-based volumetric representation, a robust Rao-Blackwellized particle-filter, a topologically flexible submap segmentation framework, and an integrated stochastic planning method for reducing SLAM uncertainty and predicting possible loop closures based on local map structure. We demonstrate our methods on several robotic platforms in both structured and unstructured large, three dimensional environments.

Contents

Abstract i
Contents ii

1 Introduction **1**
1.1 Approach 3
1.2 Contributions 5
1.3 Organization 5

2 Representing Maps **7**
2.1 Introduction 7
2.2 Related Work in 3D Maps 8
2.3 3D Evidence Grids 9
2.3.1 Updating Evidence Grids 9
2.3.2 Sensor Modeling 10
2.4 The Deferred Reference Counting Octree 11
2.4.1 Reference Counting Octree (RCO) 13
2.4.2 Deferred Reference Counting Octree (DRCO) 13
2.4.3 Compaction 14
2.5 Map Matching with 3D Evidence Grids 15
2.5.1 Related Work in Map Matching 17
2.5.2 Baseline ICP Implementation 18
2.5.3 Evidence Grid Match Score 18
2.5.4 3D Lucas-Kanade Algorithm 19
2.5.5 Region Selection 21
2.5.6 Isodox Surface Matching 21
2.6 Evidence Grid Entropy 23
2.7 Experiments 24
2.7.1 Subregion icLK – Groundhog in Dakota Mine 24
2.7.2 icLK with Heading Error – Cave Crawler in Bruceton Mine 26
2.7.3 Map Experimental Summary 27
2.8 Summary 28

3 Simultaneous Localization and Mapping **29**
3.1 Introduction 29
3.2 Related Work in SLAM 30

3.3 Models . . . 32
3.3.1 Gaussian Models . . . 32
3.3.2 Vehicle Motion Model . . . 33
3.4 SLAM Derivation . . . 34
3.4.1 Bayes Filter . . . 34
3.4.2 Particle Filter . . . 36
3.4.3 Rao-Blackwellized Particle Filter . . . 37
3.4.4 Map representation - 3D Evidence Grids . . . 39
3.4.5 Adaptive particle count . . . 40
3.5 SLAM Entropy . . . 40
3.6 Underwater Robot . . . 42
3.6.1 DEPTHX Vehicle . . . 42
3.6.2 Related Work in Underwater Localization . . . 43
3.7 Experiments . . . 44
3.7.1 Preliminary Characterization – Controlled Test Sites . . . 44
3.7.2 Localization – Zacatón . . . 52
3.7.3 Onboard SLAM – La Pilita . . . 57
3.7.4 SLAM Performance – Caracol . . . 58
3.7.5 SLAM Experimental Summary . . . 59
3.8 Summary . . . 60

4 SLAM with Segmented Maps **62**
4.1 Introduction . . . 62
4.1.1 Overview of SegSLAM . . . 64
4.2 Related Work in Submaps and Large-Scale SLAM . . . 66
4.3 Derivation . . . 68
4.4 Segmentation . . . 71
4.4.1 Motion Error Segmentation . . . 71
4.4.2 Predictive Score Segmentation . . . 72
4.5 Generating Local Metric Map Samples . . . 73
4.6 Matching . . . 76
4.6.1 Winnowing Match Candidates . . . 76
4.6.2 Matching to Candidates . . . 76
4.6.3 Discarding Segments . . . 77
4.7 SegSLAM Entropy . . . 78
4.7.1 Global Metric Maps Via Entropy Minimization . . . 79
4.8 Subterranean Robot . . . 80
4.8.1 Cave Crawler . . . 80
4.9 Experiments . . . 81
4.9.1 Loop Closing Error – CIC Parking Garage . . . 82
4.9.2 Map Goodness – Bruceton Mine . . . 84
4.9.3 Minimum Entropy Map – Gates Building . . . 86
4.9.4 Closing Overview Loops . . . 86
4.10 Statistical accuracy and consistency . . . 91

4.10.1 Consistency Experiment 1 – Ground truth in Bruceton Mine 94
4.10.2 Consistency Experiment 2 – Loop closure in Bruceton Mine 95
4.10.3 SegSLAM Experimental Summary 97
4.11 Summary 98

5 Planning with Segmented Maps **100**
5.1 Introduction 100
5.2 Related Work in Planning 101
5.3 Information Gain-Based Action Selection 103
5.4 Active Localization Using Most Discriminative Actions 104
5.4.1 Justification of the Most Discriminative Action Criteria 104
5.5 Active Localization Experiment – Axial Seamount 107
5.5.1 Baseline Localization Results 108
5.5.2 Active Localization Results 110
5.6 RRT Path Planning in 3D Octree Evidence Maps 114
5.7 Active SLAM with the Segmented Map 117
5.7.1 Prediction with the Segmented Map 118
5.8 Active SLAM Experiment – FRC Catacombs 119
5.8.1 Active Planning Experimental Summary 121
5.9 Summary 123

6 Conclusions **124**
6.1 Contributions 124
6.2 Future Work 125

A Calibration of a Spinning Laser **127**

B Extended Kalman Filter for a 6-DOF IMU **131**

Bibliography **136**

Chapter 1

Introduction

Robots allow computers to interact with the physical world. To perform physical tasks, a robot needs to have a basic set of capabilities. It must be physically capable of doing its job – be that collecting trash or mowing lawns or exploring Mars – which includes mobility and manipulation. It must be able to perceive its environment sufficiently well to accomplish its task – in most cases, the robot must be able to build some sort of map of its environment by making measurements with its sensors, and also to estimate its current location. Finally, it must be able to reason about its own mobility and manipulation together with its perception of the local environment, in such a way that it performs its task efficiently and safely.

The tasks of mapping, localization, and planning, are problems that lie at the core of mobile robotics, and to a large degree they have been solved for small, two dimensional, structured environments. To make robots generally useful in the broader world, they need to move beyond such simple environments into large, three dimensional, and potentially unstructured environments. Accordingly, they need general algorithms for mapping, localizing, planning, and exploring that work just about anywhere. This thesis is concerned with the task of developing broadly applicable methods that will allow robots to explore and operate in most environments, including indoor and outdoor, subterranean and underwater, natural and urban, flat and highly three dimensional.

We make several assumptions about what is involved in a general approach. First, we believe that the approach must be constant in computation (real-time) and linear in storage space, in order to fit within the computational constraints of real mobile robots. Second, we believe that map representations must be fully 3D and capable of representing arbitrary 3D geometry at a level of resolution that is appropriate for the robot and its task. Third, we believe that while the map needs to be locally accurate for path planning and obstacle avoidance, global accuracy can often be relaxed, again depending on the robot and its task. Finally, we believe that the method must not rely on any particular structures or features in the environment. As a result, we believe that the approach must use metric range measurements, and succeed even when these measurements are sparse and inaccurate.

These assumptions were initially motivated by the problem of exploring a large flooded sinkhole with an autonomous underwater vehicle (Figure 1.1). The sinkhole is a natural, irregular formation that had completely unknown structure, except for a crude estimate of the 300 m depth. Without knowing whether to expect a large void or constricted tunnels, a flat bottom or a random pile of rubble, we needed a robust, reliable approach that would allow an autonomous underwater

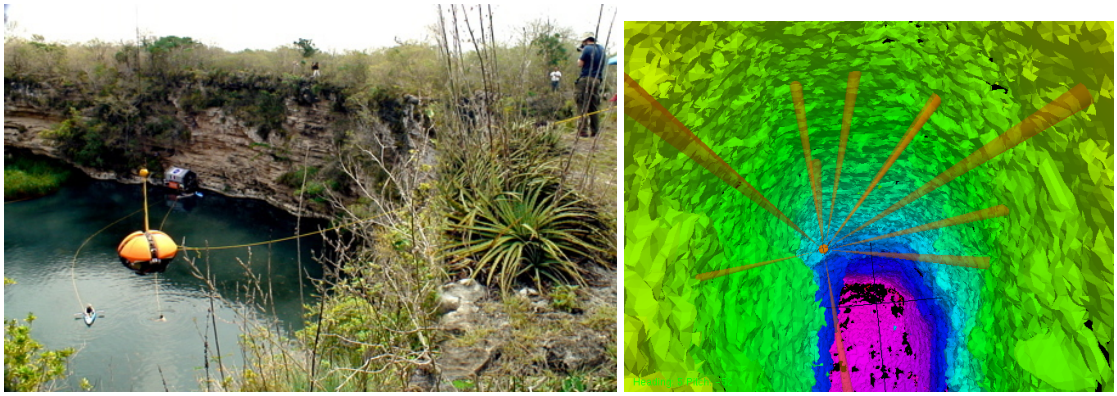

Figure 1.1: An early motivating problem was the exploration of a large flooded sinkhole (100 m diameter, 300 m depth) with an autonomous underwater vehicle using sparse sonar data (photo courtesy Dom Jonak).

robot with sparse and inaccurate sonar sensors to safely dive into the unknown depths and return after building an accurate metric map.

To clarify this task, we more carefully define the problems of mapping, localization, and planning:

Mapping is the problem of collecting and correlating a multitude of sensor measurements into a common map representation. There are many possible map representations, the main distinction being between feature-based maps, which are lists of features and information about them, and featureless metric maps, which directly represent the geometry of the environment. But all representations must deal in some way with sensor noise and uncertainty. In addition, as the robot moves around it must deal with uncertainty in its estimate of its pose in order to correctly correlate its sensor measurements.

Localization is the problem of using sensor measurements to estimate the robot's pose relative to some map. Localization must also deal with sensor noise and with uncertainty in the map. In some cases, localization may be unable to distinguish between two or more plausible poses within the map, especially if it is initialized without a certain estimate of its position within the map.

Planning is the problem of deciding what the robot should try to do next. Planning operates under constraints, such as safety, and it must also deal with uncertainty: in the map, in the current localization estimate, and in the outcome of actions. An important sub-problem is path planning, which is the problem of finding a safe and expedient route to a particular goal pose. Usually, a robot will have some over-arching task that largely determines what it should do next.

It is clear that these three problems are closely interdependent, and Figure 1.2 shows and names the overlapping regions between these problems. The interdependence between mapping (which uses the localization estimate) and localization (which uses the map) is clear, and is commonly called the Simultaneous Localization and Mapping (SLAM) problem. The dependence of planning on both mapping and localization is also clear. But it is important to note that mapping

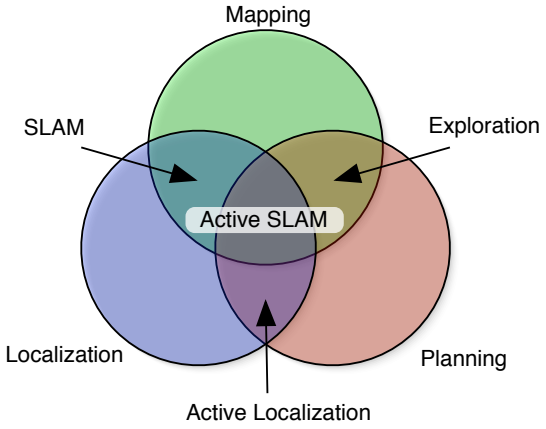

Figure 1.2: A decomposition of the tasks involved in Active SLAM, adapted from [Makarenko et al., 2002].

and localization are also both dependent on planning! This is because the planner can choose actions to deliberately reduce the uncertainty in the map and the localization estimate. This is because uncertainty, which can be quantified as *information entropy*, depends to a large degree on the route that the robot uses to explore the environment. The process of selecting actions that reduce uncertainty in both mapping and localization is known as Active SLAM. Active SLAM encompasses both the exploration phase, when the robot is building the map, and the operational phase, when the robot is using the map, with occasional refinement, to achieve its primary task.

But scaling up to truly large environments requires a second key insight beyond Active SLAM. Due to inevitable approximations, all known real-time SLAM solutions are limited in their scale of operation, usually on the order of a few hundred meters. The key insight is that the world can be divided up into more manageable pieces, or submaps (Figure 1.3). This changes the SLAM problem into a segmented SLAM problem.

Segmented SLAM represents the world using a combined metric and topological map, in which the relationships between submaps are represented by the edges of a graph, and the nodes of the graph point to the submaps. Since the robot is only ever in one place at a time, basic SLAM can be used in each one of these submaps in turn. Thus the scaling problem is avoided, but the trade-off is that segmented SLAM must manage the graph of submaps, deciding when to create a new submap, checking whether the robot has re-entered an old submap, and refining the topological relationships between submaps.

1.1 Approach

At the core of our approach is a sparse map representation that is based on evidence grids. This representation does not rely on features, which may only be available in certain environments, but instead uses active range measurements from lidar and sonar to build accurate metric representations of large 3D environments.

Our approach builds a particle filter-based SLAM method on this metric map representation.

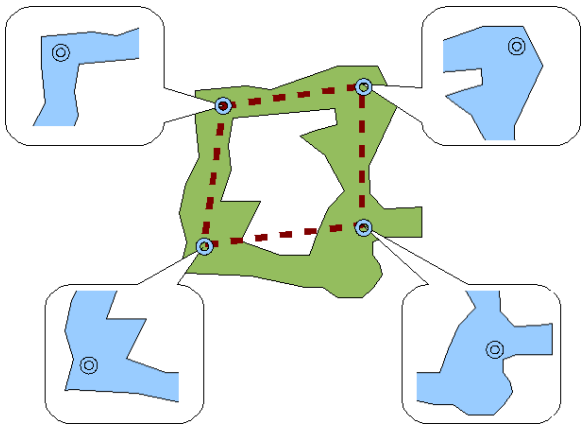

Figure 1.3: In this metric/topological map example, the real world (in green) has been segmented into four submaps. The relationships between the submaps are represented by the topological graph (dotted red lines).

This method has real-time guarantees, and is able to cope with initial uncertainty, as well as uncertainty that arises from ambiguity in the environment, by tracking multiple hypotheses about the robot location and map. Due to its basis in evidence grids and particle filters, our SLAM method is robust to poor sensor data.

In order to extend this SLAM method to yet larger scales, our approach segments the map into submaps a segmented map method, called SegSLAM. SegSLAM uses a heuristic based on the ability of the current submap to predict future measurements to find good divisions between submaps. Mapping and localization within these submaps is performed using a particle filter similar to our basic SLAM method, but generalized to allow particles to transition between segments.

SegSLAM keeps track of the topological and metrical relationships between segments with a map representation called the *segmented map*, a combined metric/topological map made up of the different versions of the metric submaps for each segment, as well the topological relations between segments expressed by particle transitions between segments. SegSLAM can generate *map samples* from the segmented map, and then directly compare and match metric submaps in order to find new connections between submaps, effectively using the submaps as macro features.

Our approach applies a stochastic planner to SegSLAM's segmented map, searching for actions that explore further, connect segments, and reduce uncertainty in order to perform Active SLAM. By evaluating plans in multiple map samples from the segmented map, a planner can estimate the SegSLAM uncertainty to find actions that are expected to reduce uncertainty. Further, the segmented map is used as a predictive model of the environment that allows the planner to make useful predictions about unobserved areas, particularly for finding actions that will connect back to previously explored regions.

Taken together, the components of our approach form a broadly applicable method that enables practical robotic exploration and operation in many large, three dimensional environments. We have applied this approach, and through experimental results have shown the following thesis.

Thesis statement: The exploration of large three dimensional environments can be addressed by factoring spatial uncertainty using a combined metric/topological map representation, and by integrating the tasks of localization, mapping, and planning so that the exploration strategy is informed both by the structure of the environment and by the ongoing uncertainty of the explorer.

1.2 Contributions

This research contributes to the field of robotics by:

- developing a compact and efficient map representation for three-dimensional environments
- developing algorithms for building, copying, and matching these maps, effectively treating maps themselves as macro features
- integrating this map representation method with a robust, real-time, multi-hypothesis method for SLAM in three dimensional environments
- developing SegSLAM, a flexible and robust segmented SLAM approach that addresses the problems of segmentation, loop closure, and scalability
- creating a stochastic planning method for real-time Active SLAM
- constructing a predictive model of the structure of the environment, allowing the stochastic planner to predict loop closures

We demonstrate these contributions on a variety of robots, with lidar and sonar sensors, in large 3D environments.

1.3 Organization

This dissertation is organized as follows:

Chapter 2 discusses map representations, and describes in detail the octree-based data structure that we have developed to efficiently represent large 3D environments. We show how to build these metric maps with range measurements, and present methods for comparing and matching overlapping maps. We present experiments that demonstrate map construction and matching using data from subterranean mines.

Chapter 3 formulates the Simultaneous Localization and Mapping problem, and presents a derivation of our particle filter-based approach to SLAM in unstructured environments. We present experimental results from running our SLAM algorithm onboard an autonomous underwater vehicle in flooded sinkholes, and demonstrate that our approach can build accurate metric maps of large natural environments in real time using low quality sensor data.

Chapter 4 introduces submap methods, which divide the world into manageable pieces and use a combined metric/topological to represent the resulting segmented map. We describe our SegSLAM algorithm, and discuss our approximate, real-time solutions to the attendant problems of segmentation, matching, and topological representation. We present experimental results to demonstrate these approaches to segmentation and loop closures, as well as SegSLAM's overall performance in several large, three dimensional environments.

Chapter 5 describes planning methods for selecting actions, or Active SLAM. We contrast information entropy-based approaches with our initial efforts to develop stochastic heuristics for selecting actions by sampling multiple map hypotheses from the SegSLAM representation. We present an experiment that demonstrates Active Localization on the ocean floor. In a second experiment, our stochastic planner demonstrates the ability to predict and close loops in a small tunnel environment.

Finally, Chapter 6 summarizes the contributions of this thesis and presents thoughts for future work.

Chapter 2

Representing Maps

2.1 Introduction

A map records relationships between elements in a region. There are many possible map representations – perhaps the most common are feature-based maps, in which the map is a list of uniquely identifiable features together with their positions, or their relative topology. The challenge with feature maps is to reliably detect and identify the feature, for example an orange traffic cone, and to correctly associate a feature seen at one time with the same feature seen later. The main advantage of features is that they are a very compact representation.

Another common representation is a series of captures of raw sensor data, also called signatures or views. The challenge with view-based maps is to develop techniques to compare views, so as to extract relative position information. Views can be used in situations where the sensor data is not good enough to do reliable feature detection, or when the set of features is unknown.

A third common representation, at least in the robotics community, is the evidence grid; a uniform discretization of space with the value of each cell assigned the probability of occupancy [Moravec and Elfes, 1985]. Since the entire space must be uniformly represented in memory, two dimensional evidence grids grow as the square of the maximum area to be mapped. The sheer size of a high-resolution evidence grid is a challenge, but they are well suited to situations where the raw sensor data is not good enough to do feature detection, nor sufficiently unique to capture a signature "view." In this case, data from multiple low-resolution or noisy sensor readings can be merged together in the evidence grid to form a complete map.

While 2D evidence grid-based SLAM is well established in the indoor mobile robot domain, it has limited applicability in truly 3D environments – largely because the 2D map simplification is only suitable in "two and a half"-dimensional environments, meaning those where only a single height needs to be associated with each 2D grid cell [Fong et al., 2003]. In a 3D evidence grid representation, space is divided into a grid of cubic volume elements, or voxels, which contain the occupancy evidence inferred from sensors. The focus of this thesis is to develop methods for the exploration of many real-world environments, and 3D evidence grids have the obvious advantage of being able to represent any static environment.

In this chapter we introduce the novel Deferred Reference Count Octree (DRCO) evidence grid data structure, an efficient 3D volumetric representation that exploits the spatial sparsity of

many environments. In addition, the DRCO is well-suited for use with the Rao-Blackwellized particle filter described in the next chapter. Next, we describe methods for comparing and matching evidence grid maps, which will be important for the submap methods of Chapter 4. We then touch on methods for computing map entropy, and finish with some experimental results in map matching.

2.2 Related Work in 3D Maps

Evidence grids, point clouds, meshes and topological graphs of connected features are all used to represent 3D maps – in fact it is common to use several representations in parallel for different tasks. Much of the work is based on highly accurate laser range-finder data. Nuchter et al. [2005] maps mine tunnels using 3D iterative closest point (ICP) on the dense laser point cloud data. Other work bridges the gap between "featureless" maps, such as point clouds and evidence grids, and feature-based maps by fitting geometric primitives to point clouds [Mahon and Williams, 2003, Weingarten and Siegwart, 2005, Hähnel et al., 2003].

Evidence or occupancy grids were introduced by [Moravec and Elfes, 1985], who also describe early work in 3D occupancy grids using stereo vision, as well as a map comparison metric [Martin and Moravec, 1996]. Hähnel et al. [2002] use 2D and 3D occupancy grids to distinguish between static and moving objects in order to filter people out of range data. They also align new range scans to evidence grids, but do not directly match two evidence grids. Yamauchi and Langley [1996] use 2D evidence grid maps as signatures for a toplogical network, and use a hill-climbing approach based on map comparisons to find transforms between maps. Stewart [1988] developed sonar beam models and demonstrated them with uniform 3D evidence grids. Early work by Connolly [1984] used range data to build octree occupancy models (without the probabilistic formulation of Elfes [1989]), and also described "trimming" the octree, which is what we call compacting. Evidence grids can be applied to other problems than estimating voxel occupancy, for example Ferri et al. [2007] use an evidence grid for plume mapping.

The DP-SLAM method of Eliazar and Parr [2006] incorporates a sophisticated data structure based on 2D evidence grids that exploits the similarity between particles of common ancestry to reduce the cost of copying and storing particle maps. However, in order to get linear ray-tracing performance, they must repeatedly process their data structure to create a cache of uniform "local maps", a complex and memory intensive process, even for 2D maps. Our DCRO does not require this extra book-keeping and yields the additional advantages of full 3D, sparse spatial representation, and overlap between particles with common ancestry – all inherent properties of the relatively simple DRCO data structure.

Kümmerle et al. [2007] extend 2D evidence grids by maintaining a list of levels for each cell, calling the resulting structure a multi-level surface map. While the resulting map is 3D, it is cumbersome to traverse and somewhat arbitrary about deciding when to add or merge levels based on new sensor data.

Θ	occupancy grid map
$\theta_{\langle x,y,z\rangle}$	the occupancy of the voxel with index $\langle x, y, z\rangle$
$p(\theta_{\langle x,y,z\rangle})$	the probability of occupancy of a particular voxel
$l_{\langle x,y,z\rangle}$	the log-odds of occupancy of a particular voxel
z_t	sonar range measurement at time t
$\mathbf{z}_t$	history of measurements from time 1 to t
$f(z)$	sonar beam model of $p(\theta\|z)$

Table 2.1: Evidence grid notation.

2.3 3D Evidence Grids

An evidence grid map Θ stores an estimate of some property for each cell of a uniformly discretization of space. In 3D, the cells are cubic units of volume, or voxels. The most common property is occupancy, so evidence grids are often also called occupancy grids [Martin and Moravec, 1996]. Occupancy cells are assumed to be Markov independent binomial random variables, which means that the evidence grid can be thought of as an extremely simple Markov Random Field. These assumptions are necessary so that the map can be quickly updated with new information. The primary operations on a map are inserting new evidence, querying to simulate measurements, and copying the entire map.

2.3.1 Updating Evidence Grids

The evidence grid formulation decomposes the problem of finding the distribution of maps given the history of sensor readings Z_t

$$p(\Theta|\mathbf{z}_t) = p(\Theta|z_1, ..., z_t) \tag{2.3.1}$$

into the problem of independently estimating the grid cell occupancy θ for each cell in the map:

$$p(\theta|\mathbf{z}_t) \tag{2.3.2}$$

by making the assumption that the cells are independent of each other. Often, the cells are initialized with $p(\theta) = 0.5$ to indicate an equal chance of being empty or occupied.

For each sensor reading, the probability of occupancy is recursively estimated with Bayes' rule:

$$p(\theta|\mathbf{z}_t) = \frac{p(z_t|\mathbf{z}_{t-1}, \theta)p(\theta|\mathbf{z}_{t-1})}{p(z_t|\mathbf{z}_{t-1})} \tag{2.3.3}$$

This update equation is simplified using the key assumption that the measurements z_i are conditionally independent given the cell θ

$$p(z_t|\mathbf{z}_{t-1}, \theta) = p(z_t|\theta). \tag{2.3.4}$$

This is a more general assumption than the *static world assumption*, which states that measurements are independent given knowledge of the map. This is a reasonable assumption when the

world is not changing and when we already know Θ:

$$p(z_t|\mathbf{z}_{t-1}, \Theta) = p(z_t|\Theta). \tag{2.3.5}$$

Often, the log-odds value for each voxel θ

$$l = \log\left(\frac{p(\theta)}{1 - p(\theta)}\right) \tag{2.3.6}$$

is stored in the map rather than the raw probabilities because it behaves better numerically, and because the Bayesian update rule for a particular voxel for some measurement z becomes a simple addition [Martin and Moravec, 1996]:

$$l = l + \overbrace{\log\left(\frac{p(\theta|z)}{1 - p(\theta|z)}\right)}^{\text{sensor model}} + \overbrace{\log\left(\frac{1 - p(\theta)}{p(\theta)}\right)}^{\text{map prior}}. \tag{2.3.7}$$

The first term on the right-hand-side is the sensor model, and the second is the map prior. If the prior $p(\theta) = 0.5$, the second term is zero and initialization simply sets all voxel log-odds values to zero. The update for each voxel can be reduced to simply summing the value of the sensor model with current voxel evidence. One important (and plainly false) assumption underlying the Bayesian insertion is that the cells are independent – that is that the occupancy of one cell is independent of the occupancy of any other cell. However, without this simplifying assumption evidence grids become intractable since the repercussions of updating a single cell could propagate through the entire map. The drawback is that maps tend to be more noisy in response to ambiguity in the measurements [Thrun, 2003].

2.3.2 Sensor Modeling

Occupancy is estimated from sensor measurements, such as a sonar range z, via a sensor model $f(z)$, which encodes the probability $p(\theta|z)$ for each cell θ that falls within the sensor's region of influence. As measurements are collected, the evidence they provide about the occupancy of each voxel is entered into the map according to the model, which is also called an inverse sensor model [Thrun, 2003] since it maps from the measurement to the physical world.

There are several methods which can be used to construct a beam model, including deriving it from physical first principles [Urick, 1983] or learning it [Martin and Moravec, 1996] using a ground-truthed map. For laser sensors we used a ray model, but sonars have significant beamwidth so we chose to use the simplest reasonable approximation – a cone that is loosely based on the beam-pattern of the sonar (Figure 2.1).

To insert the evidence given by a particular range measurement at a particular position and orientation, each voxel which falls within the beam is updated with the value of the sensor model at that point according to the Bayesian update equation 2.3.7. These updates, which are just additions, can be decomposed into fast raster operations that are performed by a 3D variant of the classic 2D Bresenham line drawing algorithm, also called *ray-tracing* [Bresenham, 1965]. For example, the sonar cone is drawn as a bundle of rays with constant negative value, with terminating voxels with constant positive values. Likewise, the simplest method to query both laser

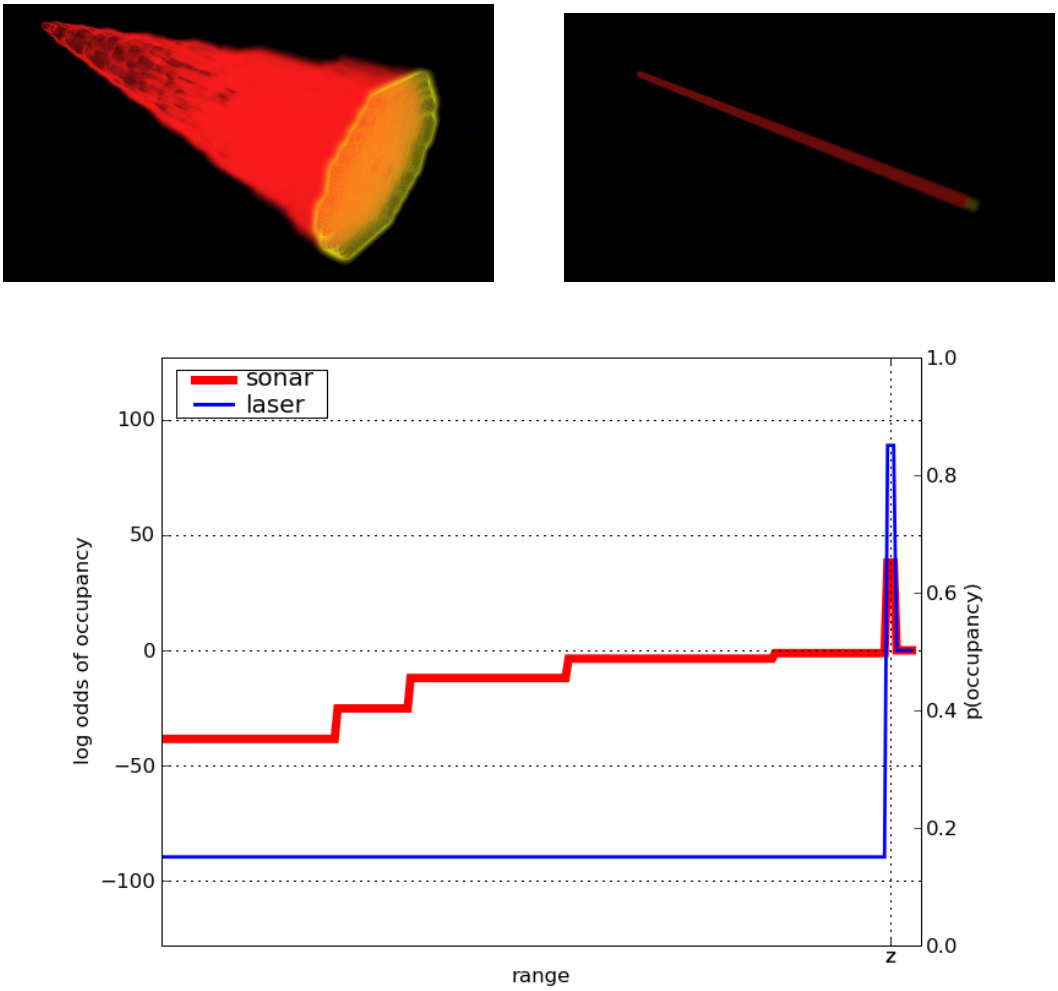

Figure 2.1: Top left: perspective view of a sonar beam model with a 6° beadwidth inserted into an evidence grid. Top right: perspective view of a laser ray model. Bottom, the log odds and probability of occupancy for a ray down the center of both models.

and sonar ranges from the 3D evidence grid is to trace a ray until some threshold (or other terminating condition) is met. Using matrix transformations for each voxel is too computationally expensive for operations such as filling in evidence cones or simulating ranges.

2.4 The Deferred Reference Counting Octree

The main difficulties with 3D evidence grids arise from the cost of copy operations and the storage requirements that increase with map size and resolution. If we store the evidence log odds as single bytes (with values between -128 and 127), then an evidence grid 1024 cells on a side requires a megabyte of memory in 2D and a gigabyte in 3D. A typical memory bus can handle transfer rates of around 400 Mb/s, and so would require over two seconds to copy such

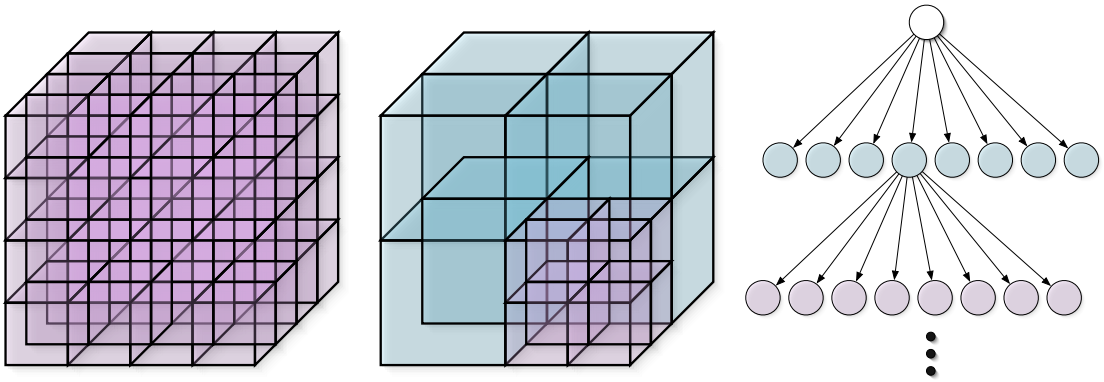

Figure 2.2: While a uniform grid (left) must represent every cell, each level of an octree divides the remaining volume into eight octants (center) and the octree can efficiently represent sparse volumes because the tree structure (right) does not have to be fully expanded.

a map. For a Rao-Blackwellized particle filter (Chapter 3), we need to store and copy hundreds of maps per second to operate in real-time. This requires a more efficient data structure than a uniform array; the octree is one such structure.

An octree is a tree structure composed of a node, or *octnode*, which has eight children that equally subdivide the node's volume into *octants* (Figure 2.2). The children are octnodes in turn, which recursively divide the volume as far as necessary to represent the finest resolution required. The depth of the octree determines the resolution of the leaf nodes. The main advantage of an octree is that the tree does not need to be fully instantiated if pointers are used for the links between octnodes and their children. Large contiguous portions of an evidence grid are either empty, occupied, or unknown, and can be efficiently represented by a single octnode – truncating all the children which would have the same value. As evidence accumulates, the octree can compact homogeneous regions that emerge, such as the large empty volume inside a cavern. Note that even with compaction the octree supports the insert, query, and copy operations, and is a drop-in replacement for the uniform array: it is possible to convert losslessly between the two representations. Insert and query can be done with a tree-traversing ray-tracing algorithm (see [Havran, 1999] for an overview).[1] We employ our own Bresenham 3D top-down approach, largely for its simplicity, which runs within a small constant (about 3) of Bresenham 3D on uniform evidence grids.

To improve performance, we use our own custom memory management for the octnodes. Octnode memory is created as needed in large blocks (about 10k nodes) which are initialized as a FIFO linked list of free nodes. The first word of each node points to the next free node. As nodes (and entire octrees) are freed, nodes are pushed onto the front of this linked list. When a new node is needed, the first entry of the linked list is pulled off and initialized with default values.

[1] Octrees have been widely used in computer graphics to sort polygons in a different application of ray-tracing.

Procedure 1 DRCO COPY(A to B)

Require: A and B are pointers to octnodes

```
1: A→defC++;
2: LAZYFREENODE(B); // See Procedure 2
3: B = A;
```

2.4.1 Reference Counting Octree (RCO)

The common approach to copying an octree is simply to traverse the entire tree copying every node (see the "Naïve" row of Figure 2.3). This is an expensive operation that can be improved if different copies share subtrees by maintaining reference counts to the octnodes: each node keeps track of the number of references to itself. We refer to this number as refCount. With reference counts, naïve copying can be replaced with copy-on-write (see the "Reference Counting" row of Figure 2.3) as follows. An octree is copied by creating a new pointer to the root node of the source octree and incrementing the refCount of all the child nodes. When either the copy or the original is modified, only subtrees which have a reference count greater than one need to be copied. Once a copy has been made, the reference count of the source is decremented. This is "copy-on-write", which is particularly useful in situations where queries to the maps are more common than insertions.

2.4.2 Deferred Reference Counting Octree (DRCO)

However, the bookkeeping for the reference counts requires a full traversal of the octree, or at least of the modified subtree, which is almost as expensive as copying. Our novel solution is to maintain deferred reference counts. Every octnode has a refCount and also a deferred reference count, or defCount (see the "Deferred Reference Counting" row of Figure 2.3). The defCount represents reference counts which have not yet been propagated to the children. The true reference count of the node is the sum of refCount and defCount, but changes in defCount do not usually trigger a full traversal through the subtree.

Copying a map now simply requires creating a new pointer to the root node of the source and incrementing the defCount (Procedure 1). If either of the two copies is modified then copy-on-write code propagates the defCount down the tree, copy the portion of the tree which will be changed, and setting the reference counts accordingly (see the "Deferred Reference Counting" row of Figure 2.3 and Procedure 3).

LazyFreeNode applies the same deferred (or lazy) principles of recursively freeing octnodes: if the octnode has defCount > 0, then it can just decrement the defCount (see Procedure 2). If the defCount $= 0$, then LazyFreeNode recurses on the child nodes.

The ray-trace write, or "insert," operation in the DRCO must deal with properly updating all these counts – the ray-trace read, or "query," operation is unchanged. When we want to insert updates into a map, copy-on-write propagates the node's defCount down to its children, sets the refCount = refCount + defCount, and sets defCount = 0 (since defCount represents the reference counts that the node hasn't propagated to its children) (see Procedure 3). If the new refCount is > 1, the node must be copied (including the pointers to its children). The deferred updating

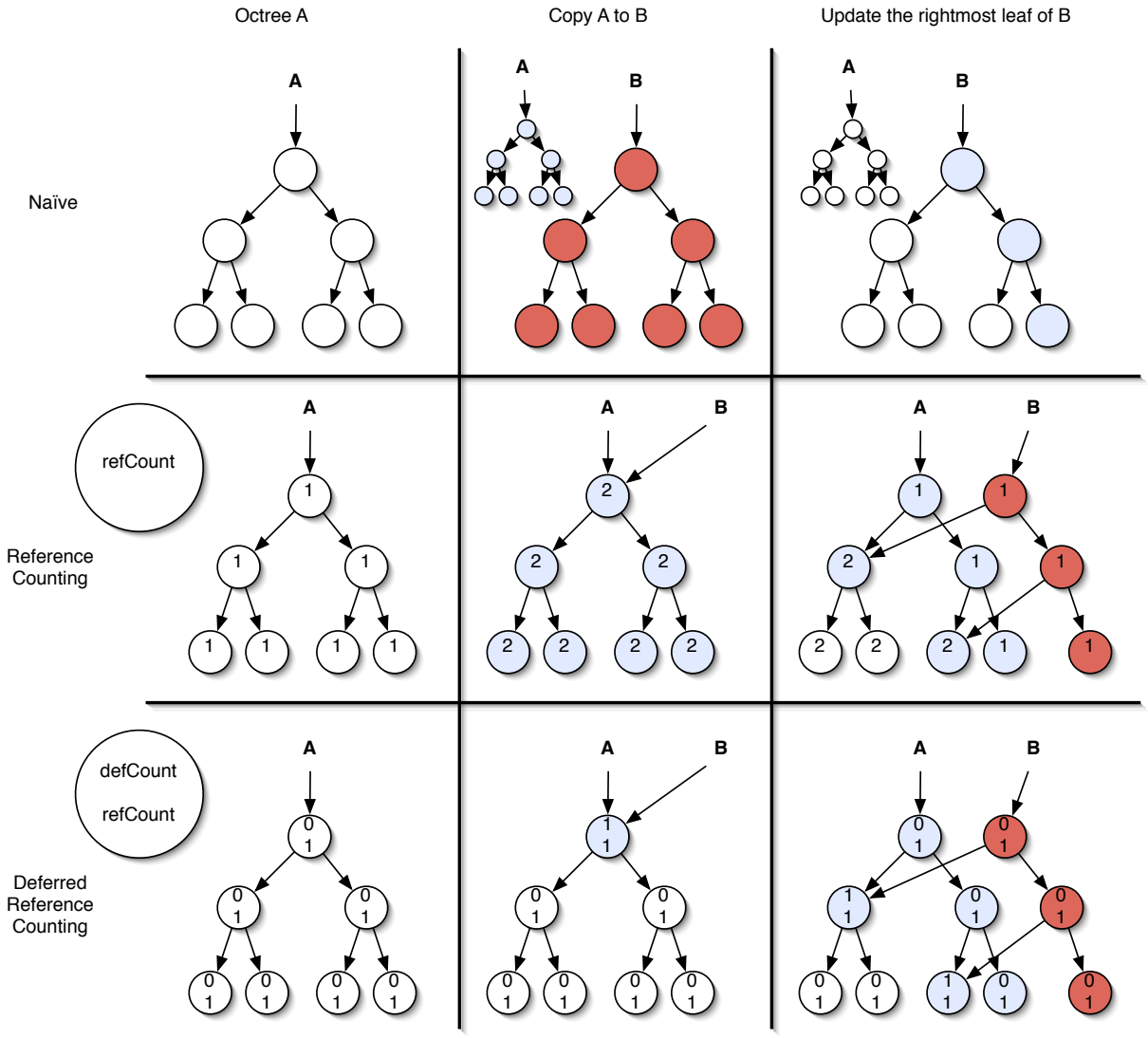

Figure 2.3: Octree diagram demonstrating the copy and write operations for three octree implementations: Naïve, Reference Counting, and Deferred Reference Counting. Blue (light gray) nodes are nodes that must be accessed, and red (dark gray) nodes are new nodes. Note that in the lower right corner (DRCO Update), the rightmost leaf does not change value, but is accessed and modified in the course of the update.

works because the insert procedure always starts at the top of the octree – this top-down property is part of our ray-trace implementation.

Our scheme for using deferred reference counts is related to the work of Baker [1994] on "anchored pointers" for functional data structures.

2.4.3 Compaction

For environments with some degree of spatial sparsity, DRCO's yield a significant reduction in map storage size and allow us to represent maps that would not even fit into memory as a uniform

Procedure 2 DRCO LAZYFREENODE(N)

Require: N is a pointer to an octnode

```
1: if N→defC > 0 then
2:     N→defC--;
3:     return
4: end if
5: N→refC--;
6: for i = 1...8 do // Recursively free children
7:     LAZYFREENODE(N→child[i]);
8: end for
9: if N→refC == 0 then // Free the node
10:    N→nextNode = globalFreeNode;
11:    globalFreeNode = N;
12: end if
```

array. But we can further reduce storage requirements by periodically compacting the octrees.

Compaction is the process of recursively traversing the tree, replacing groups of homogeneous children with a single parent with the same value. As regions of the map become well explored, fuzzy compaction can simplify regions with small amounts of noise. Periodically compacting the octrees can yield significant space savings, especially when the map will only be used for querying – in which case the values can be lossily thresholded to {0=empty, 1=occupied} (see Table 2.2).

Map	Size (MB)	Size Ratio
Uniform 1-byte/voxel evidence grid	128 MB	180%
Uncompacted octree map	**71 MB**	**100%**
Lossless compaction	55 MB	77%
{Empty, unknown, occupied} compaction	13 MB	18%
{Empty, unknown, occupied} fuzzy compaction	8 MB	11%

Table 2.2: Compaction results on a real-world map, 512^3 map = 128^3 m at 0.25 m resolution. Size ratio is relative to the uncompacted octree map.

Compaction can also have an effect on ray-tracing speed: the iterative ray tracer is accelerated by large homogeneous regions (see Table 2.3. While the octree ray-tracer is about three times slower than the uniform ray tracer, this speed difference drops dramatically when the octree is compacted. Both the compaction and the resulting ray-tracing acceleration depend on the volumetric sparsity of the particular environment.

2.5 Map Matching with 3D Evidence Grids

As discussed in the introduction, the DRCO is the map representation for our Rao-Blackwellized particle filter SLAM approach described in the next chapter. It is also be the submap representa-

Procedure 3 DRCO SETNODE(N, value)

Require: N is a pointer to an octnode, value is some constant

```
 1: if N→defC + N→refC > 1 then
 2:     if N→defC > 0 then
 3:         for i = 1...8 do // Propagate defC to children
 4:             if N→child[i] != 0 then
 5:                 N→child[i]→defC += N→defC
 6:             end if
 7:         end for
 8:     end if
 9:     newN= NEWNODE();
10:     COPYNODE(newN, N);
11:     newN→refC = 1;
12:     newN→defC = 0;
13:     newN→value = value;
14:     N→refC = N→defC + N→refC - 1;
15:     N→defC = 0;
16: else
17:     N→value = value;
18: end if
19: N = NEXTNODEINTOPDOWNTRAVERSE()
20: if N != 0 then
21:     SETNODE(N, value)
22: end if
```

Map	RPS	Map size
Uniform Array	482000 rps	512^3 bytes = 128 MB
Uncompacted Octree	161000 rps	73 MB
Compacted Octree	375000 rps	1.4 MB

Table 2.3: Ray tracing results in simulated 512^3 map, showing the number of query rays per second (RPS) for each map representation.

tion for our segmented SLAM approach described in Chapter 4. When used as submaps, we will need methods for matching two overlapping maps, as well as methods for evaluating the match quality. In this section, we describe methods for doing map matching with 3D evidence grids, and compare these methods with the standard approach, ICP.

The standard method for matching two sets of range data is to convert the ranges to a point cloud representation, and then use one of the many variants of Iterative Closest Point (ICP) [Chen and Medioni, 1991, Besl and Mckay, 1992]. While ICP is a very general and flexible technique, it can be slow to converge or converge to local minima, requiring the adjustment of several parameters that depend on the number of points, the point density, and the amount of noise in the data. Further, there are several properties of ICP that result in a tradeoff between

speed and accuracy. First, the nearest-neighbor computation is expensive with large numbers of points, such as the dense 3D point clouds from scanning lasers that produce about 70000 points per scan. It requires $O(m \log(n))$ to find m nearest neighbors amongst n points, even with an efficient search algorithm, such as a k-d tree [Freidman et al., 1977], yielding a total complexity for ICP of $O(n \log(n))$. The flip side is that the accuracy suffers when the range values are sparse or noisy or both, such as those provided by stereo vision or sonars [Diebel et al., 2004]. Also, when the two point clouds do not overlap well (as in the case of a robot that is moving while collecting scans), then ICP must know the fraction of non-overlapping points to keep them from perturbing the solution.

A final issue is that ICP does not utilize an important aspect of the information implied by the range data. By reducing the data to a point cloud, it discards the fact that the range values were collected by bouncing rays from a source to an reflective object, and the implication that the volume of space between the source and reflective object must, in fact, be empty.

We present a set of alternative methods for matching 3D range scans based directly on evidence grid maps. Evidence grids are robust to noise and variations in point density, can incorporate an indefinite number of ranges, and explicitly encode empty as well as occupied space. While 3D evidence grids can be huge when naively implemented, we use an optimized octree data structure to efficiently store sparse volumetric maps. To register a series of range scans, we build an evidence grid map for each scan, and then register them together using a several different methods.

The first two methods are based on a 3D extension of the classic 2D Lucas-Kanade template matching method, and differ only in whether we match a single large region, or multiple small regions that are selected heuristically. Our third and fourth methods involve extracting surfaces or point clouds from the evidence grids, and then running ICP – this effectively uses the evidence grid as a binning structure to uniformly resample the large number of data points.

2.5.1 Related Work in Map Matching

ICP with laser data is a very common combination in the field of robotics. An enormous number of variants have been proposed that differ in how the points are selected, matched, and weighted, many of which are compared by Rusinkiewicz and Levoy [2001]. As representative examples, Madhavan et al. [1998] use 2D ICP, as do Thrun et al. [2003], who also use a 2D global alignment algorithm to close loops. Huber and Vandapel [2003] convert the laser data to surfaces and match the surfaces, while Nuchter et al. [2005] use 3D ICP to estimate all 6 degrees of freedom (DOF) of the robot pose.

While ICP is an excellent method for use with laser range finder data, when the pointclouds are dense,precise and have a large amount of overlap, it is much harder to use when the range values are sparse or noisy, such as are provided by stereo vision or acoustic range finders (sonar) [Diebel et al., 2004], though it has been successfully employed with multibeam sonar data [Roman, 2005]. Recent approaches based on conditional random fields are more robust, and can incorporate other sources of match information [Ramos et al., 2007].

There may be an interesting connection between our method of matching with probabilistic evidence grids and the 3D normal distributions transform (3D NDT) method of Magnusson [2006] that models the probability of finding a point at a particular location.

For evidence grids, Martin and Moravec [1996] describe probabilistic methods for comparing evidence grid maps, primarily in the context of comparing experimentally-constructed maps with ground truth. We will use this approach as our gold standard check, though it is not well suited to finding the registration between maps.

2.5.2 Baseline ICP Implementation

We implemented k-d tree accelerated approximate nearest-neighbor ICP in C using the ANN library of Mount and Arya [1997]. The nearest neighbor search can be accelerated by adjusted the error bound ϵ that allows the distance to the approximately nearest neighbor to exceed the true distance to the nearest neighbor by a factor of $(1 + \epsilon)$. Thus, if $\epsilon = 0$, the algorithm will return the true nearest neighbor. Due to the reasonable initial pose estimates, we chose to eschew the computation of normals for more selective matching. However, we did implement a common variant of ICP in which only points with neighbors nearer than a maximum threshold (for example, 0.2 m) are counted, reducing the effect of outliers, and the tendency of ICP to be perturbed toward dense edges.

Several extensions for accelerating 3D ICP by down-sampling the points have been proposed [Rusinkiewicz and Levoy, 2001] [Gelfand et al., 2003], primarily for situations in which the initial scan alignment is poor, the overlap between scans is small, or when the meshes contain holes. Broadly speaking, none of these conditions are present in our experimental data (see below). We down-sampled the point clouds to around 100 thousand points, and adjusted ϵ and the clipping threshold described above. We believe that our implementation provides a reasonable performance baseline, however since an average scan consisted of 100 thousand points in a 3D point-cloud, it is likely that further specialized down-sampling could dramatically improve the performance of the ICP matcher. Generally, our ICP matching speed compares favorably with other published ICP performance [Greenspan and Yurick, 2003] [Nuchter et al., 2005].

2.5.3 Evidence Grid Match Score

For a match transform T_2^1, we can evaluate the match between two maps Θ_1 and Θ_2 by transforming every cell in map Θ_2 into the coordinate frame of map Θ_1, and comparing their values in overlapping regions. As described by Martin and Moravec [1996], the match between two particular voxels θ_1 and θ_2 is the probability that they both represent the same value, which is equal to the sum of the probability that they are both occupied and the probability that they are both empty:

$$p(\theta_1, \theta_2) = p(\theta_1)p(\theta_2) + (1 - p(\theta_1))(1 - p(\theta_2)), \tag{2.5.1}$$

and we take this product over all of the overlapping voxels in the two maps:

$$\text{MatchScore}(\Theta_1, \Theta_2) = \prod_{\{\theta_1, \theta_2 : T_2^1 \theta_2 = \theta_1 \in \Theta_1\}} p(\theta_1, \theta_2). \tag{2.5.2}$$

Since the maps store log odds,

$$l = \log(\frac{p(\theta)}{1 - p(\theta)}) \tag{2.5.3}$$

we have to back out the probabilities:

$$p(\theta) = \frac{\exp(l)}{(1 + \exp(l))}. \tag{2.5.4}$$

It would be computationally intensive to transform every point from one map to to the coordinate frame of the other map so as to be able to make a comparison. The number of voxels that need to be compared can be reduced by carefully calculating the region of intersection and only considering voxels within that region. Furthermore, since both frames have the same scale, we can reduce the number of point transformations significantly (to a minimum of three) by tracing corresponding rays through the two maps and caching the start to end point offset. This method is susceptible to ray-trace aliasing effects; even so, the direct comparison method is too slow to use as the evaluation metric for some sort of hill climbing algorithm. It does provide a gold standard for comparison with other methods, such as the 3D-Lucas Kanade method described next.

2.5.4 3D Lucas-Kanade Algorithm

The Lucas-Kanade (LK) matching method [Lucas and Kanade, 1981] is a classic 2D image tracking technique from the machine vision community that aligns a template image $T(\mathbf{u})$ with an input image $I(\mathbf{u})$, where $\mathbf{u} = (u, v)^T$ are the pixel coordinates. Following the derivation of Baker et al. [2004] of a computationally more efficient form called the *inverse compositional algorithm*, which we will call icLK, the algorithm can be generalized to 3D by considering the uniform 3D arrays $T(\mathbf{x})$ and $I(\mathbf{x})$, where $\mathbf{x} = (x, y, z)^T$ are the voxel coordinates.

The goal of icLK is to warp I to match T so as to minimize

$$\sum_{\mathbf{x}} [T(w(\mathbf{x}; \Delta\mathbf{p})) - I(w(\mathbf{x}; \mathbf{p}))]^2 \tag{2.5.5}$$

over all the voxels $\mathbf{x}$ in the template T, and where $w(\mathbf{x}; \mathbf{p})$ is a warp with parameters $\mathbf{p} = (p_1, \ldots, p_n)^T$ that maps a voxel $\mathbf{x}$ in template T to an interpolated sub-voxel $w(\mathbf{x}; \mathbf{p})$ in image I. A basic translation-only warp is

$$w_{\mathsf{3DOF}}(\mathbf{x}; \mathbf{p}) = \mathbf{x} + \mathbf{p} = (x + p_1, y + p_2, z + p_3)^T \tag{2.5.6}$$

where the parameters represent coordinate translations, and the full 6DOF warp is

$$w_{\mathsf{6DOF}}(\mathbf{x}; \mathbf{p}) = R\mathbf{x} + \mathbf{t} \tag{2.5.7}$$

where $\mathbf{t}$ is the translation parameterized by p_1, p_2, and p_3, and R is the rotation matrix parameterized by p_4, p_5, and p_6.

Since many robots have very accurate roll and pitch data, we focus on the translation-plus-yaw warp, which we will call the 4DOF warp:

$$w_{\mathsf{4DOF}}(\mathbf{x}; \mathbf{p}) = \begin{pmatrix} \cos(p_4) & -\sin(p_4) & 0 & p_1 \\ \sin(p_4) & \cos(p_4) & 0 & p_2 \\ 0 & 0 & 1 & p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.5.8}$$

icLK first linearizes Equation 2.5.5 by plugging in a first order Taylor series expansion about the identity warp $w(\mathbf{x}; \mathbf{0}) = \mathbf{x}$ for $T(w(\mathbf{x}; \Delta\mathbf{p}))$:

$$\sum_{\mathbf{x}} [T(\mathbf{x}) + \nabla T \frac{\delta w}{\delta \mathbf{p}} \Delta\mathbf{p} - I(w(\mathbf{x}; \mathbf{p}))]^2 \tag{2.5.9}$$

where $\nabla T = (\frac{\delta T}{\delta p_1}, \frac{\delta T}{\delta p_2}, \frac{\delta T}{\delta p_3}, \dots)$ is the gradient of template T evaluated at $w(\mathbf{x}; \mathbf{p})$, and $\frac{\delta w}{\delta \mathbf{p}}$ is the Jacobian of the warp. The image gradients are frequently computed by convolving a difference kernel along each dimension. We used the simplest such kernel, $[1, 0, -1]$, for the translational gradients, but for the rotational (yaw) gradient, we found the best results when we used the surprisingly large rotation $\pi/4$, so that if $w_\theta(\pi/4)$ is the warp that rotates the x and y coordinates by $\pi/4$:

$$I_\theta = I(w_\theta(\pi/4)) - I(w_\theta(-\pi/4)) \tag{2.5.10}$$

icLK then finds the least-squares solution to Equation 2.5.9 by taking the partial derivative with respect to Δp:

$$2 \sum_{\mathbf{x}} \left[\nabla T \frac{\delta w}{\delta \mathbf{p}} \right]^T [T(\mathbf{x}) + \nabla T \frac{\delta w}{\delta \mathbf{p}} \Delta\mathbf{p} - I(w(\mathbf{x}; \mathbf{p})] \tag{2.5.11}$$

and setting this expression to zero and solving for $\Delta\mathbf{p}$:

$$\Delta\mathbf{p} = -H^{-1} \sum_{\mathbf{x}} \left[\nabla T \frac{\delta w}{\delta \mathbf{p}} \right]^T [T(\mathbf{x}) - I(w(\mathbf{x}; \mathbf{p}))] \tag{2.5.12}$$

where H is the Gauss-Newton approximation to the Hessian:

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\delta w}{\delta \mathbf{p}} \right]^T \left[\nabla T \frac{\delta w}{\delta \mathbf{p}} \right]. \tag{2.5.13}$$

For the 4DOF warp,

$$H = \begin{pmatrix} \sum I_x I_x & \sum I_x I_y & \sum I_x I_z & \sum I_x I_\theta \\ \sum I_y I_x & \sum I_y I_y & \sum I_y I_z & \sum I_y I_\theta \\ \sum I_z I_x & \sum I_z I_y & \sum I_z I_z & \sum I_z I_\theta \\ \sum I_\theta I_x & \sum I_\theta I_y & \sum I_\theta I_z & \sum I_\theta I_\theta \end{pmatrix} \tag{2.5.14}$$

where $I_i = \frac{\delta I}{\delta i}$, and the sums are over the element-wise multiplication of the corresponding image gradients.

The main advantage of the inverse compositional algorithm is that the gradient ∇T, the Jacobian of the warp $\frac{\delta w}{\delta \mathbf{p}}$, the steepest descent images, $\nabla T \frac{\delta w}{\delta \mathbf{p}}$ and the Hessian H can all be precomputed and do not have to be recomputed for each iteration of Equation 2.5.12. However, the warp must be updated as the inverse composition

$$w(\mathbf{x}; \mathbf{p}) = w(w(\mathbf{x}; \Delta\mathbf{p})^{-1}, \mathbf{p}). \tag{2.5.15}$$

As a result, an extra requirement of icLK is that it can only be applied to sets of warps that form a group – luckily 3D rotations in general and w_{4DOF} in particular form groups [Shum and Szeliski, 2000].

icLK is equivalent to Lucas Kanade to the first order in Δp [Baker et al., 2004], and is only $O(PV + P^2)$ per iteration, compared with $O(P^2V + P^3)$ for Lucas-Kanade, where V is the number of voxels and P is the number of parameters; in our case $P = 4$. This may or may not be better than ICP's $O(n \log(n))$ in the number of pointcloud points.

2.5.5 Region Selection

Since icLK computation is linear in volume, matching two entire maps (or at least their overlapping region) with icLK is more computationally intensive than matching smaller regions, scaling linearly in the volume of the region. Rather than matching the entire maps, we could consider matched multiple small regions drawn from within the maps. Ideally, these small regions would be selected such that they contained strongly directional map data in both maps, like a corner (Figure 2.4). While it is possible to implement corner detectors in 3D (for example, extending the Harris detector [Harris and Stephens, 1988]), we preferred to make no assumptions about environmental structure and do no feature detection. Rather, we took the median along each dimension of the matches of a set of smaller, randomly generated, regions that were drawn from a distribution designed to generate regions likely to include a portion of the tunnel wall and ceiling (Figure 2.4) to the left and right of the current vehicle position. Before performing matching, each region was verified to contain a significant amount of positive evidence in both maps. This heuristic assured that icLK had a portion of a wall to work with. Taking the median match of multiple regions (10 in our experiment) yielded a more robust estimate of the map match than using any fixed set of robot-relative regions.

2.5.6 Isodox Surface Matching

An evidence grid accumulates "belief" about the occupancy of each voxel, estimating the probability that the voxel is occupied according to the evidence accumulated from sensor measurements. An isodox surface is the surface that separates regions of differing belief. We can extract this surface from using Marching Cubes [Lorensen and Cline, 1987]), which, like icLK, runs in O(V), the volume of the mapped space. This yields a uniformly sampled set of points (depending on the surface area of the mesh), which we then match with ICP.

Alternatively, we can simply traverse the octree and generate a point for each leaf node which exceeds a threshold. For a compacted octree (see above), this runs roughly on the order of the surface area of the mapped structures O(S) – which can be a tremendous improvement over O(V) if the environment is sparse. Using the octree in this way is almost equivalent to binning the pointcloud to reduce the influence of point density, with the advantage that the empty evidence of the beams can suppress noisy measurements.

In this section we have described a powerful set of approaches to matching maps: ICP, icLK, Isodox surface matching with ICP, and Octree-binning. We will compare results from these methods in Section 2.7, and use several of these methods as building blocks for the work described in Chapters 4 and 5. In the next section, we discuss evidence grid entropy, which is also an important building block for later work.

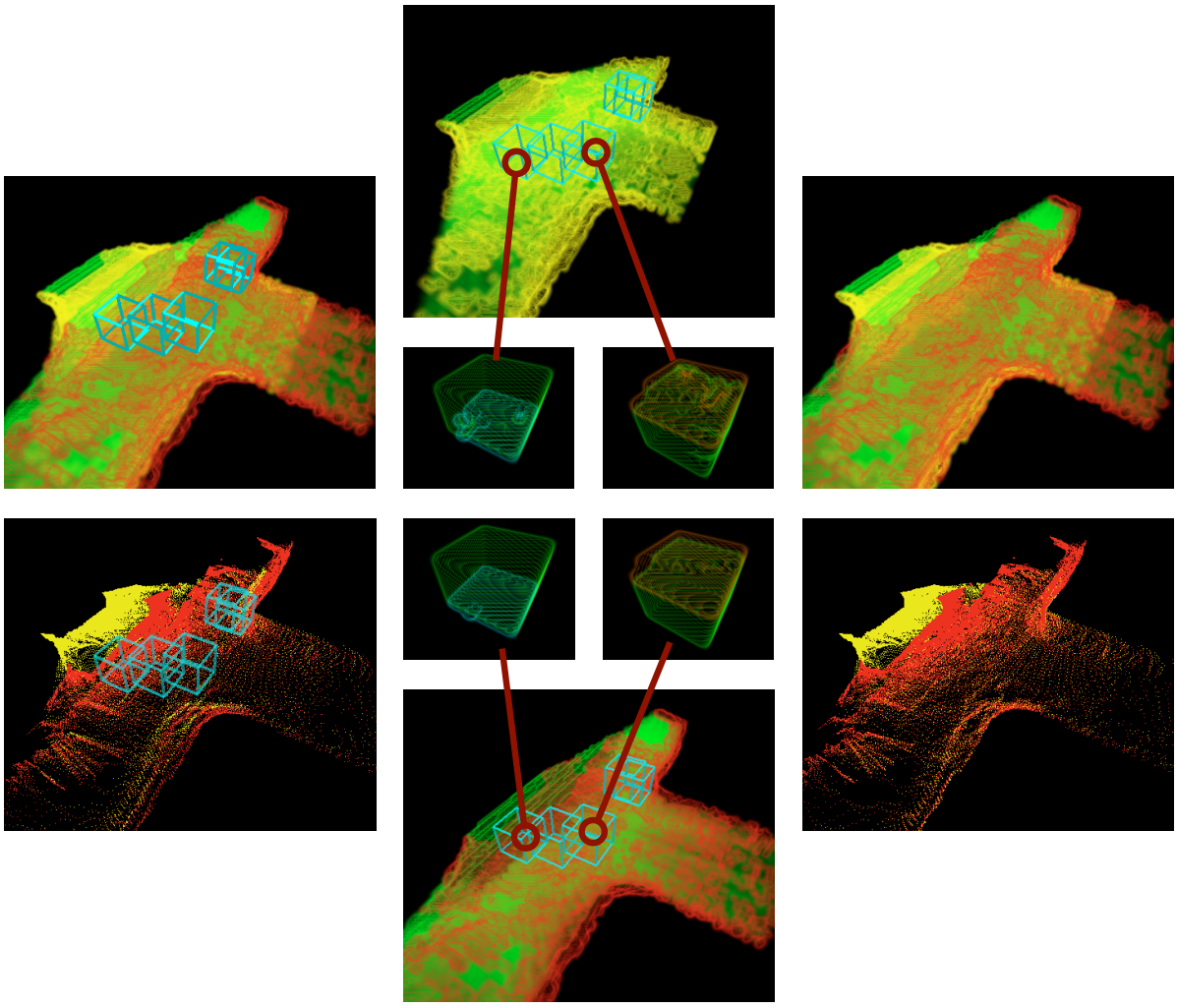

Figure 2.4: The icLK map matching process performed at a four-way tunnel intersection. At top center and bottom center are the two maps that are being matched, represented as evidence grids. The cyan boxes show five regions that have been selected according to the method in the text. On the left, the maps have been superimposed to show the initial mis-registration, represented as evidence grids (top left) and point clouds (bottom left). The center of the figure shows two example pairs of regions that have been extracted from each of the evidence grids: icLK is run to match each pair of these small regions. The right of the figure shows the resulting match, both as evidence grids (top right) and point clouds (bottom right).

2.6 Evidence Grid Entropy

Uncertainty or randomness in an estimate, such as an evidence grid map, can be quantified using the information entropy of Shannon [1948]. Information entropy H for a discrete random variable X with possible values $\{x_1, \ldots, x_N\}$ is defined as

$$H(p(X)) = -\sum_{i=1}^{N} p(x_i) \log p(x_i) = -E_X[\log p(x)] \tag{2.6.1}$$

or extended to the case when X has a continuous domain, in which case it is called the continuous or differential entropy,

$$H(p(X)) = -\int_X p(x) \log(p(x)) dx. \tag{2.6.2}$$

Each voxel $\theta_i[x, y, z]$ in the evidence grid map θ_i is a Bernoulli random variable that estimates whether the voxel is empty or occupied. Thus the entropy of a voxel is

$$H(\theta[x, y, z]) = -\rho \log(\rho) - (1 - \rho) \log(1 - \rho), \tag{2.6.3}$$

where $\rho = p(\theta_i[x, y, z])$. Under the independence assumptions of an evidence grid, the information-theory entropy of θ_i is the sum of the entropy of each voxel:

$$H(\theta) = \sum_{\forall x,y,z} H(\theta[x, y, z]) \tag{2.6.4}$$

The problem with a direct application of this metric is that the entropy of a map will vary depending on the map resolution and size because unobserved cells with $p(\theta_i[x, y, z]) = 0.5$ contribute maximum entropy, and changing resolution or size with change the number of cells, even though the observations (and the entropy) remain unchanged. This problem can be addressed by ignoring unobserved cells and adding a scaling factor γ based on the volume of the voxel:

$$H(\theta) = \sum_{x,y,z \in \text{Obs}} \gamma H(\theta[x, y, z]) \tag{2.6.5}$$

Equivalently, Blanco et al. [2008a] propose a slightly more complicated definition of map information as

$$I(\theta) = \begin{cases} \frac{\sum(1 - H(\theta[x,y,z]))}{\#_{Obs}} & \text{if } \#_{Obs} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6.6}$$

Either approach will yield the desired robustness to voxel scale and invariance to the number of unobserved cells.

Octree Entropy

Since the octrees are generally sparsely populated, ignoring unknown regions changes from a requirement to a huge speed advantage – the octree doesn't represent unknown areas. A further

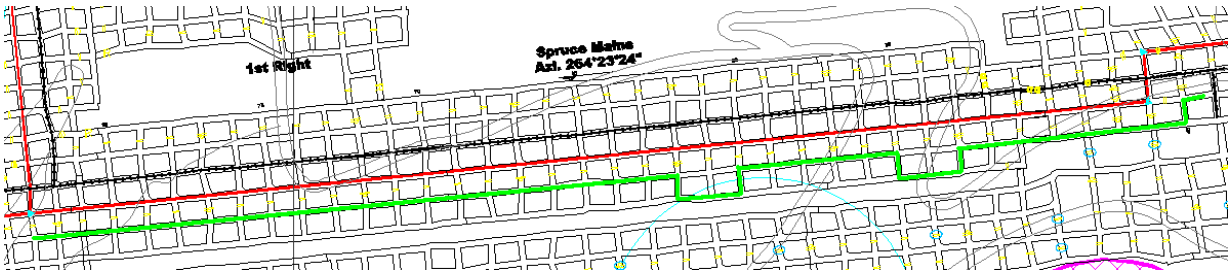

Figure 2.5: The surveyed map of the Dakota mine, showing the true robot trajectory in green, starting on the right and ending on the left.

speed improvement arises because the octree can represent homogeneous regions at a variety of leaf node scales, and all we need to do is change the scale factor in Equation 2.6.5:

$$H(\theta) = \sum_{\nu \in \theta} \gamma_\nu H(\nu) \tag{2.6.7}$$

where ν is a leaf node and γ_ν is a scaling factor that depends on the size of the leaf node. Computation is therefore linear in the observed volume rather than the map extent, which makes it feasible to compute entropy over large octree maps. And as we have already observed, for compacted octree maps, computation is nearly linear in the observed surface area.

2.7 Experiments

2.7.1 Subregion icLK – Groundhog in Dakota Mine

In this experiment, we compare the performance of sub-region icLK with a standard ICP-based approach, using data from the Dakota mine. The Groundhog mine mapping vehicle was deployed in the Dakota Mine No. 2 underground coal mine in March 2005. It made a 1 km traverse through parallel corridors, collecting data while under manual control (Figure 2.5). Our ground truth for this dataset may have error on the order of 4 m as it was derived from measurements on a paper mine map.

Groundhog continuously collected 2D laser scan data in the horizontal plane. Every 3-4 m, or about every 20-60 seconds, it stopped and used a nodding platform to pitch the horizontal laser scanner through 60 degrees (Figure 2.6) to collect a 3D scan of about 70 k ranges. Groundhog's attitude information came from both a Crossbow IMU400 and a KVH yaw gyro. While the roll and pitch data were satisfactory, we were able to significantly improve the Groundhog's dead reckoning by estimating the gyro's yaw bias while the robot was stopped to collect the 3D scans. This simple strategy yielded heading with negligible drift over the course of the traverse.

In the report on the original Groundhog deployment, sequential pairwise ICP alignment of the 3D scans drifted approximately 6.5 m for every 100 m of linear traverse, and iterative ICP alignment with all nearby 3D scans yielded a drift of approximately 1.5 m every 100 m, after several days of processing in Matlab [Whittaker and Thayer, 2005]. As we were able to effect

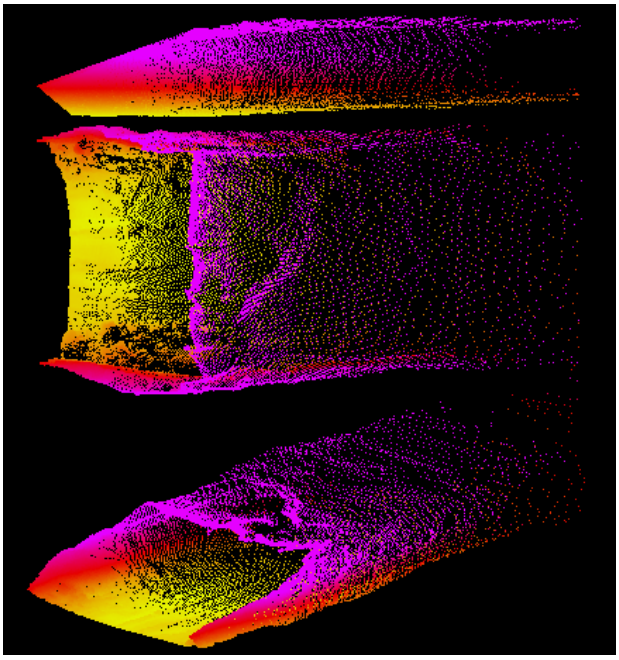

Figure 2.6: Side, top, and isometric views of a single 3D scan of a tunnel in Dakota Mine No. 2, collected by nodding (pitching) a horizontal laser scanner through 60 degrees.

such a dramatic improvement in the dead-reckoning (as described above), we considered that it was necessary to re-run ICP in order to make a fair comparison with icLK.

For the Dakota Mine dataset, we compared ICP with the regional version of icLK, which drew 10 regions from a distribution weighted to include the tunnel walls. The distances from the ground-truth end point, as well as the computation times for the various approaches are given in Table 2.4. Both icLK and our new sequential pairwise ICP yielded position drift of less than 1 m per 100 m – a significant improvement over the previous result, and comparable to the full iterative optimization originally reported [Whittaker and Thayer, 2005]. We believe this excellent performance is related to our improved gyro heading estimate. From examination of the matches, it appears that the most significant component of the dead reckoning error is the velocity estimate. Unfortunately, this leads to displacements along the tunnel that can be difficult to detect in straight sections (Figure 2.5). As a result, most of the error for both the ICP and the LK methods was along the axis of the long, straight traverse.

Though they produce substantially the same trajectory and final position error, 3D LK significantly outperforms ICP, running 10-20 times faster, depending on value of the approximate nearest neighbor parameter, ϵ. Evidence grid map construction and ICP point cloud projection time are not included in the processing time, however they were small: 0.17 s and 0.08 s respectively for a typical 83 k range scan. As the maximum usable range of the laser scanner was about 30m, the maps were 32 m cubed at 0.5 m resolution, and the matched regions were 4 m on a side. Processing was done on an Intel P4 3 GHz.

Method	Time (s)	Error (m)
DR		61.5
ICP $\epsilon = 0$	2030	7.5
ICP $\epsilon = 1$	1131	12.5
10 region icLK	97	6.7

Table 2.4: Summary of results for Groundhog's 1 km Dakota Mine traverse consisting of 265 3D scans with approximately 75 k ranges per scan.

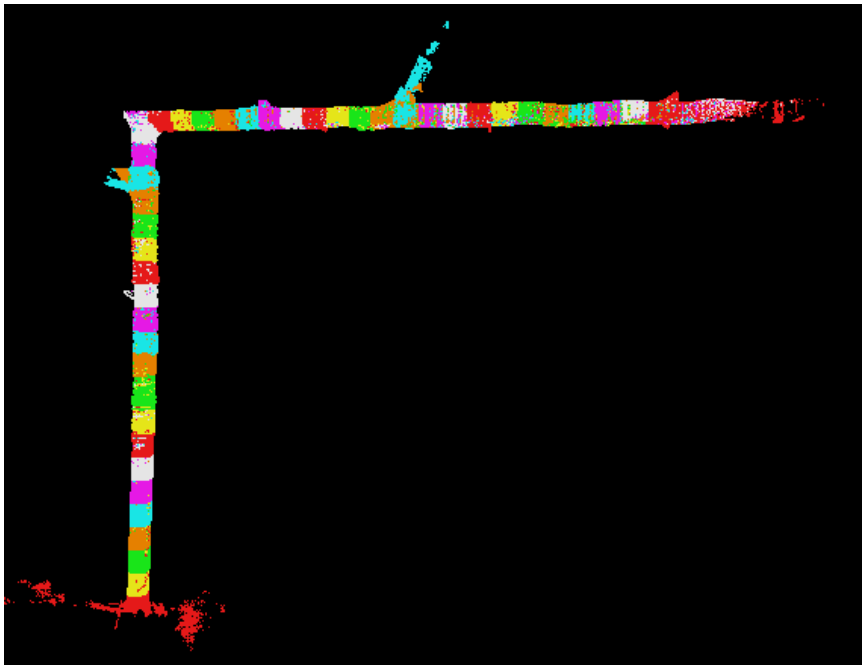

Figure 2.7: The Bruceton mine dataset, showing the 42 laser range scans as point clouds of about 100 k points each, each scan in a different color. Each leg of the L is about 100 m long.

2.7.2 icLK with Heading Error – Cave Crawler in Bruceton Mine

In this second experiment, we compare icLK and ICP using a more challenging dataset from the Bruceton mine that includes significant heading error. Cave Crawler, the second generation robotic subterranean mapping vehicle, uses spinning, rather than nodding, SICK laser range finders. It was deployed in the Bruceton Research Mine in December 2006, and traversed an L-shaped corridor for a total distance of about 200 m (Figure 2.7). Every few meters the vehicle stopped moving and collected a scan of the tunnel. At the same time, a Total Station was used to survey four points on the vehicle, giving highly accurate ground truth in all 6DOF for each scan origin.

For the Bruceton Mine dataset, we compared ICP with both icLK and isodox ICP. Cave Crawler collected so many points per scan that we had to decimate the point clouds by a factor of 5 to get them down to about 100 k points for ICP. Unlike Groundhog, Cave Crawler does not have a KVH gyro, and although we used the same heading bias estimation technique as before its

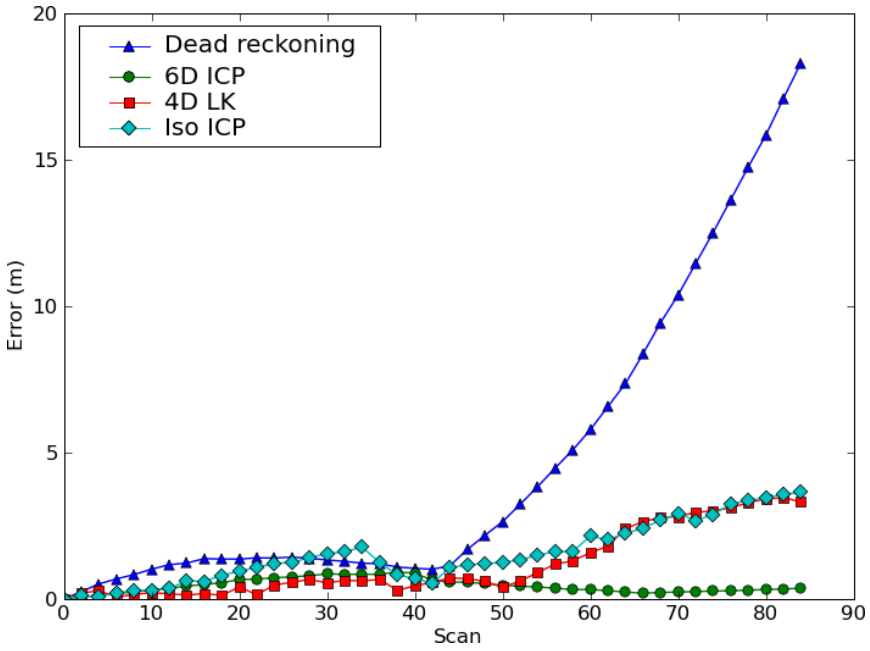

Figure 2.8: Position error for the Bruceton Mine dataset, comparing several different matching methods. The trajectories were rotated in order to minimize the ICP error, since we believe that error to be due to a misalignment between the survey and laser headings.

heading estimate was much worse. For the evidence grid-based matching techniques to address this error we needed to raise the angular resolution of the maps, so we constructed 32 m evidence grids with 0.125 m resolution for each scan, and used a single large (8 m) matching region for icLK and isodox ICP, rather than multiple small regions.

After examining the ICP match, we believe that there is a 4 degree misalignment between the surveyed heading and the laser scan heading. Thus, we rotated all the matched trajectories so as to minimize the ICP heading error in Figure 2.8. Run times and final error are summarized in Table 2.5. It is clear that the single 64^3 voxel region is much slower to match than the 10 smaller, lower resolution 16^3 voxel regions used for Groundhog. We found that the higher resolution and larger single region were absolutely necessary in the face of Cave Crawler's significant heading error – where Groundhog had basically no heading error. Both isodox ICP and icLK had similar errors, which we think is largely due to this heading error and the fact that the evidence grid-based methods necessarily involve discretization. In fact, we can try to predict the effect of this discretization: if take the arctangent of the width of a single voxel (0.125 m) over the scale of the entire matched region (8 m), we see that a match could be off by one degree due to discretization. Multiplying the total distance traveled (200 m) by the tangent of 1 degree, we have an predicted error of 3.5 m, quite close to the actual value.

2.7.3 Map Experimental Summary

We have developed the Deferred Reference Count Octree, an efficient map structure for storing and manipulating 3D evidence grids. The volumetric sparsity of the octree representation,

Method	Time (s)	Error (m)
DR		18.3
ICP $\epsilon = 0.2$	214	0.4
icLK	82	3.3
isodox ICP	60	3.6

Table 2.5: Summary of results for Cave Crawler's 200 m Bruceton Mine traverse consisting of 42 3D scans with approximately 100 k ranges per scan.

together with update efficiency yielded by copy-on-write are the key innovations that allow the real-time implementation of a Rao-Blackwellized particle filter for 3D SLAM using range data, even sparse, low-resolution range data.

One of the most important parameters for an evidence grid is the voxel resolution. In our experiments, the voxel resolution varies depending on the task and sensor. For the AUVs, which generally operate far from obstacles and use sonars, the voxel resolution was generally 0.5 or 1.0 m. For the subterranean robots, which operated in much more constrained environments and used highly accurate lidars, the voxel resolution was usually 0.125 m.

2.8 Summary

We have developed the Deferred Reference Count Octree, an efficient map structure for storing and manipulating 3D evidence grids. This is the key innovation that allows the efficient implementation of a Rao-Blackwellized particle filter for 3D SLAM using range data, even sparse, low-resolution range data, as will be described in Section 3. An issue that will also be addressed in Section 3 is precisely how the robot divides, or segments, the world up into sub-maps.

We have demonstrated methods for quickly and accurately aligning 3D evidence maps using a variety of methods including a 3D extension of Lucas-Kanade matching. We have shown that, depending on the number of ranges used, this method can be faster than traditional ICP-based methods and yield comparable accuracy. We also believe that evidence-grid based techniques are more robust, in that the evidence grids can incorporate an unlimited number of ranges, and inherently filter out noisy ranges, without the need for special techniques for decimating the point cloud or tuning the ICP point selection method. However, the discretization of the evidence grid can lead to heading error, as we describe above.

We are exploring better heuristics for selecting matching regions, as well as robust techniques for combining the corrections from multiple matches. We are looking to robust methods, for example the RANSAC algorithm of Fischler and Bolles [1981], to incorporate in the matching step.

The DRCO map representation, map matching methods, and map entropy estimators described in this chapter are the building blocks for the approaches described in the following chapters.

Chapter 3

Simultaneous Localization and Mapping

3.1 Introduction

Simultaneous Localization and Mapping (SLAM) is the task of building a map of the environment from sensor data and simultaneously using that map to localize, or estimate the robot's trajectory. In most cases the robot has various sensors to measure its own motion and sense its local surroundings. This sensor data is inevitably noisy, and must be appropriately filtered as part of SLAM.

There are a number of different methods which are used to perform SLAM, the most common being 2D approaches based on features extracted from sensor data and the well-known Extended Kalman Filter. As we have discussed in Chapter 1, we believe that a general SLAM solution must support 3D and should not rely on any particular set of features.

In this chapter, we present a featureless SLAM method based on the particle filter, another standard approach that is often used when certain requirements of the Kalman Filter formulation (unimodal position distributions, feature detection) cannot be satisfied. Particle filters, which are a sequential Monte Carlo method, maintain a large set of samples that can be used to approximate arbitrary distributions. Particle filters are robust, real-time approximations to the Bayesian formulation of SLAM.

In the most general SLAM case, particle filters would have to sample over the entire state space of the vehicle. As long as this state space is fairly low dimensional, as in the case of estimating the 3D position of a vehicle, a few hundred samples will adequately represent the true distribution. But for SLAM, the state space of the vehicle is both the vehicle position *and* the map. This is a very high dimensional space, so we can't possibly apply the basic particle filter: each particle would be a sample from the space of all possible maps as well as the vehicle position within that map.

The solution is to factor the state space into two parts: the vehicle position, and the map. The key to this factorized particle filter, known as a Rao-Blackwellized particle filter (RBPF), is that if we know the vehicle position at each timestep, then we can estimate the map using a closed form solution, such as a Kalman filter or an evidence grid. Of course, the vehicle position is *not* known at each timestep, rather the particles represent hypotheses, or guesses, of the true trajectory (and statistically, at least one is very close to the truth). In our case, we use use Deferred Reference

Count Octree evidence grids to represent the maps that result as a combination of the sensor measurements and the hypothesized trajectories (see Chapter 2 for a full description of our map structure).

A fundamental limitation of the RBPF approach called particle depletion limits the scale of loops that can be reliably closed by an RBPF, as a function of the number of particles. We investigate methods for opportunistically using as many particles as possible in real-time, and since several of the RBPF steps depend on ray-tracing operations, the algorithm can use more particles when the vehicle is in enclosed spaces with shorter ranges.

The main contribution of this chapter is our Rao-Blackwellized particle filter with DRCO maps for real-time SLAM in 3D environments. We begin with a derivation of our filter, briefly discuss methods for estimating the uncertainty in the SLAM solution, and then conclude with results from running SLAM in large-scale underwater environments.

3.2 Related Work in SLAM

Bayesian filtering provides the underlying probabilistic framework for recursively estimating the state of a dynamical system. Most of the methods described below, including Kalman filters and particle filters rely on a Bayesian foundation.

The first SLAM methods depended on the extraction of features from sensor data, and combined motion controls and feature observations with an Extended Kalman Filter (EKF) [Smith et al., 1988] [Smith et al., 1990], with early implementations and results by Moutarlier and Chatila [1989], who segmented line segment features from laser scans, and Leonard and Durrant-Whyte [1991], who used sonar to find unique geometric features. The EKF linearizes the motion and sensor models, which may be a problem in highly nonlinear sitations, for example with range-only beacons that form a ring of probability [Djugash and Singh, 2008]. A major advantage of the EKF is that it maintains a complete covariance matrix and mean vector for all of the features, which is important for the data association problem (recognizing that two observations are of the same feature). On the other hand, updating this full covariance is computational prohibitive, requiring $O(n^2)$, where n is the number of features, which limits the application of EKF SLAM to environments with a few hundred features.

There has been an enormous amount of work in finding efficient methods for large-scale SLAM. A taxonomy of these approaches can be attempted according to their map representation or estimation algorithm (see [Thrun, 2002] for a slightly outdated classification), but the fundamental property of all approaches is how they cope with spatial sparsity or spatial independence. In this related work section we consider methods which implicitly exploit sparsity, and delay discussion of methods which explicitly exploit sparsity by dividing the world into submaps until Chapter 4 – although the distinction can be blurry.

Avoiding the issue of feature detection, Lu and Milios [1997] proposed a featureless (and non-Bayesian) approach that uses odometry measurements and raw scan data to construct a sparse network of constraints between vehicle poses, which can be solved with batch-iterative nonlinear optimization techniques. This batch method was extended by Gutmann and Konolige [1999] to allow incremental updates to the constraint network, including loop closures. Duckett et al. [2002] used relaxation methods to yeild an interative algorithm with $O(n^2)$ updates, except

for loop closures which required $O(n^3)$. The approach was improved by Frese et al. [2005], who used multilevel relaxation methods to perform updates to the network in $O(n^2)$ time, even for loops. Hähnel et al. [2003] improves SLAM performance by doing lazy data association. They solve for maximum likelihood data association while tracking other association hypotheses in a tree that is expanded along a frontier of equal likelihood, allowing them to swap to alternate data associations if the map becomes inconsistent.

The Sparse Extended Information Filter (SEIF) of Thrun et al. [2004] is a dual formulation of the EKF called the Extended Information Filter (EIF) that maintains the information matrix (the inverse of the covariance matrix), which is naturally nearly sparse (as proved by Frese [2005]). SEIF methods encourage this sparseness by pruning small off-diagonal elements in the information matrix, which can lead to inconsistent maps [Eustice et al., 2005c]. In the information formulation, updates are constant time, but recovering the state (mean and covariance) is complicated, so approximations are used to do partial reconstructions, for example for data association, which tend to be overconfident. GraphSLAM [Thrun and Montemerlo, 2005] is an offline algorithm that bridges between offline optimization methods and EIF methods by building a graph of information constraints, which are efficiently reducing via variable elimination and then solving with nonlinear optimization.

Also using the information formulation, Paskin [2002] showed how to represent the SLAM belief state as a thin junction tree that requires linear space, and almost near time. A specialized thinning operation called variable contraction is applied periodically to keep the solution sparse and computationally efficient. Demonstrating the thin line between sparse methods and submap methods, we consider the closely related Treemap Frese [2006] to be a submap method, to be discussed in the next chapter.

Exactly Sparse methods, which include the Exactly Sparse Delayed State Filter [Eustice et al., 2005a] and the Exactly Sparse EIF [Walter et al., 2007] maintain sparse information matrices without the need to artificially prune elements, and therefore do not have the tendency towards overconfidence. Square Root Smoothing and Mapping (SAM) [Dellaert and Kaess, 2006] is also an information formulation of SLAM that exploits spatial sparsity, using Cholesky or QR decomposition to take the square root of the information matrix to recover the optimal (maximum a posteriori) state, but since the smoothing is performed over the entire trajectory computation complexity grows without bound over time.

Recent work has shown promising results from integrated non-parametric Gaussian process regression models with several types of Bayesian filters Ferris et al. [2006], Ko and Fox [2008].

As an alternative to the Gaussian-based formulations, particle filters are a sequential Monte Carlo method that uses a large set of samples, or particles, to represent a continuous distribution, such as the SLAM state. Particle filters become increasingly accurate as the number of particles increase. Particle filters provide a proven implementation of Bayesian filtering for systems whose belief state, process noise, and sensor noise are modeled by non-linear probability density functions (for a summary of particle filters, see Arulampalam et al. [2002]).

Our approach to SLAM is based on the Rao-Blackwellized Particle Filter (RBPF) introduced by Murphy [1999]. Each RBPF particle represents a hypothesis about both the vehicle pose and the map. FastSLAM [Montemerlo et al., 2002] was one of the first applications of this idea, using independent Kalman filters to estimate the position of each landmark. Like us, Stachniss [2006] and Eliazar and Parr [2006] extended Rao-Blackwellization to use evidence grid-based

data structures, but only in 2D and without the advantage of the spatial and temporal sparsity yielded by our Deferred Reference Count Octree (DRCO) map, described in Chapter 2. Hähnel et al. [2003] and Grisetti et al. [2005] show methods for improving the performance of FastSLAM and evidence-grid RBPFs respectively by modifying the prediction and resampling steps of the particle filter.

Particle depletion Arulampalam et al. [2002] is the limiting factor on the scale of RBPF SLAM. Interestingly, a careful analysis of the linearizations inherent in the venerable EKF SLAM approach show that there is similar max loop limitation [Bailey et al., 2006] – although the Unscented Kalman Filter has been shown to be more robust [Martinez-Cantin and Castellanos, 2005]. Martinez-Cantin et al. [2007] likewise propose methods to improve particle filter robustness to particle depletion. To detect particle depletion, Fox [2003] describes Kullback-Leibler distance (KLD) sampling, which estimates the number of samples needed at each iteration such that the error between the true density distribution and the discrete particle filter approximation is below some bound. This approach is applied to the bathymetry-map localization mentioned above by Bachmann and Williams [2003]. In short, they conclude that too few particles will poorly approximate the true posterior and too many particles take too long to process (so that measurements have to be thrown away).

To summarize the position of our approach within the large SLAM field, it is a constant-time algorithm based on the robust statistical properties of the RBPF. We do not use features: our map representation is the DRCO, which exploits the spatial sparsity of many environments, as well as the fact that particle maps are usually very similar. This allows us to perform SLAM in large 3D environments with arbitrary (though static) geometry, using sparse and noisy range sensors. As with all RBPFs, we are susceptible to particle depletion, which ultimately limits the scale of our algorithm to a few hundred meters, although we show that opportunistically using more particles ameliorates the problem.

3.3 Models

3.3.1 Gaussian Models

Many of the characteristics of a SLAM system are determined by the underlying vehicle motion and sensor models. Due to their attractive mathematical properties (primarily their simplicity), Gaussian models are a popular choice for representing the noise or uncertainty in taking actions or making measurements.

If the vector random variable y has a multivariate Gaussian, or normal, distribution $y \sim N(\mu, \Sigma)$ with mean μ and covariance Σ, then the probability density function (pdf) is

$$f(y) = \frac{1}{|\Sigma|^{1/2}(2\pi)^{N/2}} \exp\left(-\frac{1}{2}(y-\mu)^\top \Sigma^{-1}(y-\mu)\right) \tag{3.3.1}$$

where $|\Sigma|$ is the determinant of Σ.

When motion and measurement models assume an error model with a Gaussian distribution, they are called Gaussian models. Thus, for vehicle state x_t and a motion u_t, the motion model

from one time step to the next is

$$p(x_t|x_{t-1}, u_t) \tag{3.3.2}$$

or, to draw a sample from this distribution, we can equivalently write

$$x_t = h_t(x_{t-1}, u_t) + r_t \tag{3.3.3}$$

where $h(x_{t-1}, u_t)$ is, for example, the dead-reckoned position update that depends on the previous state and the navigation update u_t (derived from wheel encoders, etc), and $r_t \sim N(0, \Sigma_{x_t})$. The covariance Σ_{x_t} is indexed by time because it may vary over time depending on the state of the nav sensors or the environment.

Note that this assumes that the motion model is a *Markov*, or memoryless, process: the probability of the current state x_i given the previous state x_{i-1} is independent of all earlier states.

Likewise for the a range value z_t, the measurement model is

$$p(z_t|x_t, \Theta) \tag{3.3.4}$$

or, for generating samples from this distribution

$$z_t = g_t(x_t, \Theta) + v_t \tag{3.3.5}$$

where $v_t \sim N(0, \Sigma_{z_t})$. Note that unlike feature-based approaches, this measurement model doesn't use any data associations: the fusion of multiple range measurements is handled by the evidence grid map representation.

3.3.2 Vehicle Motion Model

We briefly describe a general motion model for several different types of vehicles. The sensor models and their interaction with the evidence grid map have already been discussed in Section 2.3.2.

The vehicle state vector x of the vehicle is the static 6 DOF vehicle pose q, together with its first and second derivatives:

$$q = [\text{roll, pitch, yaw, north, east, down}] \tag{3.3.6}$$

$$x = [q, \dot{q}, \ddot{q}] \tag{3.3.7}$$

As described above, this state vector is updated according to samples from the vehicle model

$$x_t = h(x_{t-1}, u_t) + r_t \tag{3.3.8}$$

where u is the motion update vector, in this case the measurements from the navigation sensors, and $r_t \sim N(0, \Sigma_{x_t}$ is the corresponding Gaussian noise for each measurement. For example, if a GPS position fix is available, then the x and y fields of u are set to the fix position, and the corresponding fields of Σ_{x_t} are set to 15 m, or the standard deviation of the GPS fix.

The model synthesizes quantities that haven't been measured. Thus if there is no measurement of position, then the vehicle model will attempt to integrate the velocity measurements, and if there is no measurement of velocity, it will integrate the accelerations. This vehicle model, and particularly the way that measurements are combined, are a simple approximation to a Kalman filter: if there were ever multiple sources of information for a particular measurement it would be better to combine them with a Kalman filter and then generate a sample for the additive error according to the resulting covariance matrix.

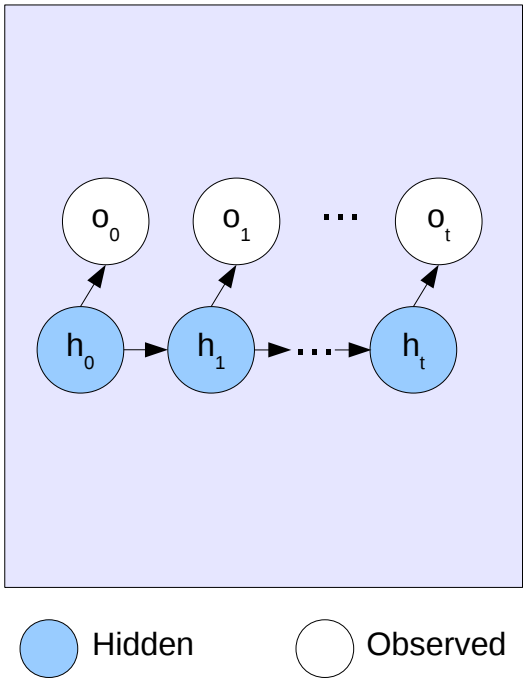

Figure 3.1: A generic Bayes filter graphical model, in which the observations o_i are dependent on the hidden states h_i.

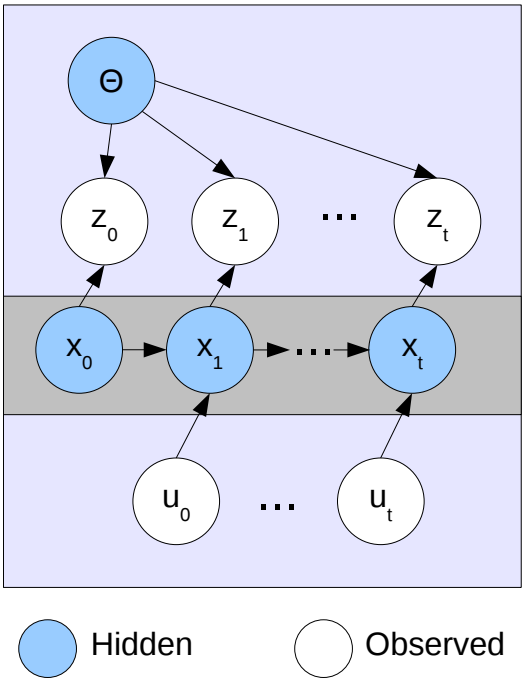

Figure 3.2: SLAM graphical model, showing how the hidden state can be factored into the map Θ and the vehicle pose x_i.

3.4 SLAM Derivation

The goal of a SLAM system is to estimate the probability distribution at time t over all possible vehicle states x and world maps Θ using all previous sensor measurements $\mathbf{z}_t$ and navigation updates $\mathbf{u}_t$ (for a complete list of notation, see Table 3.1):

$$p(x, \Theta | \mathbf{z}_t, \mathbf{z}_t) \tag{3.4.1}$$

This distribution is called the *SLAM posterior*. The recursive Bayesian filter formulation of the SLAM problem is straightforward to derive, but is usually computationally intractable to solve in closed form.

3.4.1 Bayes Filter

If we have a general state space model with hidden variables $\mathbf{h}_t = \{h_0, h_1, \ldots, h_t\}$ and observed variables $\mathbf{o}_t = \{o_1, o_2, \ldots, o_3]\}$, we would like to be able to perform inference on the hidden variables. Given the observed variables, Bayesian inference on the hidden variables relies on the joint posterior distribution $p(\mathbf{h}_t|\mathbf{o}_t)$. The corresponding Bayesian belief network, or graphical model, is show in Figure 3.1. Assuming that the hidden variables have some initial distribution $p_0(h_0)$ and a transition model $p(h_t|h_{t-1})$, and that the observations are conditionally independent

given the hidden process (yielding the marginal distribution $p(o_t|h_t)$), a Bayesian filter can be used to derive a recursive expression for this posterior

$$p(\mathbf{h}_t|\mathbf{o}_t) = p(\mathbf{h}_{t-1}|\mathbf{o}_{t-1})\frac{p(o_t|h_t)p(h|h_{t-1})}{p(o_t|\mathbf{o}_{t-1})}, \tag{3.4.2}$$

which shows how the posterior can be updated using observation and transition models.

To derive this recursive expression for the SLAM posterior, which can be described using the Bayesian belief network shown in Figure 3.2, the vehicle trajectory and map are hidden variables $\mathbf{h}_t = \{\mathbf{x_t}, \Theta_t\}$, and the sensor measurements and motion updates (which may be from a control or from navigation sensors) are observed variables $\mathbf{o}_t = \{\mathbf{z}_t, \mathbf{u}_t\}$, which yields the SLAM posterior equation 3.4.1. As a first step, we apply Bayes rule:

$$p(x_t, \Theta|\mathbf{z}_t, \mathbf{u}_t) = \eta\ p(z_t|x_t, \Theta, \mathbf{z}_{t-1}, \mathbf{u}_t)\ p(x_t, \Theta|\mathbf{z}_{t-1}, \mathbf{u}_t) \tag{3.4.3}$$

Where η is the normalizing constant from Bayes rule.

Following Montemerlo et al. [2002]'s derivation of FastSLAM, we can simplify this expression by exploiting the observation that the measurements z_t are independent of prior measurements $\mathbf{z}_{t-1}$ and the motion updates $\mathbf{u}_t$ given the pose x_t and the map Θ:

$$= \eta\ p(z_t|x_t, \Theta)\ p(x_t, \Theta|\mathbf{z}_{t-1}, \mathbf{u}_t) \tag{3.4.4}$$

We can condition the right term on the prior pose by integrating over all poses at time $t-1$ by using the Total Probability Theorem $p(x) = \int p(x|y)p(y)dy$

$$= \eta\ p(z_t|x_t, \Theta) \int p(x_t, \Theta|x_{t-1}, \mathbf{z}_{t-1}, \mathbf{u}_t)\ p(x_{t-1}|\mathbf{z}_{t-1}, \mathbf{u}_t)dx_{t-1} \tag{3.4.5}$$

Then expand the leftmost term inside the integral using the definition of conditional probability:

$$= \eta\ p(z_t|x_t, \Theta) \int p(x_t|\Theta, x_{t-1}, \mathbf{z}_{t-1}, \mathbf{u}_t)p(\Theta|x_{t-1}, \mathbf{z}_{t-1}, \mathbf{u}_t)p(x_{t-1}|\mathbf{z}_{t-1}, \mathbf{u}_t)dx_{t-1} \tag{3.4.6}$$

And simplifying the first term inside the integral by observing that according to our motion model, x_t is only a function of x_{t-1} and u_t:

$$= \eta\ p(z_t|x_t, \Theta) \int p(x_t|x_{t-1}, u_t)\ p(\Theta|x_{t-1}, \mathbf{z}_{t-1}, \mathbf{u}_t)p(x_{t-1}|\mathbf{z}_{t-1}, \mathbf{u}_t)dx_{t-1} \tag{3.4.7}$$

Now combining the rightmost terms of the integral (again using definition of conditional):

$$= \eta\ p(z_t|x_t, \Theta) \int p(x_t|x_{t-1}, u_t)\ p(x_{t-1}, \Theta|\mathbf{z}_{t-1}, \mathbf{u}_t)dx_{t-1} \tag{3.4.8}$$

We can drop the latest motion update u_t, leaving $\mathbf{u}_{t-1}$, since it doesn't provide any new information about x_{t-1} without z_t.

At last we have a recursive formula for computing the SLAM posterior at time t given the SLAM posterior at time $t-1$:

$$\underbrace{p(x_t, \Theta|\mathbf{z}_t, \mathbf{u}_t)}_{\text{new posterior}} = \eta\ \underbrace{p(z_t|x_t, \Theta)}_{\text{meas. model}} \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{motion model}}\ \underbrace{p(x_{t-1}, \Theta|\mathbf{z}_{t-1}, \mathbf{u}_{t-1})}_{\text{old posterior}} dx_{t-1} \tag{3.4.9}$$

$\#_x$	number of particles
$\#_z$	number of simultaneous range measurements
$x_t^{(m)}$	vehicle state of the m-th particle at time t
$\mathbf{x}_t^{(m)}$	trajectory of m-th particle from time 0 to t $= \{s_0^{(m)}, s_1^{(m)}, s_2^{(m)}, \ldots, s_t^{(m)}\}$
$z_t^{(n)}$	a particular range measurement n at time t
z_t	all range measurements at time t
$\mathbf{z}_t$	history of measurements from time 0 to t $= \{z_0, z_1, z_2, \ldots, z_t\}$
u_t	vehicle dead-reckoned innovation at time t
$\mathbf{u}_t$	history of dead-reckoning from time 0 to t $= \{u_0, u_1, u_2, \ldots, u_t\}$
$\Theta^{(m)}$	map of m-th particle
$w_t^{(m)}$	weight of m-th particle at time t
$h(x_{t-1}, u_t) + r_t$	motion model $= p(x_t \mid u_t, x_{t-1}), r_t \sim N(0, \Sigma_{x_t})$
$g(s_t, \Theta) + v_t$	measurement model $= p(z_t \mid x_t, \Theta), v_t \sim N(0, \Sigma_{z_t})$

Table 3.1: Particle filter notation.

which indeed has the form of the recursive Bayes filter equation 3.4.2.

Unfortunately, the integral over all previous vehicle positions is usually intractable to compute and must be approximated. The most popular approximations involve the use of Kalman filters or particle filters, which is the approach that we discuss next.

3.4.2 Particle Filter

Particle filters are a sequential Monte Carlo approximation to the recursive Bayesian filter. In particular, particle filters provide an implementation of Bayesian filtering for systems whose belief state, process noise, and sensor noise are modeled by non-parametric probability density functions Arulampalam et al. [2002].

In our case, the particle filter maintains a discrete approximation of the SLAM posterior using a (large) set of samples, or *particles*, which include the entire SLAM state space: both position and map $s = \{x_t, \Theta\}$. Then a weighted approximation to the recursive Bayesian SLAM filter is:

$$p(s_t \mid \mathbf{z}_t, \mathbf{u}_t) \approx \sum_{i=1}^{\#_s} w^{(i)} s_t^{(i)} \tag{3.4.10}$$

where we have a set of $\#_s$ samples : $[s^{(1)}, s^{(2)}, \ldots, s^{(\#_s)}]$, with weights $[w^{(1)}, w^{(2)}, \ldots, w^{(\#_s)}]$.

Assuming that we have such a set of weights and particles, we can update the particles by drawing samples from $q()$:

$$s_t^{(i)} \sim q(s_t \mid s_{t-1}^{(i)}) \tag{3.4.11}$$

where $q(x_t|x_{t-1}, z_t)$ is a distribution that is easy to draw samples from, and is an approximation to $p(x_t|x_{t-1}, z_t)$ that has fatter tails, assuring that when we draw samples from $q()$, we'll get coverage of $p()$.

In turn, the weights are updated as

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(z_t|s_t^{(i)})p(s_t^{(i)}|s_{t-1}^{(i)})}{q(s_t^{(i)}|s_{t-1}^{(i)}, z_t)} \tag{3.4.12}$$

In the optimal case, $q()_* = p()$, but since its not easy to sample from that distribution, we can instead use the motion model $p(x_t|x_{t-1})$, which is a normal distribution sub-optimal approximation that has the great advantage of allowing us to simplify the weight update equation to

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t|s_t^{(i)}) \tag{3.4.13}$$

An issue that is illustatrated by our use of s_t to wrap $\{x_t, \Theta\}$ above is that particle filters sample over the entire state space of the vehicle. As long as this state space is fairly low dimensional, as in the case of estimating the 3D position of a vehicle, a few hundred samples will adequately represent the true distribution. But for SLAM, the state space of the vehicle is both the vehicle position *and* the map. This is a very high dimensional space: each particle would be a sample from all possible maps and all possible positions within that map. Rao-Blackwellized Particle Filters Doucet et al. [2000] address this problem by maintaining the map portion of the state in a computationally efficient closed form (in our case, the evidence grid). Thus, our particles are composed of both a pose and the map which has been constructed according to the particle's trajectory and the range sensor data.

3.4.3 Rao-Blackwellized Particle Filter

The key insight of Murphy [1999] is that the SLAM posterior distribution can be factored into two parts, or marginals: the path distribution and the map distribution. Furthermore, knowing the vehicle's trajectory $\mathbf{x}_t$ makes the observations $\mathbf{u}_t$ conditionally independent, so that the map sample Θ can be computed in a closed form. The process of factoring a distribution such that one part can be computed analytically is known as Rao-Blackwell factorization Doucet et al. [2000]. As a result, following Montemerlo et al. [2002] we express the posterior over *trajectories*, factoring the distribution as

$$p(s_t|o_t) = p(\mathbf{x}_t, \Theta|\mathbf{z}_t, \mathbf{u}_t) = p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{u}_t)\ p(\Theta|\mathbf{x}_t, \mathbf{z}_t). \tag{3.4.14}$$

In a Rao-Blackwellized particle filter, each particle represents both a sample $\mathbf{x}_t^{(i)}$ from the distribution of vehicle trajectories, and the sample map $\Theta^{(m)}$ which results from that trajectory combined with the sensor measurements $\mathbf{z}_t$. Since we update the particle maps at every time-step, they represent the combination of aensor measurements and vehicle trajectory – so each particle only needs to store the current map $\Theta^{(m)}$ and pose $x_t^{(m)}$ (rather than the whole trajectory $\mathbf{x}_t^{(m)}$).

For practical purposes, when SLAM is being used to provide a pose for the rest of the vehicle control software we need to turn the set particles into a single point estimate. If the posterior

distribution is Gaussian, then the mean is a good estimator, but other estimators may be better if the distribution becomes non-Gaussian – which is indicated by high position variance.

In implementation, the particle filter algorithm based on the Sampling Importance Resampling Filter of Gordon et al. [1993] has the following steps:

1. **Initialize** The particles start with their poses x_0 initialized according to some initial distribution and their maps Θ (possibly) containing some prior information about the world. This is called the *prior distribution*, and it may be very large if the initial position is uncertain.

2. **Predict** The dead-reckoned position innovation u_t is computed using the navigation sensors. A new position x_t is predicted for each particle using the vehicle motion model:

$$x_t = h(x_{t-1}, u_t) + r_t. \tag{3.4.15}$$

   which takes the available measurements and samples from the Gaussian noise model for each measurement. Under prediction alone, the particles will gradually disperse according to the navigation sensor error model, representing the gradual accumulation of dead reckoning error.

   This resulting distribution of the particles is called the *proposal distribution*.

3. **Weight** The weight w for each particle is computed using the measurement model and the range measurements (from $\#_z$ different ranges):

$$w = \eta \prod_{n=1}^{\#_z} p(z_t^n | x_t, \Theta), \tag{3.4.16}$$

   where η is some constant normalizing factor. In our implementation, the real range measurements z are compared to ray-traced ranges $\hat{z}$ using the particle pose and map. We compare the simulated and real ranges using the measurement model

$$z = g(x_t, u_t) + v_t, \tag{3.4.17}$$

   which is assumed to have a normal noise model (represented by v_t, so

$$p(z|x, \Theta) = \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{\frac{-(\hat{z}-z)^2}{2\sigma_z^2}}. \tag{3.4.18}$$

   Substituting into the expression for particle weight and taking the logarithm of both sides shows that maximizing this weight metric is very close to minimizing the intuitive sum squared error metric:

$$\log w = C - \frac{1}{2\sigma^2} \sum_{i=1}^{\#_z} (\hat{z} - z)^2, \tag{3.4.19}$$

   where $C = \#_z \times \log\left(\sqrt{2\pi\sigma^2}\right)$.

4. **Resample** The $O(n)$ algorithm described in Arulampalam et al. [2002] is used to resample the set of particles according to the weights w such that particles with low weights are likely to be discarded and particles with high weights are likely to be duplicated.

   Resampling too often can cause the particle cloud to collapse to a single point. A rule of thumb introduced by Doucet et al. [2000] based on a metric by Liu [1996] is used to decide whether or not to resample based on the effective number of particles:

$$\#_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}(w^{(i)})^2} \tag{3.4.20}$$

   so that resampling is only performed when the effective number of particles $\#_{\text{eff}}$ falls below half the number of particles, $N/2$.

   Another trick that is useful in slowing the convergence of the particle filter is to add noise to the resampled particle positions, effectively re-initializing the particle filter around the particles with high weights. The amount of resampling noise can be gradually reduced, as in simulated annealing. After resampling, the set of particles is a new estimate of the new SLAM posterior.

5. **Update** The measurements z are inserted into the particle maps $\Theta^{(m)}$ to update the map according to the sonar beam model of each measurement relative to the particle position. This is when maps must be copied and updated. We save duplicate insertions by inserting *before* copying successfully resampled particles. Note that in this paper, we are primarily testing localization, and do not update the map after it has been constructed.

6. **Estimate** Generate a position estimate from the particles.

7. **Repeat from Predict**

3.4.4 Map representation - 3D Evidence Grids

The main difficulty with using 3D evidence grids as the map representation for the RBPF arises from the cost of storing and manipulating large, high resolution maps. If the evidence log-odds are represented as single bytes (with values between -128 and 127), then an evidence grid 1024 cells on a side requires a megabyte of memory in 2D and a gigabyte in 3D. This is completely intractable when each Rao-Blackwellized particle has its own map that may need to be copied when the particle is resampled.

We have developed the Deferred Reference Count Octree (DRCO) described in depth in Section 2, that offers a compact and efficient octree-based data structure that is optimized for use with a particle filter. The main advantage of an octree is that the tree does not need to be fully instantiated if pointers are used for the links between octnodes and their children. Large contiguous portions of an evidence grid are either empty, occupied, or unknown, and can be efficiently represented by a single octnode – truncating all the children which would have the same value. As evidence accumulates, the octree can compact homogeneous regions that emerge, such as the large empty volume inside a chamber.

The efficiency of the DRCO + RBPF combination depends on the circumstances. It will be most efficient when the environment and particle filter are amenable to the exploitation of spatial locality (particles share most of the map in common when the vehicle is only modifying small regions) and volumetric sparsity, since octrees compactly represent a map that is mostly empty or full. Most large-scale outdoor environments seem to exhibit these properties.

3.4.5 Adaptive particle count

The processing time of a single iteration of the SLAM algorithm varies significantly depending on local world geometry. This is because of the range dependency of ray-tracing time, but the implication is that the particle filter could opportunistically use more particles and still maintain real-time performance. Furthermore, the weighting and resampling steps take approximately the same amount of time given a fixed number of particles. How to best spend the available computational resources: weighting more particles in the hopes of finding good ones, or resampling more particles in order to maintain particle diversity? Our approach is to weight as many particles as possible within a desired time limit (determined by the desired rate of the particle filter), but then to fully resample according to those weights. With DRCO's, resampling particles is fast. We modify the particle filter algorithm as follows:

1. Start with a huge number of particles (in our case 5000: more than could ever be supported computationally)
2. During the weighting step randomly pick particles and weight them until the time limit is reached, set all other particle weights to zero
3. Fully resample the particle set – particles which were not weighted will always be replaced with weighted particles

Note that it is important to randomly pick particles during the weighting phase to avoid sorting effects, such that the distribution is accurately sub-sampled.

3.5 SLAM Entropy

The entropy of the RBPF SLAM posterior distribution includes the entropy of both the particle poses $x^{(k)}$ and also of the analytic particle maps $\theta^{(k)}$: $H(p(x,\theta))$. As discussed in Section 2.6, the entropy of a single random variable probability density function is defined as

$$H(p(X)) = -\int_X p(x)\log(p(x))dx = -E_X[\log(p(x))], \tag{3.5.1}$$

while the expected entropy of X given some other random variable Y can be derived as

$$\begin{aligned} H(p(X|Y)) &= -E_{X,Y}[\log(p(x|y))] && (3.5.2)\\ &= -\int_{X,Y} p(x,y)\log(p(x|y)dxdy && (3.5.3)\\ &= -\int_{Y} p(y)\int_{X} p(x|y)\log(p(x|y))dxdy && (3.5.4)\\ &= -\int_{Y} p(y)H(p(X|y))dy && (3.5.5)\\ &= -E_{Y}[H(p(X|Y))] && (3.5.6) \end{aligned}$$

Using these definitions, we can derive the entropy of the two-variable SLAM posterior as

$$\begin{aligned} H(p(X,\Theta)) &= -E_{X,\Theta}[\log(p(x,\theta))] && (3.5.7)\\ &= -E_{X,\Theta}[\log(p(x)p(\theta|x))] && (3.5.8)\\ &= -E_{X,\Theta}[\log(p(x)) + \log(p(\theta|x))] && (3.5.9)\\ &= -E_{X}[\log(p(x))] - E_{X,\Theta}[\log(p(\theta|x))] && (3.5.10)\\ &= H(p(X)) + E_{X}[H(p(\Theta|X))] && (3.5.11) \end{aligned}$$

yielding the sum of the pose entropy and the expected entropy of the map conditioned on the pose. For the particle filter, we can approximate the entropy of the belief state by decomposing it into the entropy of the particle cloud and the average entropy of the particle maps.

$$H(p(x,\theta)) \approx H(p(X^{(i)})) + \sum_{i} w^{(i)} H(p(\theta^{(i)})) \tag{3.5.12}$$

As we have shown in Chapter 2.6, we have analytic methods for computing the entropy of individual octree evidence grid maps. Thus the expected map entropy can be computed as the (weighted) average of the entropies of the particle maps as in Stachniss [2006], though this may be very expensive.

Computing the entropy of the particle poses is approximated with heuristics. The method described by [Stachniss, 2006] approximates the cloud with a multivariate normal distribution, and then computes the differential entropy of the distribution:

$$H(p(X)) \approx H(\mathbf{MVN}(X)) = \frac{d}{2}(1 + \log 2\pi) + \frac{1}{2}\log|\Sigma| \tag{3.5.13}$$

where $|\Sigma|$ is the determinant of the covariance matrix.

This approximation is only accurate in cases in which the pose cloud is nearly unimodal, although it will usefully report high entropy for multimodal distributions. Roy and Thrun [1999] and Stachniss [2006] refine this heuristic by computing its average value over multiple points in time, though Blanco et al. [2008a] describes how even this improved metric can become undefined after closing a loop. A more mundane problem is that the entropy of the maps dominates the entropy of the poses, meaning that for example a small amount of exploration will result in a greater change in entropy than a major convergence in the particle poses.

Rather than decomposing the belief state, Blanco et al. [2008a] proposes using the entropy of the expected map (rather than the expected map entropy), which is approximated by constructing a new evidence grid as the weighted average of all the particle evidence grids (in a single reference frame), and then computing the entropy of the resulting blurry map:

$$H(p(X,\Theta)) \approx H\left(\sum_{k}^{\#_x} \theta^{(k)}\right). \tag{3.5.14}$$

While this method is conceptually simple and yields very useful entropy values (for evaluating loop closures, exploration, etc.), it requires constructing the average map, which is O(PV), with P particles and V voxels per map – and the blurry nature of the average map means that the efficiencies of the octree representation are diminished. Further, the averaged map will not distinguish between, for example, a strongly peaked bi-modal distribution and a weakly-peaked unimodal distribution.

The estimates of entropy for evidence grids discussed in the last chapter were simple and accurate, given the assumptions inherent to evidence grids. As this section has shown, the best entropy estimators for SLAM are very approximate. We will return to the issue of entropy in later chapters.

3.6 Underwater Robot

The DEep Phreatic THermal eXplorer (DEPTHX) hovering autonomous underwater vehicle, nicknamed Clementine, was assembled at the Stone Aerospace facility near Austin, TX. Our first integrated tests of control and localization were performed in an acoustic test tank at the Austin Research Laboratory in the fall of 2006. During the winter of 2006-2007, we began our first long range tests in a flooded quarry, also near Austin, and began integration with the science payload. Our first cautious forays into a cenote were in La Pilita during two field expeditions in February and March of 2007. La Pilita is only about 100 m deep and is very accesible for launching and recovering the vehicle, so it made an excellent testing site. Finally, in May 2007 we mounted the final expedition to explore Zacatón – at the very tail end of the expedition we were able to explore both Caracol and Verde in less than two days.

3.6.1 DEPTHX Vehicle

Clementine (Figure 3.3) has a full suite of underwater navigation sensors, including a Honeywell HG2001 Inertial Measurement Unit (IMU), two Paroscientific Digiquartz depth sensors, and an RDI Navigator 600 Doppler Velocity Log (DVL). There is also a Conductivity, Temperature, and Depth (CTD) sensor for measuring the speed of sound for corrections to sonar-based measurements. The specifications for the INS are roll/pitch: 0.2° 2σ, yaw: $0.4^\circ 2\sigma$, for the DVL velocities 0.3 cm/s 1σ, and for the depth sensors 0.01% of full range (10 cm for our 1000m rated sensor). In particular, Clementine's high-grade IMU has negligible yaw drift over time. Under certain circumstances, these sensors can provide dead-reckoned navigation on the order of 0.5% of distance traveled [Larsen, 2000]. Over the course of a ~ 4 hour 2km mission, this would yield

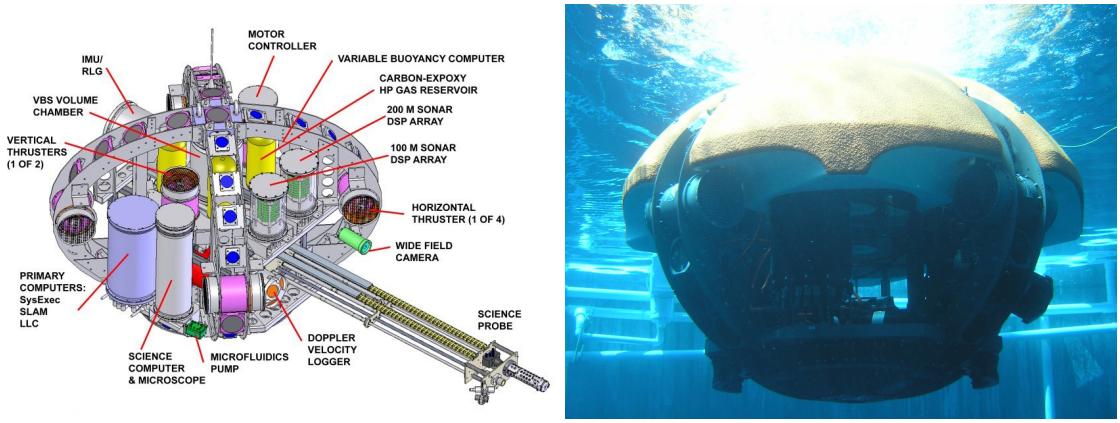

Figure 3.3: A schematic of Clementine's structure and components. Eleven pressure vessels house computing, batteries, sensors, and science instruments. Diameter is approximately 2m, weight 1.3 metric tons. ©Stone Aerospace, 2006. On the right, a photograph of Clementine in the ARL test tank, courtesy Marcus Gary.

an error of around 10m, which would be perfectly acceptable in open water conditions but is a serious concern within confined tunnels. Additionally, there are times when the DVL cannot provide velocity measurements, at which point the quality of the dead-reckoned solution degrades significantly. This makes a clear case for augmenting the dead reckoning system with a SLAM solution.

In addition to the standard dead-reckoning sensors, Clementine has a mapping system: an array of 54 pencil-beam sonars that provide a constellation of range measurements around the vehicle. The sonar array is in the shape of three great circles (Figure 3.3, for a comparison of sonar geometries see Fairfield et al. [2005]). This array geometry allows measurements from previously mapped regions while exploring new regions, which is vital for SLAM. The sonars have long ranges (some 100m and others 200m) but lack the resolution (5 cm), update rate (1 Hz), low noise (10 cm), and point density of a laser scanner, making feature recognition and association difficult.

Clementine's onboard SLAM computer was a 1.8 GHz Pentium M with 1 Gb of RAM.

3.6.2 Related Work in Underwater Localization

GPS does not work underwater, nor do other radio localization methods. A typical Autonomous Underwater Vehicle (AUV) uses a combination of depth sensors, inertial sensors, and Doppler velocity sensors to compute a dead-reckoned estimate of its position while at depth (for an overview of underwater navigation methods, see [Kinsey et al., 2006]). With high accuracy attitude and depth sensors the uncertainty in the AUV's 3D pose (roll, pitch, yaw, x, y, z) is primarily in x and y. Most underwater navigation systems are based on Kalman Filters which merge Doppler velocity and inertial measurements [Larsen, 2000]. Corrections to the unbounded drift error inherent in such systems have been achieved by using the global positioning system (GPS) while on the surface [Healey et al., 1998] or beacon-based long baseline (LBL) acoustic positioning systems [Whitcomb et al., 1999]. But frequently surfacing for GPS fixes may not be

possible or desirable, and LBL beacons, which are typically used in long-duration open-water operations, must be deployed and surveyed before use. Neither of these approaches is viable in Zacatón. SLAM offers an attractive method to bound dead-reckoning error because it allows the vehicle to be completely self-contained and unrestricted – and it yields a map of the environment.

In the underwater domain, sonar sensors are not capable of providing the resolution necessary to resolve and recognize features. There has been work on off-line SLAM methods using tunnel cross-sections, or slide images, which can be derived from sparse sonar ranges as long as the environment is tunnel-shaped [Bradley et al., 2004]. In the case where there are free floating artificial features, scanning sonars have been shown to have high enough resolution to support feature-based SLAM [Williams et al., 2000]. Alternatively, in clear water with good lighting, SLAM has been demonstrated via video mosaicing [Eustice et al., 2005b] and also a combination of vision-based feature detection and sonar [Williams and Mahon, 2004].

Underwater localization has also been demonstrated in cases where there is variation in the sea-floor and the vehicle has a prior map of the bathymetry [Williams, 2003, Leonard et al., 1998, Fairfield and Wettergreen, 2008]. Many underwater environments are characterized by large monotonous regions where there has been promising work with Synthetic Aperture Sonar (SAS) to support range-and-bearing SLAM [Newman et al., 2003b].

A successful underwater SLAM implementation using a Delayed-State Kalman Filter SLAM with multibeam submaps (using point clouds and ICP) has been shown by Roman [2005], who also analyzes map segmentation methods as described above. Eustice et al. [2005a] describe the Exactly Sparse Delayed-State Filter (ESDSF) SLAM by directly comparing "view-based" maps (camera images), which sit between explicit feature recognition and featureless maps.

3.7 Experiments

In this section we present experiments using our SLAM approach onboard Clementine. In the following sections, we describe results from controlled environments which allowed us to characterize the performance of the SLAM system before deploying Clementine in the unknown cenotes. We then follow the natural progression from dead reckoning, to mapping, to Localization with a prior map, to SLAM. To illustrate each stage, we present results from our test locations, starting in a quarry in Texas before moving to the cenotes La Pilita, Zacatón, Caracol, and Verde.

3.7.1 Preliminary Characterization – Controlled Test Sites

One of the drawbacks of exploring unexplored environments, such as the cenotes, is that it is truly dangerous to run experiments, and there is no ground truth with which to compare the results. In these preliminary experiments, we characterized the performance of our SLAM system in simple, safe, controlled environments, with ground truth, so that we could predict how it would behave when we sent Clementine into the cenotes.

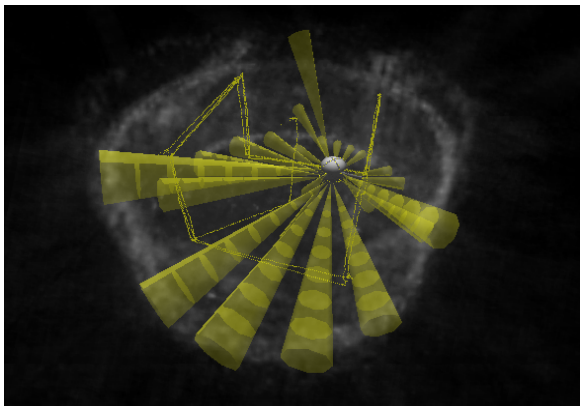

Figure 3.4: This figure shows the 3D trajectory of Clementine in the ARL test tank, as well as a rendering of the vehicle and its sonar beams. The vehicle is surrounded by the cloudy evidence map constructed by SLAM, where opacity indicates occupancy.

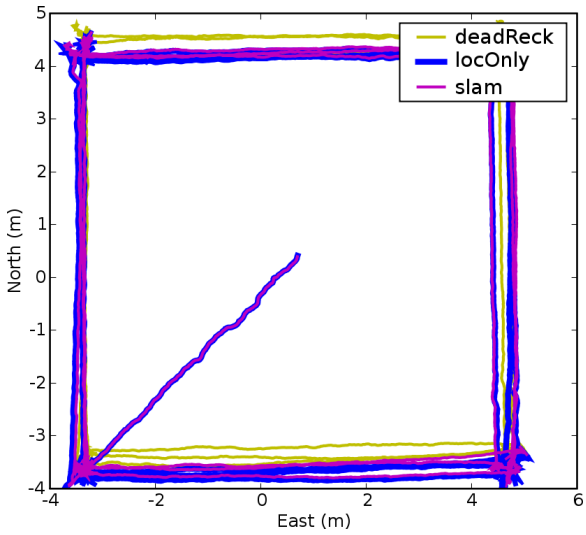

Figure 3.5: Planar XY view of the trajectories of the various localization solutions in the ARL test tank. The deadReck solution looks quite square as it was used to navigate during the test, but the true vehicle trajectory is shown by locOnly (localization with a prior map and 3000 particles).

Small-scale SLAM in the ARL Tank Test

The large wooden test tank at the University of Texas at Austin Applied Research Lab (ARL) is a cylinder 38 feet (11.6m) deep and 55 feet (16.8m) in diameter. To test SLAM in this environment, Clementine drove three cycles around a 3D box pattern (Figure 3.4), using dead-reckoning for localization and navigation. The box pattern was 8m on a side and 5m deep, and each cycle took about 13 minutes for a total run time of 40 minutes. The vehicle rotated $\sim 5^{\circ}/s$ during ascent and descent in order to obtain better sonar coverage of the walls.

To establish the ground truth trajectory of the vehicle, we localized with 3000 particles and a manually constructed 0.25m resolution prior map of the ARL tank. The dead-reckoned trajectory drifted from the ground-truth by $\sim$0.5m, which agreed with our observations during the test. We then ran SLAM using 500 particles (with no prior map), which yielded a bounded localization error of $\sim$0.1m (Figures 3.5 and 3.6). To characterize the performance of the particle filter and DRCO in a more challenging scenario, we turned to the synthetic Wakatón environment.

SLAM in the Simulated Wakatón Environment

In May 2005, the DEPTHX team lowered a 32-sonar probe, called the DropSonde, into Zacatón to a depth of 200m [Fairfield et al., 2006]. A similar probe, called the Digital Wall Mapper (DWM) was used in 1998-1999 to map several kilometers of the Wakulla Springs cave system in Florida [Stone et al., 2000]. The DWM was a diver-driven sensor sled, while the DropSonde was simply lowered on a cable from a barge. Both probes were based around the same hardware: a Honeywell HG2001 ring laser gyro IMU, two depth sensors, and a radial array of 32 pencil

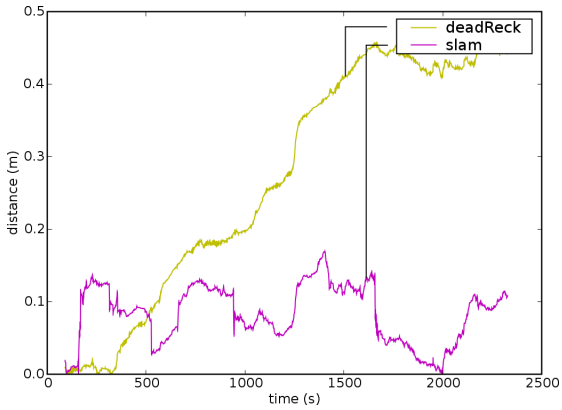

Figure 3.6: Distance between various localization solutions in the ARL test tank. The ground truth was established using localization with a prior map and 3000 particles – dead reckoning drifts away while SLAM error is bounded.

beam sonars.

The problem with using either of the DWM or DropSonde datasets to test our SLAM method was that neither dataset contained sonar data with a reasonable geometry for SLAM. In both cases, the data was collected by driving a ring of sonars along the axis of a cylindrical tunnel. In this orientation, the sonars provide no "look-back" to previously mapped regions.

Our solution is to create a synthetic world by building partial maps from the datasets and merging them together. In the case of the DropSonde data, we recorded all six degrees of freedom: the x,y position of the barge and the depth, roll, pitch, and heading of the probe. Together with the sonar data, this DropSonde pose data provided an excellent map of the first 200m of Zacatón. The Wakulla Springs data also contained excellent attitude, heading, and depth, but there was no ground truth for x and y position except for a few widely spaced waypoints. Using these waypoints to estimate the IMU drift rates, we generated a reasonable trajectory which was consistent with the sonar data, and used this to construct maps of two small tunnels, which we grafted onto the base of the partial Zacatón map (Figure 3.7). By combining the two datasets, we created a a high-fidelity model of Zacatón including challenging hypothesized features, such as small tunnels, loops, and bell domes. From this combined model, which we called "Wakatón", we could simulate sonar ranges for any desired sonar geometry with any ground truth path, including loop closure. We generated virtual vehicle trajectories and then used sensor noise models to simulate sensor readings for that trajectory.

The sensor noise was generated from zero-mean normal distributions. We elected to use conservative (high) noise values for three reasons: we have little information about the performance of the various sensors on Clementine in Zacatón, we wanted to encourage particle diversity, and we wanted to cause a clear distinction between dead-reckoning and SLAM on the same dataset. Accordingly, the DVL velocity noise was $0.2m/s$ 1σ, the IMU yaw noise was 1° 1σ, the depth sensor depth noise was $0.01m$ 1σ, and the sonar range noise was $1m$ 1σ. We compared the simulated sonar ranges with real sonar data to verify that the synthesized map generated realistic data. Figure 3.7 shows the two trajectories that were used to generate the following results.

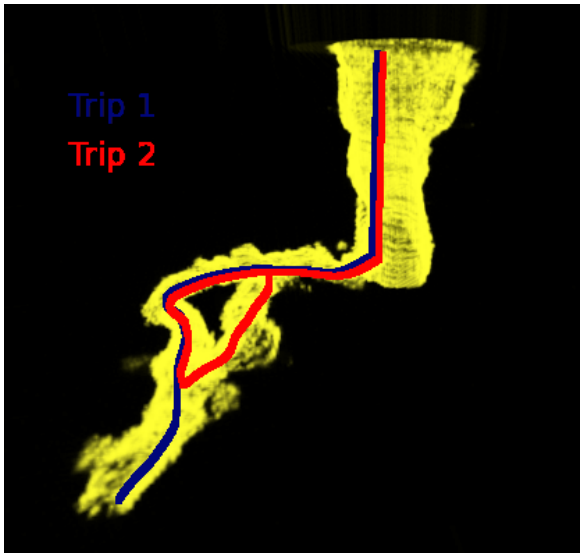

Figure 3.7: This figure shows the two test dives superimposed on the synthetic Wakatón map, which was constructed by merging portions of the Zacatón and Wakulla Springs datasets. Dive 1 went out to the furthest extremity of the tunnel and back, while Dive 2 went around the loop.

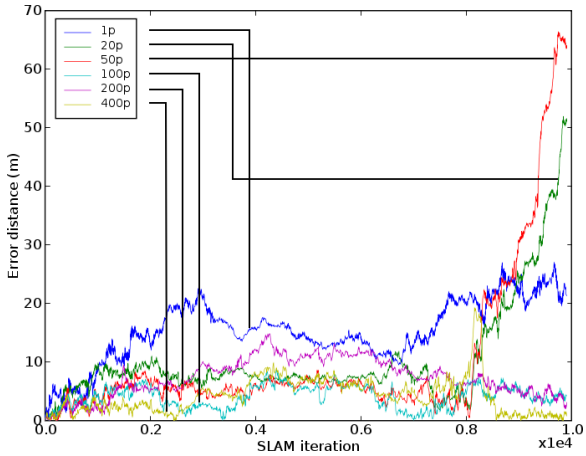

Figure 3.8: SLAM localization error during Dive 1 through Wakatón with 1, 20, 50, 100, 200, and 400 particles. One particle is equivalent to dead-reckoning. With 20 and 50 particles, the filter appears to diverge near the end of the dive. The performance clearly improves as the number of particles increases beyond 100 particles.

SLAM Characterization Figure 3.8 shows that as the number of particles increases beyond 100 the localization error decreases, although the error does not go to zero. Below 100 particles the filter diverges, and so it actually performs worse than just dead-reckoning (which is the same as 1 particle with no noise). This is due to particle depletion, when the filter is unable to adequately sample the posterior distribution with so few particles.

Figure 3.9 shows that SLAM with 200 particles has a relative accuracy of about 1m. The first dive has significant *absolute* error (as shown in Figure 3.8), the second dive (which used the map constructed during the first dive) is close enough that we will be able to consistently return to within about a meter of the same location for sampling.

Deferred Reference Count Octree Characterization The deferred reference count method rolls the cost of propagating reference counts into the insert operation, but the overall performance improvement is substantial. Without deferred reference counts, 70% of the SLAM processing time during a Wakatón dive is consumed by maintaining reference counts (for map copies) while only 30% is being used for ray-tracing (for weighting and updating the maps). Deferred reference counting reduces the map management overhead by a factor of ten, to 7%. But this under-represents the actual performance gain in the context of the particle filter: a major cost for the particle filter comes from particles which are resampled (copying the pose and map from a more highly weighted particle) in one time-step, and then are resampled *again* in the next time-step – without ever inserting anything into their maps. DRCO's give us this for free, simply

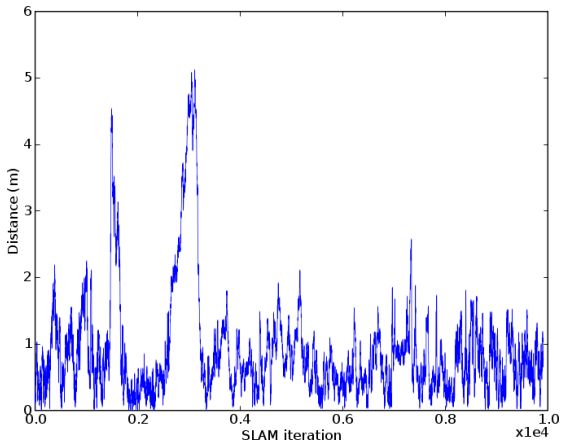

Figure 3.9: SLAM repeatability in Wakatón shows that with 200 particles the vehicle re-localized to within about 1m of its first estimated trajectory. This figure shows the distance between the first estimated trajectory, and the second estimated trajectory that used the map constructed during the first dive.

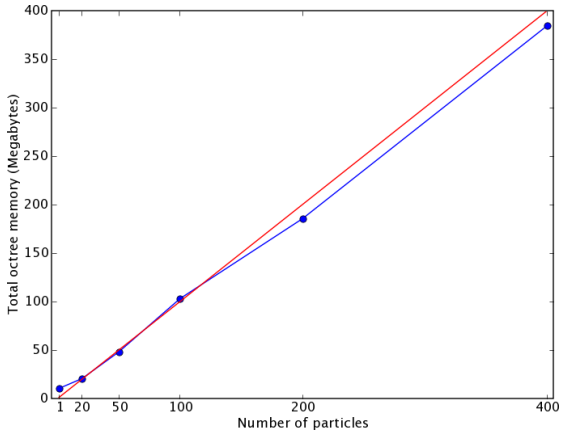

Figure 3.10: There is a slightly sub-linear relationship between number of particles and octree map memory in Wakatón (each particle taking about 1 Mb of memory). The red line shows what a linear relationship would be.

incrementing and decrementing the defCount of the root node.

Octree ray-tracing is about three times slower than uniform ray-tracing in our implementation. However, due to the enormous gains in map efficiency, octree maps allow the SLAM system to support hundreds of particles with high-resolution maps; something that isn't even possible with uniform maps (for example, the 400 particles with 0.5m resolution 512^3m maps used in the experimental scenario).

Figure 3.10 shows that the DRCO storage efficiency of the particle filter increases with the number of particles (due to increased amounts of overlap between particles). Each particle takes about 1 Mb of memory, and about ten seconds of computation time. Memory usage and computation time increases cubically with map resolution, as expected.

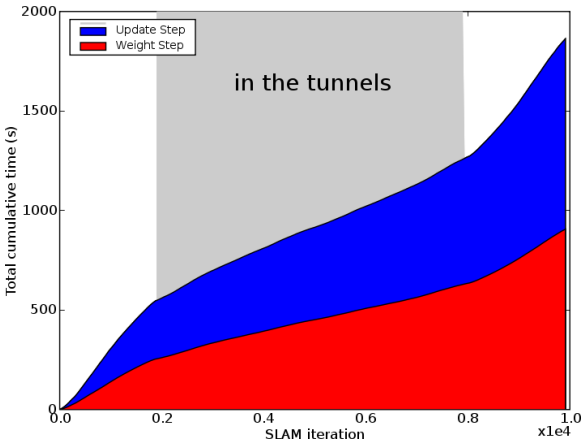

Figure 3.11: Cumulative processing time of the SLAM loop with 200 particles is shared almost equally by two of the steps: weighting and updating. It also shows that while the vehicle is in the tunnels at the bottom of Wakatón, the iteration duration is shorter, as we would expect since the sonar ranges are shorter.

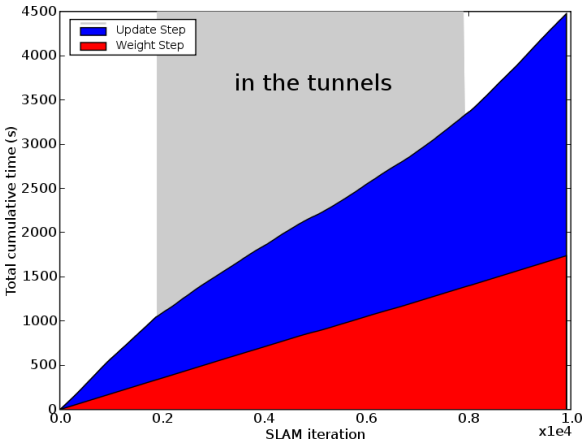

Figure 3.12: Cumulative processing time of the time limited variant of the particle filter. For this run, the weighting step was limited to 0.18s. Compare the improved consistency versus a fixed number of particles (Figure 3.11).

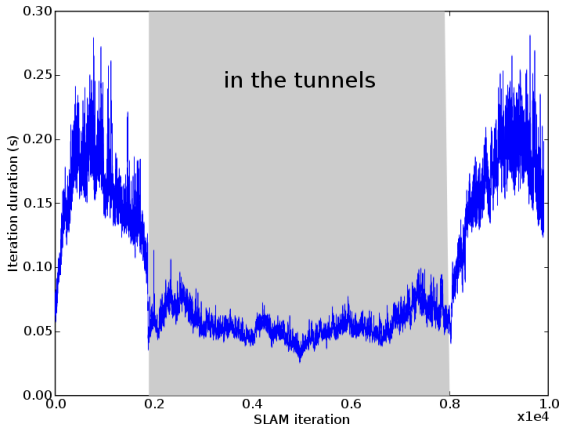

Figure 3.13: Processing time of the SLAM loop drops significantly while the vehicle is in the tunnels at the bottom of Wakatón, as we would expect since the sonar ranges are shorter. This indicates that the particle filter could use more particles while in the tunnels.

Adaptive Particle Count Characterization Figure 3.11 shows that SLAM time is roughly split between the weighting and resampling steps, and Figure 3.13 demonstrates that the amount of processor time for a single SLAM iteration varies significantly between when the vehicle is in the main Wakatón cylinder and when it is in the much more constricted tunnels. The difference is due to the fact that the weighting and update steps rely on ray-tracing operations which are proportionately faster in small spaces. This encourages the idea that the particle filter could do better if it could vary its particle count: Figure 3.15 shows the number of particles that were evaluated for different values of the weighting time limit. Figure 3.12 shows the improved

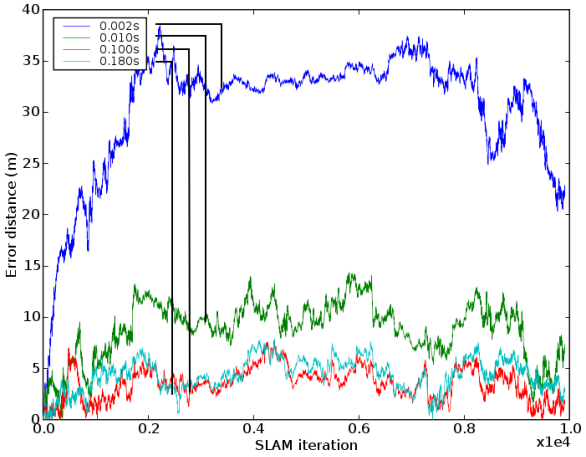

Figure 3.14: Localization error over time for different time limits for the weighting function. As expected, the particle filter performance improves as the amount of time it is allowed to spend weighting particles increases.

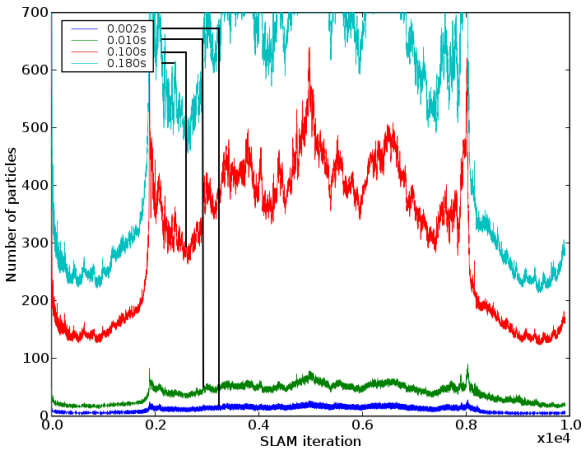

Figure 3.15: Number of particles evaluated (weighted) over time for different time limits for the weighting function (on the SLAM processor). As expected, the particle filter is able to evaluate more particles as the amount of time it is allowed to spend weighting particles increases.

timing consistency for adaptive particle count version of the particle filter (although there is still variation in the amount of time necessary for the weighting step).

The idea of the adaptive particle count is to use as many particles as possible in real-time *without* discarding any data. This is the dual of the KLD sampling technique of Fox [2003], which uses as many particles as needed at the expense of dropping data. The adaptive particle count method appears to provide an improvement over an equivalent (from a real-time performance perspective) fixed particle count, particularly while in enclosed areas. As an empirical observation, enclosed areas are precisely when the SLAM solution lacks good localization information and needs more particles. The most important contribution of adaptive particle count is the reliable real-time performance in the face of unknown world geometry (shorter *or* longer ranges). However, it does not provide any guarantees about avoiding under-sampling. We believe that a better system would incorporate both KLD sampling and temporal constraints to determine the particle count.

The particle filter/octree combination is stable with regard to the fraction of particles that are resampled at each time step (which can vary between 0 and $N_{part} - 1$). Although discarding a particle is costly because of the potential for a recursive freeNode, the node does not have to be updated – which saves many ray insertions. What does seem to be a reliable indicator of the duration of an iteration is the average range length, which makes sense given that ray-casting is the dominant operation and is linear in the range length.

Real-Time Characterization Figure 3.15 shows that for the desired 1Hz SLAM timing, which is achieved with a weighting time of 0.1s, the SLAM processor (a 1.8 GHz Pentium M) is capable of supporting between 100 and 600 particles, with an average of around 300.

We expect that map quality will degrade as the sonar ranges increase as a natural consequence

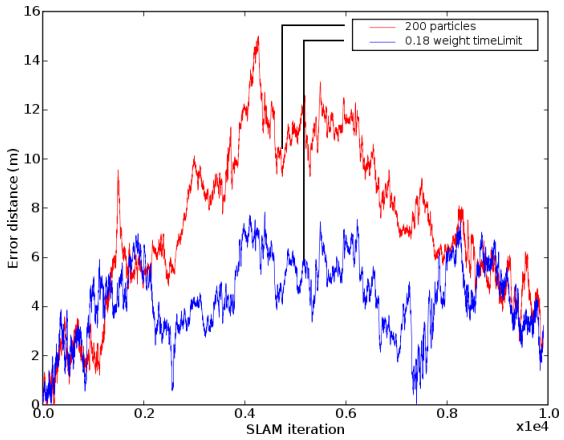

Figure 3.16: Comparison of localization performance of the two variants of the particle filter. In blue, the particle count was allowed to vary, but the weighting time limit was set to 0.18s. In red, the particle filter used a fixed particle count of 200, which is the maximum static number of particles that could be safely specified to achieve the same 0.18s real-time performance (Figure 3.15).

of the spreading of the sonar beams, and also the fact that coverage will tend to be more sparse (per unit area). At the same time, localization quality may well be better with long ranges because the vehicle will be sensing over a large area, which should provide strong localization information. It seems that as sonar ranges increase, precision degrades but overall accuracy improves. The opposite is true in small tunnels. This phenomenon explains why we see improved performance with a varying number of particles (Figure 3.16): the filter maintains accuracy in enclosed areas because of the increased number of particles which it can support.

Dead Reckoning Performance in the Quarry

Dead reckoning provides a gradually drifting position estimate, but it is also the basis of the motion model that is used to predict particle positions for Localization and SLAM, so it is useful to understand its performance. Figure 3.17 shows the performance of the dead reckoning system during a long raster mission to map a flooded quarry near Austin, TX [Kantor et al., 2007]. This mission had a total path length of over a kilometer, at the end of which the positioning error was 3.16 meters. This gives a respectable dead reckoning accuracy of less than half a percent of distance traveled. It should be noted that GPS fixes used as ground truth in this experiment were taken using a hand held non-differential GPS receiver from a moving boat, so it is likely that the dead reckoning estimate is actually more accurate than the "ground truth."

The red dots in Figure 3.17 denote locations where the DVL failed to achieve bottom lock. During this 6300 second mission, was without DVL measurements for a total of about 780 seconds. Some of these dropout periods were as long as sixty seconds. Without any source of velocity information it would have been impossible to achieve a reasonable dead reckoning estimate, so we used a Kalman filter to switch to IMU-derived velocities that were corrected by the DVL measurements, when available.

By inspecting the dead reckoning system, we found that the most significant sources of error were mis-calibration between the IMU and DVL frames, DVL velocity scale factor bias, and

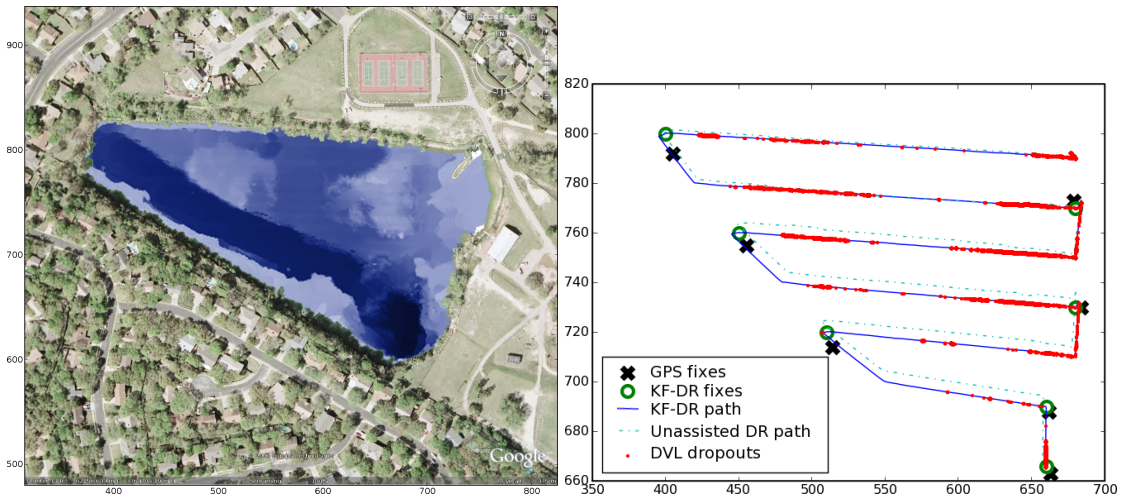

Figure 3.17: Left: A bathymetry map of the quarry, created by interpolating the sonar data from several raster survey missions, darker blue is deeper, max depth in the south corner was about 15m (Background satellite imagery courtesy Google, ©Europa Technologies, 2007). Right: a comparison of dead reckoning localization performance with and without the IMU velocity Kalman filter over a raster scan mission of the quarry. Coordinates are in truncated UTM northings and eastings.

random-walk IMU yaw error.

3.7.2 Localization – Zacatón

Sistema Zacatón is a chain of flooded *cenotes* (sinkholes) in Tamaulipas, Mexico. The water-filled formations are thought to have formed as hydrothermal groundwater dissolved through a layer of limestone [Gary, 2002].

The mineral-rich water of the cenotes supports colorful microbial mats in the photic zone and has exotic geochemical features which make it an excellent match for an exploration and scientific sampling mission. The goal of the DEPTHX project is to autonomously explore and map Sistema Zacatón, including any underlying tunnel systems, and then to use various environmental signatures (such as thermal plumes) to direct focused sample collection – with the goal of detecting and sampling unusual microbiota.

Over the course of three field expeditions to Mexico, the Clementine explored four of the cenotes of Sistema Zacatón (Figure 3.18). We began by testing the mapping system, then localization with a prior map, and finally SLAM.

Mapping the cenotes

The dead-reckoning error was usually less than 1% of distance travelled, but the range data from the sonar array was sparse, noisy, coarse, and low rate. Furthermore, the noise signatures of the sonars varied wildly between cenotes, being best in Caracol and worst in Verde, but generally poor (Figure 3.20). Our evidence-grid approach is well suited to this type of noisy data because it can combine multiple readings to reject the noise (Figure 3.21). Using the dead-reckoning

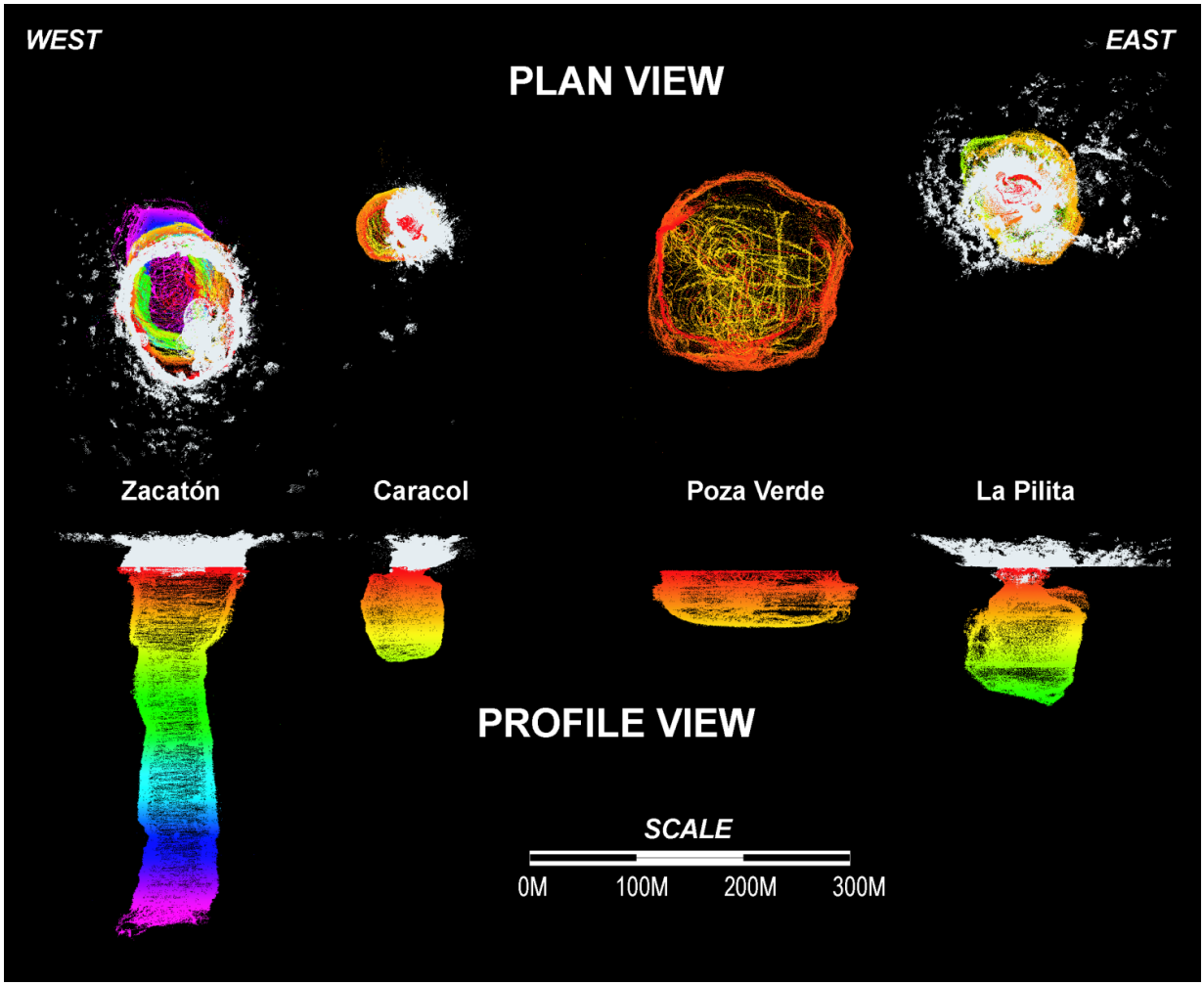

Figure 3.18: Plan and profile views of Sistema Zacatón, all distances are to scale. These point clouds include lidar data (in white) and sonar data (depth cued color).

and sonar data collected during long missions into La Pilita, we constructed an evidence-grip map, which could also be converted to a mesh by extracting an occupancy isosurface, for ease of human interpretation (Figures 3.22 and 3.23). In the evidence grid representations, individual sonar beams caused by spurious sonar measurements can be seen projecting through the surface of the cenote (Figure 3.22:Left). From an algorithmic standpoint, these spurious measurements do not usually degrade the quality of the map, since the holes they bore in the cenote walls are patched by other sonar measurements, as can be seen when the evidence grids are converted to meshes by isosurface extraction (Figure 3.22:Right). Since there is no ground truth for the cenote geometry, we can't make firm statements about the map quality – however by comparing maps from multiple missions, we are convinced that the maps are at least consistent.

These prior maps were next used with our particle filter to localize the vehicle.

Figure 3.19: At 110 m in diameter and over 350 m in depth, the *cenote* Zacatón in central Mexico is a unique flooded sinkhole. A platform for conducting preliminary sonar tests is tethered in place.

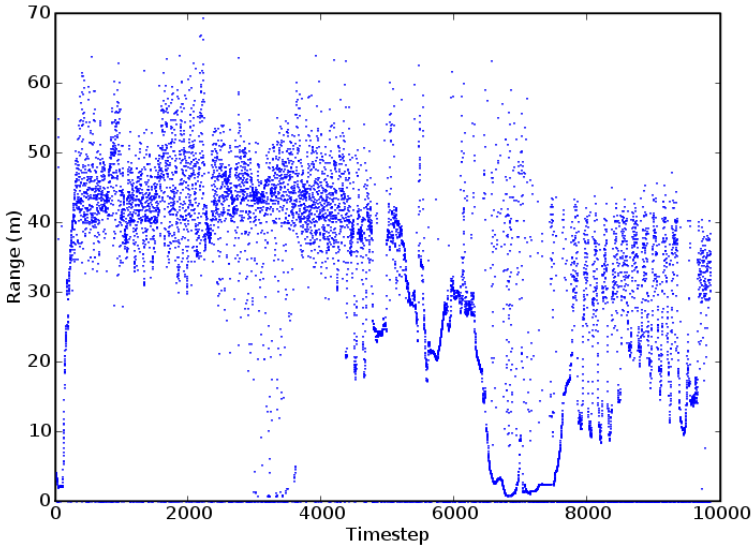

Figure 3.20: This plot of ranges from a single sonar transducer over time attempts to convey the level of noise in the sonar ranges. This is a forward-facing sonar, and between timesteps 6000 and 8000 the vehicle was fairly close to a wall, so the sonar ranges are reasonably good. Much of the rest of the time it is hard to see any signal in the noise, though the sinusoid after timestep 8000 is due to vehicle rotation.

Localization in Zacatón

Particle filter localization with a prior map, usually a map built using an optimized dead reckoned trajectory, is faster and less memory intensive than full SLAM because there is only one map which is not updated in real time. Localization is also more robust than SLAM, partly because the reduced computation requirements mean that more particles can be supported, but primarily because the static map reduces local minima in the particle weighting function.

Generally, dead reckoning error was too low over the 4 hour mission periods to detect significant errors, at least with the very crude start and end point ground truth that was available. How-

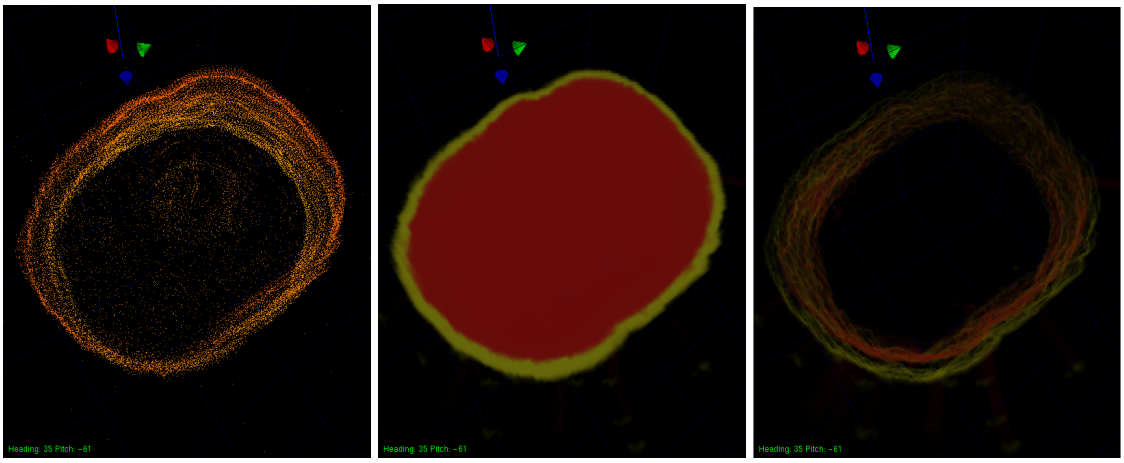

Figure 3.21: An example of how evidence grids cope with spurious sonar range values. Left, a slice from a noisy point cloud. Middle, a representation of the evidence grid where red shows low probability of occupancy, and yellow high. Right, the same evidence grid, but showing low and high occupancy isosurfaces – note that the noisy ranges inside the cenote have been filtered out.

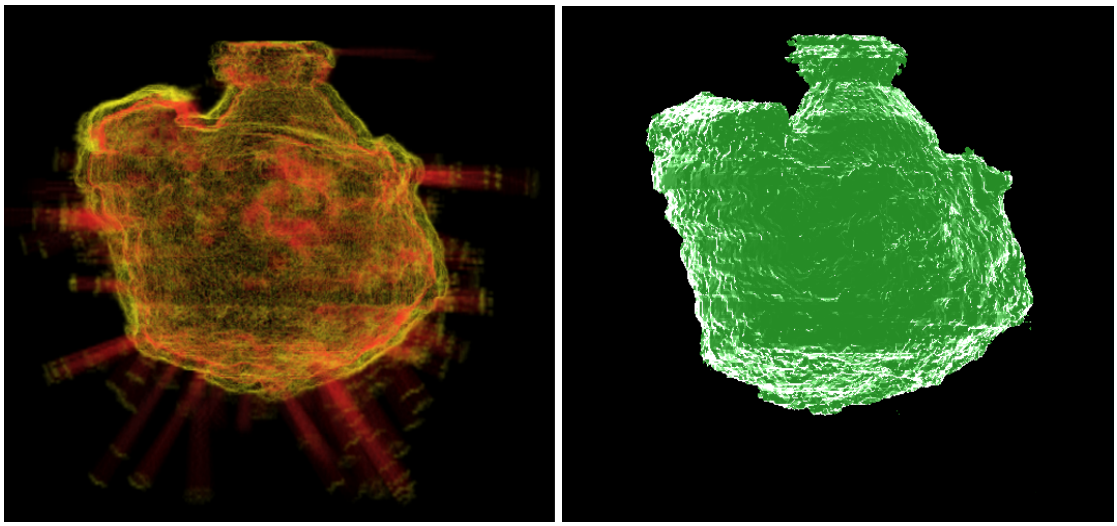

Figure 3.22: Left: An east-facing orthogonal view of an evidence grid of La Pilita. The maximum depth of La Pilita is 117 m, and its width is about 100 m, while the narrow neck that opens to the surface is only about 30 m wide. In this figure, yellow indicates an occupancy isosurface, and red indicates a vacancy isosurface. Right: The same evidence grid, converted to a mesh by extracting an occupancy likelihood isosurface, and shaded with a fresnel shader to bring out surface details.

ever, during our longest mission to the bottom of Zacatón, the IMU had an internal pitch/roll fault which caused it to report $\pm 2^{\circ}$ pitch and roll excursions. Although Clementine safely returned to the surface, dead reckoning error was 12.5 m (0.76 % of DT) due to the incorrect attitude data.

On the other hand, a second mission to the bottom of Zacatón exhibited excellent dead reckoning performance, returning to the origin with 0.36 m error after traveling 1760 m (0.02% of distance traveled), with very few DVL dropouts (and no IMU fault). It would be very difficult to

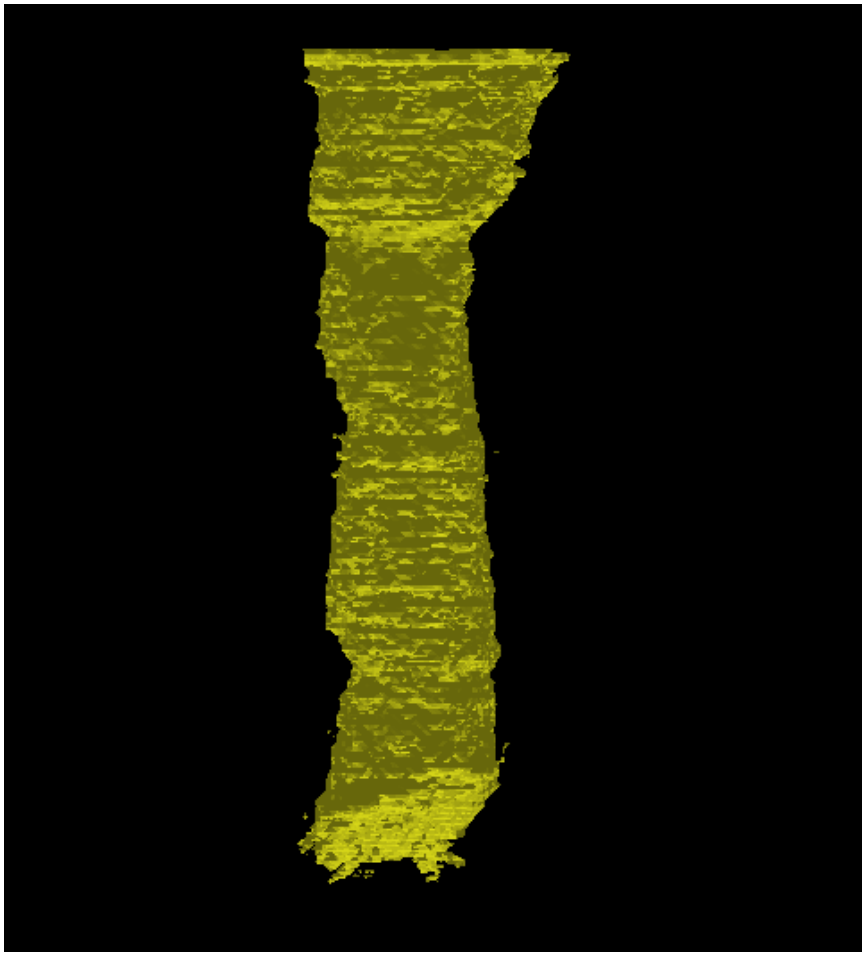

Figure 3.23: Side view of full depth (318 m) mesh map of Zacatón. In this rendered image, the evidence map of Zacatón has been converted to a mesh by extracting an occupancy likelihood isosurface.

demonstrate any improvement from localization or SLAM on this second mission without more ground truth information.

To test our method's ability to localize in the face of erroneous sensor data, we built a map with data from the second mission (using the excellent dead reckoning), and then attempted to perform particle filter localization with the data from the mission with the IMU fault. With a modest 300 particles, localization yielded a 0.2 m drift after traveling 1644 m (0.01 % of DT), compared to the dead-reckoning drift of 12.5 m (0.76 % of DT) (Figure 3.24).

A simple test of the localization system is to take a map (constructed by using optimized dead reckoning), initialize the particle cloud over a broad area, and then see if the filter can use the map and the new range data to quickly and reliably converge to near the known true position. This test was easy to run, and was quite successful in the simple geometry of the cenotes (Figure 3.25).

After running Localizaton with a prior map onboard Clementine to verify the operation of the particle filter, we experimented with running SLAM in the cenotes. Generally the onboard CPU ran with between 150-300 particles (manually adjusted to accommodate CPU load). During

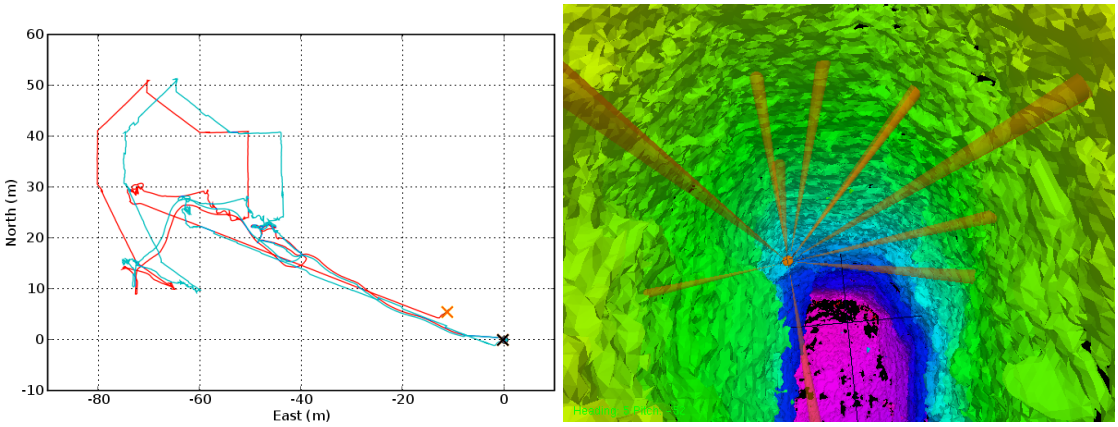

Figure 3.24: Left: This example of localization in Zacatón shows how the particle filter is able to use the prior map to reject corrupt attitude data: dead reckoning in red and localization (with the bad data but with a good map) in cyan. The 'return to origin' error, marked by the X's, was 12.4 m for dead reckoning and 0.2 m for localization. Right: a visualization of the prior map and the vehicle, showing the typical sparsity of the sonar data used to localize.

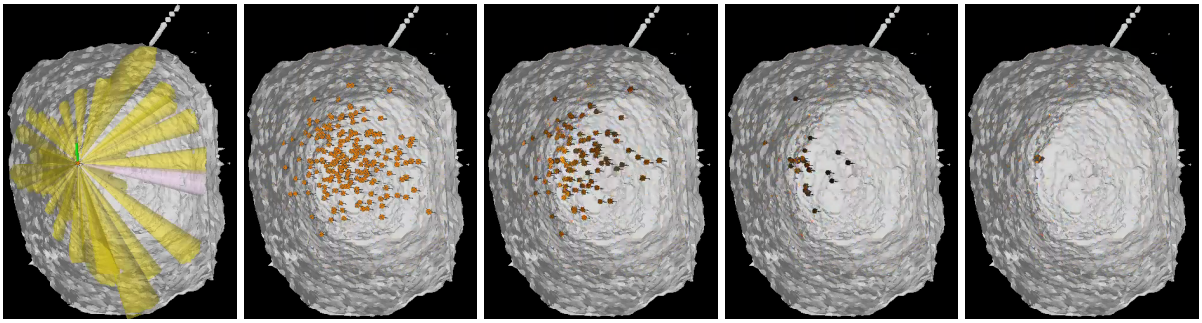

Figure 3.25: Kidnapped robot example in Caracol.

our onboard tests, SLAM was configured to be very conservative and to fall back to DR if the solutions diverged significantly.

3.7.3 Onboard SLAM – La Pilita

In La Pilita, SLAM ran onboard Clementine with 300 particles and provided the real-time pose to control the vehicle (Figure 3.22). The lack of ground truth makes it hard to make any strong assertions about the accuracy of the SLAM solution, except that it was reliable and comparable to dead reckoning without any DVL dropouts: both solutions usually differed by less than a meter upon returning to the starting location (our only ground truth point).

SLAM Failure in Verde Verde has a different morphology than the other cenotes, 150 m across and only 30 m deep, an example of a more typical lake environment, though we did find that Verde has some interesting undercut walls. Furthermore, the temperature profile in Verde shows distinct layers, more like a typical lake. Perhaps due to these conditions, the acoustic

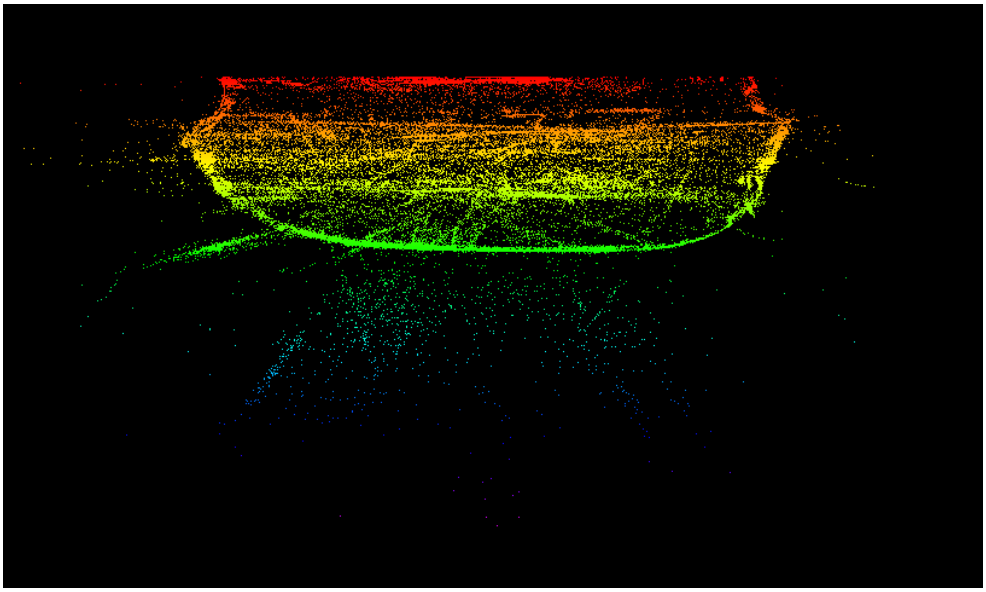

Figure 3.26: This thin slice of the sonar pointcloud from Verde illustrates the high level of sonar noise present in Verde – the interior of the cenote should be empty! The denser center band of noise is due to the fact that Clementine spent most of the time operating within that depth band.

environment in Verde was poor, resulting in noisy sonar readings (Figure 3.26) and frequent DVL dropouts. We did not even attempt to run SLAM onboard in these conditions, and offline attempts with logged data failed.

3.7.4 SLAM Performance – Caracol

Caracol is the smallest cenote, and we were able to adjust the sonar gains to yield very low range measurement noise. As we have seen, the dead reckoning performance is generally very good. However, when the DVL loses bottom lock, dead reckoning degrades significantly. The DVL often lost bottom lock in two (common) situations: while Clementine was on the surface waiting for the next mission, and during near-wall operations. These intermittent DVL dropouts caused enormous dead reckoning errors, despite our attempts to substitute the IMU's velocity estimate.

We have several metrics for evaluating navigation performance. As previously described, we manually drove Clementine under a plumb-bob at the start and end of each mission, which yielded the loop closure error. Another metric is based on the map: intuitively, accurate navigation will produce a high-quality map, with none of the offset surfaces that indicate position drift (Figure 3.27). While this "map quality" is easy to observe, a more objective metric is that of map entropy. As discussed in Section 2.6, the information-theory entropy H of the evidence grid map Θ is the sum of the entropies of each voxel θ_i, where $p_i = p(\theta_i)$ is the probability that voxel θ_i is occupied:

$$H(\Theta) = -\sum_i p_i \log(p_i) - (1 - p_i) \log(1 - p_i) \tag{3.7.1}$$

This is because evidence grid maps represent space with a grid of independent binary random

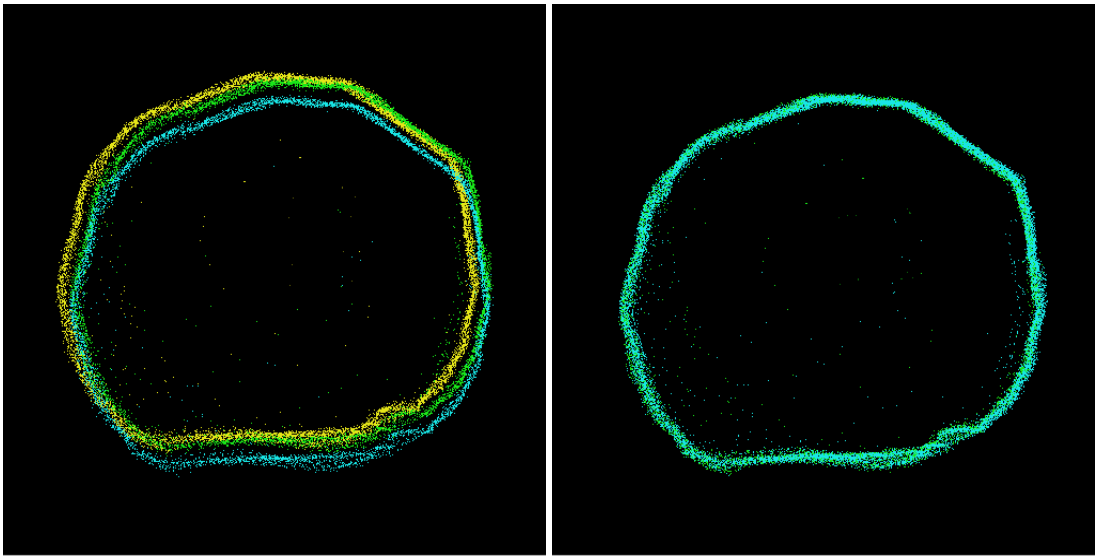

Figure 3.27: A top-down view of a slice of the dead reckoning (left) and SLAM (right) pointclouds constructed over the course of several missions (color coded). Dead reckoning shows significant drift, while SLAM keeps the map properly aligned.

Algorithm	Entropy	Loop error
Dead reckoning	38436.4	2.68
20p SLAM	33317.7	1.33
50p SLAM	32045.2	0.41
150p SLAM	32312.7	0.29
300p SLAM	31917.0	0.26
600p SLAM	31952.3	0.25

Table 3.2: SLAM results in Caracol

variables indicating occupancy. By constructing evidence grid maps with various navigation solutions, we can compare the resulting entropy of the maps. Maps with low entropy are very "certain," that is they are composed of sharply delimited regions of very low or very high probability of occupancy. High entropy maps are blurrier, with regions closer to 50% probability of occupancy.

Clementine collected data in Caracol over the course of two hours, running several missions and returning to the surface between missions. Dead reckoning loop-closure error was 2.7 m due to frequent DVL dropouts. SLAM loop-closure error was only 0.3 m. Similarly, a the map drift (compared visually in Figure 3.27) and the more rigorous SLAM map entropy were both significantly lower, as shown in Table 3.2.

3.7.5 SLAM Experimental Summary

In this chapter, we described our approach to robust real-time 3D SLAM with a Rao-Blackwellized particle filter and Deferred Reference Count Octree map representation. As introduced by Doucet

et al. [2000], Rao-Blackwell factorized particle filters, in which each particle is composed of both an analytically constructed map and a vehicle position within that map, is the key insight that allows the application of the non-parametric particle filter approach to SLAM. Our approach to RBPF SLAM efficiently exploits spatial *sparsity* because the DRCO compactly represents large unobservable regions and/or large homogenous regions. Our approach also exploits spatial *locality* because many RBPF particle maps have identical regions, which are automatically exploited as a result of particle filter resampling and copy-on-write.

Due to these efficiencies, we showed that adding additional RBPF particles has very low storage cost, meaning that the limiting factor on the number of particles is the computation involved in ray-tracing for the weighting and updating steps. Since ray-tracing computation is linear in the length of the rays, we showed how our adaptive particle count method opportunistically uses more particles in enclosed environments (with shorter range measurements) to improve SLAM performance.

One common but important modification due to Liu [1996] to the standard particle filter formulation is to suppress resampling until the number of effective particles drops below a threshold. This ammeliorates the effects of particle depletion.

The best possible dead reckoning is important for minimizing the amount of sample space that the particles must cover, particularly the heading estimate. Both underwater vehicles discussed in this dissertation have high-grade IMU's with negligible heading error, as well as an absolute depth sensor with very little error. This means that the particles are primarily spread over the two horizontal degrees of freedom, rather than over all size degrees of freedom. While the subterranean vehicles had much lower-quality IMU's, they had the great advantage of being able to come to a complete stop, at which point the IMU biases can be accurately measured.

For each different vehicle, it is important to use the proper motion error model for the particle predict step, with the emphasis on accurately characterizing the yaw error and velocity error. Since these error parameters may vary according to unknown environmental parameters (water currents, slippery floors, etc.), it is generally good to use pessimistic (high) values.

3.8 Summary

In this chapter, we described our approach to robust real-time 3D SLAM with a Rao-Blackwellized particle filter and Deferred Reference Count Octree map representation. Our approach efficiently exploits spatial *sparsity* because the DRCO compactly represents large unobservable regions and/or large homogenous regions. Our approach also exploits spatial *locality* because many RBPF particle maps have identical regions, which are automatically exploited as a result of particle filter resampling and copy-on-write.

Due to these efficiencies, we showed that adding additional RBPF particles has very low storage cost, meaning that the limiting factor on the number of particles is the computation involved in ray-tracing for the weighting and updating steps. Since ray-tracing computation is linear in the length of the rays, we showed how our adaptive particle count method opportunistically uses more particles in enclosed environments (with shorter range measurements) to improve SLAM performance.

We then demonstrated our approach to 3D SLAM onboard an underwater vehicle in a series

of challenging underwater environments. Due to the serious risk of losing the vehicle during its exploration of these environments, we described our extensive experiments to characterize our system before deployment. Our method was then able to use sparse, noisy, low resolution and low rate sonar data to build maps, localize, and run SLAM within the flooded cenotes of Sistema Zacatón. Our system generated the first ever maps of these large-scale natural formations.

In the chapters that follow, we will also present results from using this 3D SLAM approach with laser data in tunnel-like mine and building environments, and with multibeam sonar on the ocean floor.

We have shown that our method is a general, robust approach to real-time SLAM with range data in large-scale 3D environments with arbitrary structure. However, as with all real-time particle filter solutions, computational constraints on the number of particles limits scalability, in particular the ability to close large loops. In the next chapter, we will show how to use our RBPF SLAM as a building block for a segmented SLAM approach that attempts to address scalability and loop closure.

Chapter 4

SLAM with Segmented Maps

4.1 Introduction

Many SLAM methods, including our Rao-Blackwellized Particle Filter (RBPF) approach described in the previous chapter, work well on a limited scale. In particular, they build a very good maps on the scale of tens or perhaps hundreds of meters. To succeed even at that scale, most SLAM methods exploit spatial independence, or sparsity.

There are, however, a group of SLAM methods that explicitly exploit spatial sparsity by segmenting the world into submaps. Most of these methods use a combination of metric and topological maps, in which the relationships between submaps are represented by the edges of a graph, and the nodes of the graph point to the submaps. The submap segmentation is usually designed such that their scale is well within the capabilities of a regular SLAM approach, such as the one presented in the previous chapter. Thus the scaling problem is avoided, but the trade-off is that the submap algorithm must manage the graph of submaps, deciding when to create a new segment, when to re-enter an old segment, and how to represent different hypotheses about the topological relationships between segments.

We present a robust, real-time, submap-based approach called SegSLAM that uses an extension to the particle filter prediction step to determine when a particular particle should transition to a new segment or re-enter an old segment: weighting, resampling, and updating are still applied. As the particles of a regular RBPF are a discrete approximation to the distribution over poses and the metric maps $p(x, \theta)$, the particles of SegSLAM are a discrete approximation to the distribution over poses and segments, where the poses are in the local coordinate frame of the segments. Unlike RBPF particles, SegSLAM particles do not encode a complete trajectory hypothesis, instead the trajectory must be reconstructed by stitching together compatible segments, which are stored in a data structure called the segmented map. The segmented map encodes a probabilistic graph, in which the particle transitions from one segment to another (during the predict step) are the edges of the graph, but there may be many different metric maps for each segment, one for each particle. Reconstructing a trajectory can be seen as drawing a sample from this probabilistic graph by stochastically picking edges and nodes to create a (partial or complete) metric map, which can then be used for loop detection and planning.

Another way of interpreting sampling from the segmented map is in the context of particle

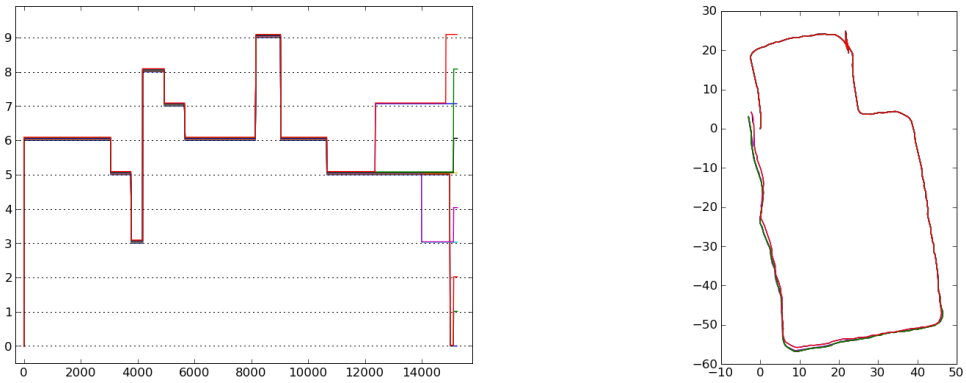

Figure 4.1: Trajectory depletion: Results from running our RBPF with 10 particles around a loop in Bruceton mine. Left, the particle ancestry over time shows that all the particles share a fairly recent common ancestor (particle 5). Right, the particle trajectories tell the same story, showing two competing hypotheses emerging near the lower right corner. The particle filter fails to close the loop, no surprise with only 10 particles.

depletion, which is the limiting factor on the scalability of our RBPF SLAM approach. Particle depletion occurs when the filter is not able to maintain an adequate representation of the underlying distribution over trajectories (poses and maps). For any resampling particle filter with a finite number of particles, all the particles will eventually come to share a common ancestor as an inevitable result of the resampling step. When this occurs, the particle filter effectively has only one hypothesis about what happened before the oldest common ancestor, and any errors in this hypothesis are unrecoverable. The particle depletion problem often arises in the context of closing a loop, as shown in Figure 4.1. The particle filter needs to maintain many viable trajectories hypotheses around a loop in order to have a selection to choose from when it closes the loop. The difficulty in doing this with an RBPF comes from the fact that the particle trajectories are encoded in the maps, so the filter must maintain a map for each of these hypotheses. Maps are usually expensive to maintain, so computational capacity limits the number of trajectories that the filter can support which in turn limits the loop length that the filter can reliably close.

SegSLAM improves this situation in two ways. First, the segmented map represents an exponential number of trajectories, which can be sequentially sampled (Figure 4.2). Second, we can detect or refine topological relationships between submaps by using the map matching techniques described in Chapter 2 to directly match the segments, treating them like scans or even features. Its important to note that the particles do not have to segment or re-enter at the same time, but the resulting segments will not be temporally compatible and can't form part of a reconstructed trajectory. In particular, the ability of different particles to independently segment and re-enter is how SegSLAM can represent different topologies.

This chapter begins with a more thorough overview of the SegSLAM algorithm. We then review related work in submap methods, and then provide a derivation of the modified SegSLAM predict step. Next, we address the issues of segmentation, metric map reconstruction, and map matching or re-entry. We briefly discuss SegSLAM entropy, and then present the results of three

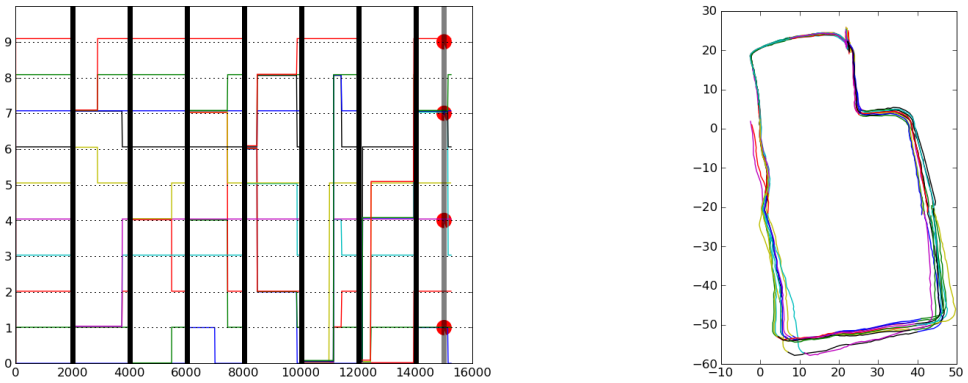

Figure 4.2: Maintaining trajectory variety: Results from running SegSLAM with 10 particles around a loop in Bruceton mine. Left, the particle ancestry over time shows how segmentation, illustrated by vertical black lines, maintains particle diversity, while still discarding unlikely hypotheses. In addition, the red dots indicate that four of the particles have found a map match, closing the loop. Right, this subset of reconstructed trajectories is split between the two main topological hypotheses: that the loop closed and that it didn't. In this plot, different colors indicate different segments as well as different particles.

different real-world experiments. We conclude the chapter with a summary.

4.1.1 Overview of SegSLAM

Every approach to SLAM has strengths and weaknesses. In Chapter 3 we discussed and demonstrated why our RBPF approach is well suited to SLAM with range data in large-scale 3D environments. But there are drawbacks to the approach we have outlined so far; the key issue being the particle depletion problem illustrated above. To summarize, due to computational constraints, a particle filter has a finite number of particles and it must periodically resample these particles in order to avoid the degeneracy problem [Arulampalam et al., 2002] in which one particle has almost all the weight, and effectively model the underlying distribution with its finite particles. An unavoidable consequence of resampling is that of particle depletion or sample impoverishment – as shown in Figure 4.1. In concrete terms, this means that the particle filter has a decreasing chance of detecting loops over progressively longer distances. Further, even if the robot can use some other means to detect a loop (vision, RFID, etc), the evidence grid maps of the RBPF particles can't be updated to reflect the trajectory correction – they'd have to be reconstructed from scratch.

The goal of SegSLAM is to address the problem of loop closure and evidence grid rigidity, while maintaining a solid Bayesian foundation, support for arbitrary 3D environments, and a robust real-time implementation. Accordingly, SegSLAM is a constant-time algorithm, and requires storage linear in the volume of space explored. An important caveat is that SegSLAM does this by relaxing the need to constantly maintain a globally metric map.

SegSLAM extends the standard particle filter SLAM formulation by extending the prediction step to include the current particle segment, as well as the particle position within that segment.

As the robot travels around, SegSLAM applies a segmentation heuristic that uses either an estimate of the gradual accumulation of motion error, or an estimate of how well the current map predicts the next few seconds of range data. These segmentation heuristics reflect the idea that a submap should be small enough not have significant position error, and that a submap should be able to predict measurements as long as the robot remains within the submap. The heuristic looks ahead a few seconds in time to pick the best moment to segment, so SegSLAM runs a few seconds behind the current time. The SegSLAM position estimate is brought up to the current time by appending the dead-reckoned trajectory, which is accurate enough over a few seconds to bring the estimate up to date.

When the segmentation heuristic indicates that its time to segment, SegSLAM stores the old particle segments in the segmented map and creates new segments for the participating particles, placing each at the origin in a new local coordinate frame with a new map. As might be expected, in unconstrained (open, outdoor) environments, segmentation is basically periodic, depending largely on the motion error model. In constrained (indoor, structured) environments, segmentation is largely determined by the interaction of the sensor and environmental structure. This corresponds to an intrinsic definition of place: a map of a place will predict measurements within that place very well, but fail to predict measurements from another place.

To detect re-entry into a previously mapped area (matches or loop closure), SegSLAM reconstructs the relative transforms between the current submap and other nearby submaps, sampling from the set of particle trajectory segments stored in the segmented map in order to come up with a reasonable local metric representation. Note that this reconstruction can be truncated at any search depth, can be allowed to run for a fixed amount of time, or can be run until a complete trajectory has been reconstructed.

Using this partial metric reconstruction, it searches for proximity or overlap between submaps, which might indicate a loop. This matching process can be considered as a kidnapped robot localization problem in which the partial metric reconstruction gives an informed prior to the localization process, and the match is considered good if the particle filter converges and has low measurement error. Matching using localization is an example of Markov Chain Monte-Carlo (MCMC), in which the true distribution is approximated by generating a sequence of samples according to a simple mixing rule. Because MCMC localization methods can be too slow to apply at the rates needed to perform real-time loop closure, we also investigate how potential overlaps can be refined using map matching techniques from Chapter 2 which operate directly on the submaps.

With either method, we can quickly assemble a weighted list of potentially matchable segments and particle positions transforms into the respective segments. This list always includes the current segment and may include a completely new segment – segmenting can be seen as matching to a new map. The particle filter predict step then probabilistically selects between these options, sampling from the set of paired segments and transforms. Most of the time, the particle will remain in its current segment, behaving just like a regular RBPF particle. However, when a particle switches to another segment, the match is recorded in the segmented map structure such that the new relative transformation (the re-entry or loop closure) between the two submaps can then be used in subsequent iterations of relative transform reconstruction, leading to loops in the submap topology. Since SegSLAM only reconstructs the local submap transforms, it does not need to globally propagate information about new loops. Rather, particles that correctly

detect loops will have higher weights, and will tend to be resampled, shifting more particles to a region of the hypothesis space that includes the loop.

In summary, the SegSLAM predict step doesn't just sample from the vehicle motion model during the predict step, but also samples from the map topology model. The rest of the particle filter process is unaffected: weighting, resampling, and updating occur as if the particles were not spread among different submaps with fundamentally different implications for the submap topology. In this way, the finite computation resources of the particle filter are properly distributed more closely to the true distribution over trajectories.

SegSLAM also supports a gradual transition from exploration and mapping to long-term localization in two ways. First, well known segments can be locked to prevent updates, such that particles that re-enter those segments perform localization. This avoids the evidence erosion problem that degrades the long-term operation of RBPF SLAM with evidence grids. Second, submaps can be merged together when their relative positions become certain, or when they are discovered to share significant overlap. In the limit, this would ideally yield a single global metric map.

4.2 Related Work in Submaps and Large-Scale SLAM

Submap decomposition methods attempt to exploit the locality of large environments: only a small subset of landmarks are visible at any time. This limitation can actually be turned to an advantage by only updating one submap at any given time. The difficulty with sebmap approach is then deciding when to build a new submap, how to re-enter old submaps, and how to estimate the transforms between submaps.

Submap representation Submap methods have usually combine both metric and topological representations, where the nodes of the topological graph point to a metric submap, and the edges of the graph represent the connections between submaps, although some methods are primarily topological [Kortenkamp and Weymouth, 1994, Choset and Nagatani, 2001, Remolina and Kuipers, 2004]. As in the previous chapter, this metric information is usually represented as feature-based maps, for example [Tardós et al., 2002, Bosse et al., 2004, Estrada et al., 2005, Newman et al., 2003b, Lisien et al., 2003], but evidence-grid based submaps are not uncommon [Yamauchi and Langley, 1996, Jefferies et al., 2004, Schultz and Adams, 1998].

Segmentation The broad goals of segmentation are to limit the metric map size and accumulated error, and to make the submaps as independent as possible. Statistically independent is often asserted by giving each submap to have its own reference frame: in an early approach, each landmark had its own frame Bulata and Devy [1996]. Decoupled Stochastic Mapping [Leonard and Feder, 2001] is somewhat unique in that it divides the environment into overlapping submaps, but these maps share a globally reference frame: as a result this approach is fast but overconfident about transforms between submaps.

The best segmentations can only be found in retrospect, in post-processing, although we (and others) use a limited amount of look-ahead to pick the best segmentation points. This can be done on anthropomorphic or logical grounds using doors and intersections [Kuipers and Byun, 1991],

based on estimates of accumulated localization error [Chong and Kleeman, 1999, Bosse et al., 2004], the maximum desired number of features in each submap [Tardós et al., 2002, Estrada et al., 2005], the detection of special feature-rich regions [Simhon and Dudek, 1998], or even in post processing for offline approaches [Thrun, 1998, Friedman et al., 2007]. Frese [2006] describes the Treemap algorithm, an O(log n) approach that uses a hierarchically subdivided map. Lisien et al. [2003] uses the generalized Voronoi graph as motive behind segmentation (and as the topological map), a simple distance criteria for creating new landmarks, and then combines local maps by aligning the landmarks along the edges between the maps. Blanco et al. [2009] provides a probabilistically grounded method for segmenting based on normalized cuts.

As mentioned above, we use limited lookahead and a predictive score metric that estimates how well the current map predicts future measurements, combined with a localization error metric, as our segmentation criteria.

Matching Also called re-entry, loop closure, submap transform estimation, matching can be thought of evaluating pairwise hypotheses about submap connections. Many feature-based approaches use specially selected subsets of features near the edges of the submaps to match [Bosse et al., 2004, Estrada et al., 2005, Frese, 2006]. Constant Time SLAM (CTS) [Newman et al., 2003b] opportunistically fuses the feature estimates from multiple submaps to improve the global feature estimate (and the relations between submaps). Evidence-grid based methods use map overlap [Jefferies et al., 2004] or evidence grid matching [Yamauchi and Langley, 1996] to detect matchings.

In our approach to matching, we either directly apply the Octree map matching techniques of Chapter 2 to register the submaps, or we use particle filter localization, under the assumption that if localization is successful we have re-entered the target submap.

Topological Hypotheses If matching tests pairwise hypotheses about submap connections, topological hypotheses encompass all the submaps and how they fit together. Modayil et al. [2004] discuss a framework for dealing with uncertainty and error between the local metric, global topological, and global metric levels, but multi-hypothesis topological methods usually don't impose loop consistency (global optimality) in the interests of speed.

Some submap methods only support a single topological hypothesis. For example, Estrada et al. [2005] use nonlinear optimization to find loop closures between local maps, but the optimization is brittle in that it only yields a single topological hypothesis. Likewise, the closely related Compressed Filter of Guivant and Nebot [2001], Local Map Sequencing method of Tardós et al. [2002], and Constrained Local Submap Filter of Williams et al. [2002] build local submaps, and then periodically fuse them into a global map. Recent results for this method show improved $O(n)$ performance using a divide and conquer strategy Paz et al. [2007].

At the opposite end of the spectrum some methods track *all* possible topological hypotheses: Remolina and Kuipers [2004] and Savelli and Kuipers [2004] maintain trees of all possible topologies, sometimes with only weak sensing assumptions [G. Dudek, 1993], but these complete approaches fail in unstructured or "degenerate" environments.

As a middle ground, the ATLAS framework [Bosse et al., 2004] uses heuristics to select *some* topological hypotheses, and uses a variety of criteria for promoting or discarding hypothe-

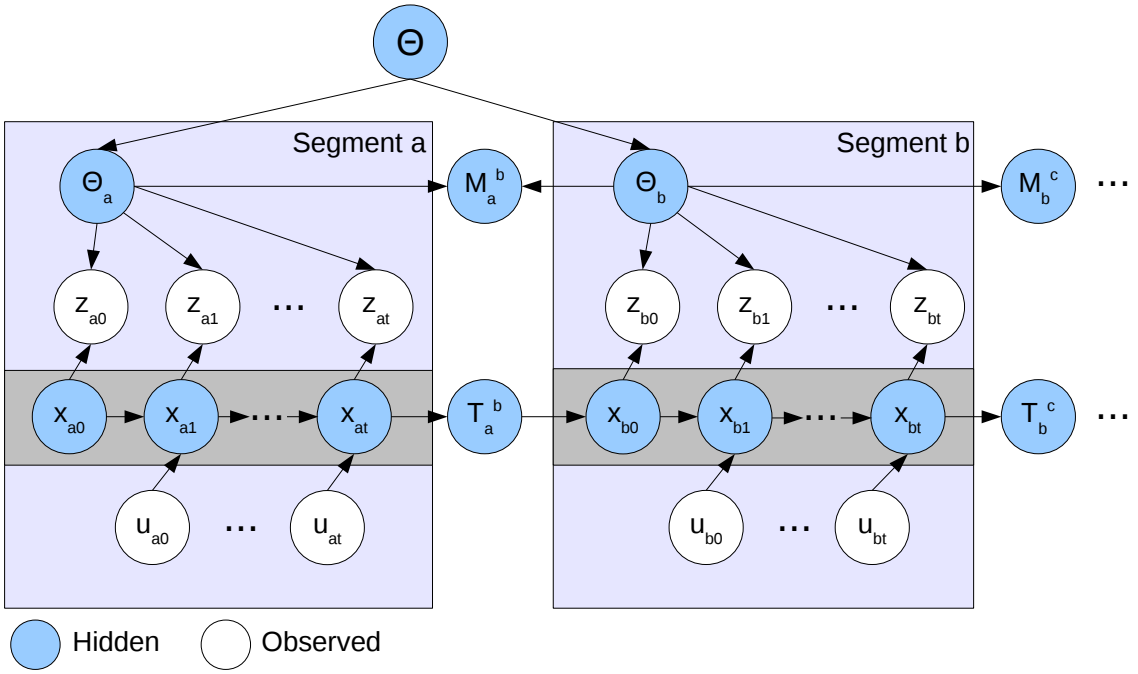

Figure 4.3: SegSLAM graphical model – segments are related only by the relative transform T between their coordinate frames and the match M, or overlap, between them.

ses. The ATLAS criteria for selecting topologies are somewhat ad-hoc, and Ranganathan and Dellaert [2004] apply more rigorous Bayesian inference to what they call Probabilisitic Topological Maps (PTM). They use a Markov-Chain Monte Carlo approach the estimate the distribution over possible topologies by sampling from the space of partitions of landmark measurements. Similarly, Blanco et al. [2008b]'s Hybrid Metric-Topological (HMT) SLAM uses the particles of an RBPF to sample topology between evidence-grid metric submaps, while using Kalman filters to estimate the transformations between maps.

Our approach is closely related to these methods, in that the segmented map data structure that underlies out SegSLAM approach is also an MCMC-based estimate of the distribution over both metric maps and topologies. SegSLAM particles do not represent a complete history from the entire vehicle trajectory, instead we use map reconstruction to grow large-scale metric map samples from the current particles as needed, for example for use in loop detecting. Thus we can consider a much larger set of topologies in an any-time fashion. SegSLAM is thus a constant-time/any-time algorithm based on the RBPF of the previous chapter, but integrated with a probabilistic topological map.

4.3 Derivation

In Section 3 we showed how the SLAM problem can be formulated as a Bayesian graphical model that exploits the conditional independence of measurements given the vehicle poses. SegSLAM makes the further assumption that the world can be spatially segmented into regions

that are conditionally independent given the relative transform and the map match score between the regions (Figure 4.3).

As described above, each Rao-Blackwellized particle consists of submap θ_i and a pose x_t relative to the coordinate frame of that map: $s_t = \{x_t, \theta_i\}$. For compactness, the set of particles over time is written $\mathbf{s}_t = \{s_1, s_2, \ldots, s_t\}$. The *overall* map Θ_t of the filter is a graph where the vertices V_t are the submaps and the edges E_t are the coordinate transforms between submaps:

$$\Theta_t = \langle \{\theta_i\}_{i \in V_t}, \{T_i^j\}_{i,j \in E_t} \rangle.$$

Remembering that the maps are represented as evidence grids, which are in turn composed of independent cells such that the probability of the map is the product of the probabilities of the cells:

$$p(\Theta) = \prod_{c \in \Theta} p(c)$$

We can then see how Θ can be factorized into submaps:

$$p(\Theta) = \prod_{i \in V} \frac{p(\theta_i)}{\prod_{j \in V} p(\theta_i \cap \theta_j)}$$

Where $p(\theta_i \cap \theta_j) = \text{MatchScore}(\theta_i, \theta_j)$ is pairwise map match score, a measure of how much the two maps overlap defined in Equation 2.5.2, that for notational simplicity is set to 1 when $\theta_i \cap \theta_j = \emptyset$. When the submaps are independent and the match is 0 for all but the self-match, then this reduces to

$$p(\Theta) = \prod_{i \in V} p(\theta_i),$$

the definition of conditional independence. And when the maps are all equal and the match is 1 for all pairs,

$$p(\Theta) = \frac{p(\theta_i)^{\#_V}}{p(\theta_i)^{\#_V - 1}} = p(\theta_i)$$

With a slight abuse of notation, we can condition on this match score M, writing

$$p(\Theta) = \prod_{i \in V} p(\theta_i | M)$$

Now in the segmented map the submaps don't share a common coordinate frame, so we must also condition on the relative transforms between submaps:

$$p(\Theta) = \prod_{i \in V} p(\theta_i | M_i) p(T)$$

Now that we know how to factor the segmented map, we can address the segmented SLAM posterior

$$p(\mathbf{s}_t, \Theta_t | \mathbf{u}_t, \mathbf{z}_t)$$

Using the same Rao-Blackwell factorization as in Chapter 3 to reduce the dimensionality of the particle space, this posterior can be factored as:

$$p(\mathbf{s}_t, \Theta_t | \mathbf{u}_t, \mathbf{z}_t) = p(\mathbf{s}_t | \mathbf{u}_t, \mathbf{z}_t) p(\Theta_t | \mathbf{s}_t, \mathbf{u}_t, \mathbf{z}_t) \tag{4.3.1}$$

The posterior of the trajectory $\mathbf{s}_t$ can be factorized as the product of its submap segments, remembering that $s_t = \{x_t, \theta_i\}$, and since the segments are in different coordinate frames they are independent:

$$p(\mathbf{s}_t|\mathbf{u}_t, \mathbf{z}_t) = \prod_{i \in V} p(\mathbf{s}_i|\mathbf{u}_i, \mathbf{z}_i).$$

where $\mathbf{s}_i$ is the subset of poses in segment i. Thus the pose portion of the segmented SLAM posterior can be safely treated a series of independent SLAM sessions. We now need to address the segmented map.

As we have shown in Equation 4.3, the posterior of the overall map Θ_t can be factored into metric submaps θ_i and transforms T_i^j because they are conditionally independent given the particle trajectories (see Figure 4.3):

$$p(\Theta_t|\mathbf{s}_t, \mathbf{u}_t, \mathbf{z}_t) = p(\{\theta_i\}|\mathbf{s}_t, \mathbf{z}_t)p(\{T_i^j\}|\mathbf{s}_t, \mathbf{z}_t)$$

And the joint posterior of the submaps $p(\{\theta_i\}_{\forall i)}$ can be factorized because the metric submaps are conditionally independent given particle trajectory:

$$p(\{\theta_i\}_{\forall i}|\mathbf{s}_t, \mathbf{u}_t, \mathbf{o}_t) = \prod_k p(\theta_k|\mathbf{s}_k, \mathbf{o}_k)$$

And since the posterior of the trajectory can be factorized into its segments, the posterior over transforms can be factorized as

$$p(\{T_i^j\}|\mathbf{s}_t, \mathbf{u}_t, \mathbf{o}_t) = \prod_{i,j \in E} p(T_i^j, \mathbf{s}_t, \mathbf{u}_t, \mathbf{o}_t).$$

All told, we have shown how we can analytically compute the distribution over the segmented map Θ of Equation 4.3.1 by keeping track of the transforms between submaps.

Thus the segmented RBPF must track the particle poses x, as well as the transformations T that come from the robot's transition from one submap to another, or from map alignment techniques. We already know how to analytically construct evidence grid maps for the distribution of θ_i. The particle filter approximation to the recursive Bayesian formulation can be derived as in Chapter 3, with the important modification that the particle filter must now sample from the distribution of submaps during the predict step:

$$s_t^{(i)} = \{x, \theta_i\} \sim q(s_t|\mathbf{s}_{t-1}, \mathbf{u}_{t-1}, \mathbf{z}_{t-1})$$

As before, we can use the motion model to update $x_t^{(i)}$, but now we also sample from $q(\theta)$

$$\theta_i = q(\theta|\mathbf{s}_{t-1}, \mathbf{u}_{t-1}, \mathbf{z}_{t-1})$$

There are three main possibilities: first, that the vehicle is still in the same submap region and should stick with the current submap, second, that the vehicle has entered a completely new region and should start a new map, and third, that the vehicle has re-entered a previously mapped region and should switch to an old map. We call these last two cases segmentation (new map) and matching (old map), respectively. In the next few sections, we describe our methods for segmentation and matching in detail.

4.4 Segmentation

Intuitively, we want to segment the world into small submaps that have the property that when the robot is within a submap, it can see most of the contents of the submap. Similarly, when the robot is within one submap, it shouldn't see much of any other submaps. Coincidentally, the submaps should be metrically accurate and certain: most of the metric uncertainty should be contained in the links between submaps. This intuitive description might work well for a structured environment, like a series of rooms connected by narrow doorways, but will obviously unravel in unstructured environments, like a woods.

We can use the concept of contiguous regions to determine submap segmentation. While the robot is within a contiguous region, its range sensors are likely to collect measurements which lie within the contiguous region, and unlikely to collect measurements in different regions. This property is also called simultaneous visibility or overlap Blanco et al. [2006], and ties together two important characteristics. First, the contiguous region is highly observable (the robot will be able to sense most of the region at the same time) which means that standard SLAM will work well and there will be negligible mapping and position errors. Second, since there are relatively few measurements which span between different regions, most of the global trajectory uncertainty will arise in the transitions between regions. This bottom-up criteria for submaps arises from the structure of the environment and the inherent properties of the sensors, and only coincidentally will align with an anthropomorphic classification of doorways, rooms or hallways.

To broaden the intuition, Roman [2005], Estrada et al. [2005], and Paz and Neira [2006] discuss the important factors in determining when to start a new map. The goal is to balance the amount of noise inherent in the sensors against the gradually accumulating error in the dead-reckoned trajectory. So long as the dead-reckoning error is lower than the sensor noise, adding more information to the map will increase the accuracy with which it can be matched with other maps: the map must contain enough variation to be matched strongly.

Our situation is somewhat complicated because we do not simply use dead-reckoning while building submaps, but actually run the RBPF, from which we get multiple likely submaps for each segment. However, we can still use similar heuristics for deciding when to segment. One of the simplest is to periodically segment, under the assumption that the dominant source of error is from dead reckoning, and that the dead reckoning error rate is roughly constant. This is a bad assumption for vehicles that have maneuver-dependent error rates, such as turning versus driving straight.

We experimented with several different segmentation metrics based on either estimation motion error, or the predictive score. We discuss these two methods next.

4.4.1 Motion Error Segmentation

As mentioned above, a segment should have minimal internal position error. This is a straightforward proposition if we only consider the dead reckoning error: a position error model that accounts for the uncertainty incurred by different maneuvers can be used to segment when the position error reaches a threshold.

For a simple 2D kinematic vehicle model, dead reckoned position error is a factor of velocity

error v_{err} and heading rate error u_{err} integrated over time:

$$\text{position}_{\text{err}} \propto (\alpha_1 v_{\text{err}} + \alpha_2 u_{\text{err}})t$$

for some scaling coefficients α. When we incorporate vehicle dynamics, the error terms are functions of the vehicle state x: for example, the velocity error will be frequently be worse for a wheeled vehicle at high accelerations due to wheel slippage.

$$\text{position}_{\text{err}} \propto (\alpha_1 v_{\text{err}}(x) + \alpha_2 u_{\text{err}}(x))t$$

Accurately estimating these functions for different environments is a considerable task, especially when there are unknown biases like wind, ocean currents or wheel slippage. We simply use a pessimistic model that generally over-estimates the position error.

Baldly applying the dead reckoning error model would result in near-periodic segmentation, which ignores the fact that we run particle filter SLAM explicitly to correct short-term dead reckoning error. While it is possible to use the distribution of the particle cloud as an estimate of the position uncertainty when doing localization, it is a questionable technique when doing SLAM, and completely inapplicable when using segmented SLAM, in which the particle positions may be in different coordinate frames. Position error estimation in the segmented SLAM formulation necessarily entails entropy estimation. Conceptually, segmenting before the entropy grows too high makes sense, but even rough approximations for SegSLAM entropy are computationally expensive (as we'll see below).

We fall back on a slightly less pessimistic dead reckoning error model that takes into account the expected amount of improvement yielded by the SLAM system. For many vehicles, position error is dominated by the heading error u_{err} and as a result, the motion model-based segmentation metric will tend to favor segmentation after hard turns.

4.4.2 Predictive Score Segmentation

One definition of a good segmentation is that when the vehicle is in area a, submap θ_a accurately predicts the sensor measurements, and when the vehicle is in area b another submap θ_b predicts the measurements:

$$p(z_a|x_a, \theta_a) >> p(z_a|x_a, \theta_b)$$

and

$$p(z_b|x_b, \theta_b) >> p(z_b|x_b, \theta_a)$$

To turn this insight into a segmentation metric, we use an estimate of the probability of future measurements z' given the current map:

$$p(z'|x', \theta) \propto \text{predictiveScore}(z', x', \theta) = \prod_{n=1}^{\#_{z'}} p(z'^n|x', \theta), \tag{4.4.1}$$

which has the same form as the particle weighting equation 3.4.16. To use the "future" measurements z', SegSLAM must be run a few seconds in the past, so that its current maps are a

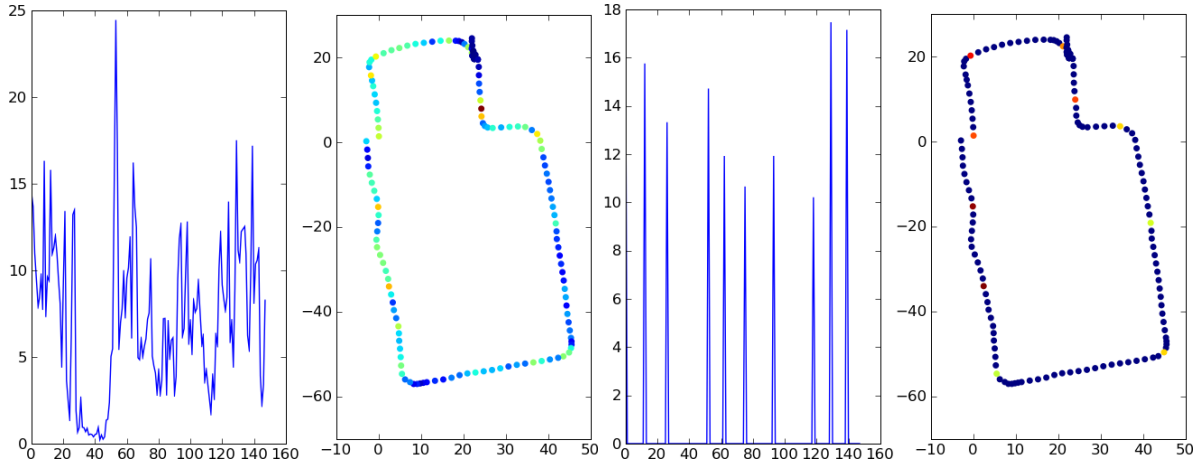

Figure 4.4: A portion of the Bruceton mine, showing the predictive score as a plot and a scatter-plot (left) and the resulting predictive score and segmentation points after a minimum segment size criteria has been enforced.

few seconds old. The future pose data x' is computed by running dead-reckoning on the future motion measurements u'.

The assumption is that when the predictive score decreases suddenly, the robot has left the current submap and SegSLAM should segment. The predictive score heuristic depends on two parameters: the length of the predictive window and the segmentation threshold.

Blanco et al. [2008b] has a more exact segmentation method using graph cuts, but then needs to reconstruct the maps, a slow procedure in 2D and an intractable one in 3D. We take the penalty of sub-optimal segmentation in exchange for real-time speed.

From our experience with the predictive score metric, we find that when the robot is travelling around a well-compartmentalized environment, predictive score clearly indicates advantageous segmentation points. But in large, open environments, predictive score degrades to periodic segmentation – there are no particularly advantageous places to segment.

4.5 Generating Local Metric Map Samples

Suppose that the robot explores an environment and segments several times. SegSLAM will have a constructed a segmented map: for each segment, which can be identified by its starting and ending times, there will be multiple particle maps (Figure 4.5). Different combinations of temporally compatible segments can be stitched together to form complete trajectories – this is like sampling from the distribution of all plausible maps that are encoded in the segmented map. In this section, we describe how to generate samples from the segmented map.

The SegSLAM particle filter has multiple particles, each of which has its own trajectory (and map), and each of which may segment or match at different times. After a period of time, the filter will have a large collection of map segments for different particles at different times. In order to search for loops, or matches, between these segments, we must reconstruct the relative transforms between segments. This is a simple process as long as we only need to chain segments

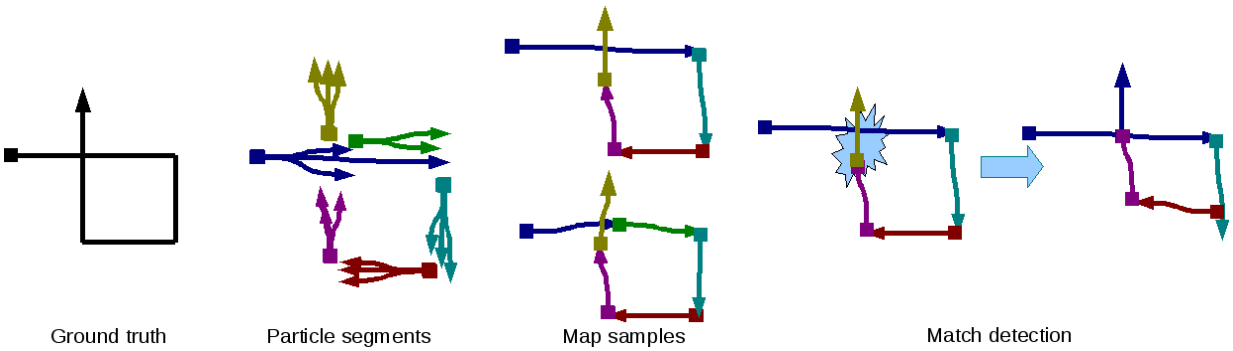

Figure 4.5: Using just the vehicle trajectories (rather than the submaps that are actually used) this figure illustrates the process of segmentation, map sample generation, and matching. Note that after matching SegSLAM does not enforce global consistency between the red and turquiose segments.

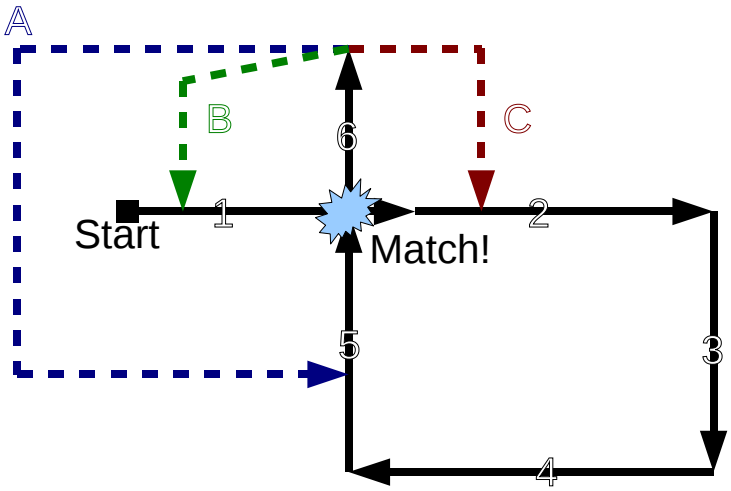

Figure 4.6: A transform reconstruction scenario: after building segments 1,2,3,4, and 5, the robot reconstructs its relative trajectory by chaining backwards through the segments and then uses map matching to detect a loop closure 5-1. It then proceeds on segment 6, and again needs to reconstruct relative trajectories so that it can properly recognize another match if it takes any of paths A, B, or C.

backwards through time. But when matches are detected, the reconstruction process should take advantage of these new connections, and the process becomes more complex: see Figure 4.6.

Rather than implicitly storing all of the transforms between all of the segments (or even temporally-sequential segments), we use the following data structures:

Segment A Segment stores the following information: a reference to the octree map, the start and end times of the map, the start and end transforms that correspond to those times, and importantly, a link to the parent segment, if the segment is a result of a match. For debugging purposes, the entire particle trajectory through the segment may be stored.

StartMap and EndMap These data structures map from a moment in time to a list of all the Segments that start or end (respectively) at that time.

When a segmentation occurs, a new Segment with a new octree map is created for each particle, and kept up to date as the particles change due to resampling and as time progresses. When a particle segments again, the old segment is entered into the StartMap and EndMap, and a new one is created. The start transform T_s for a new segment is simply the identity matrix, while the end transform T_e is the particle pose at the end of the Segment.

Likewise, when a match occurs, a new Segment is created for the matching particle. In this case however, the Segment starts with a reference to the matched octree, and with T_s equal to the particles matched position within the coordinate frame of the matched octree. Recall that our octree representation supports deferred reference counts and copy-on-write – so the old map (which belongs to some older Segment) is not altered. Further, the old Segment is linked to the new Segment as its parent. This allows us to later recover the parent's start and end times and transforms for expansion.

Consider a particle's trajectory over time: $[x_0^{(i)}, \ldots, x_t^{(i)}]$. We can index that same trajectory by the segments: $[x_\alpha^{(i)}, \ldots, x_{\alpha'}^{(i)}, x_\beta^{(i)}, \ldots, x_{\beta'}^{(i)}, x_\gamma^{(i)}, \ldots, x_{\gamma'}^{(i)}, \ldots]$, where x_α is the first pose of submap α and $x_{\alpha'}$ is the last pose. For each of the segments α, β, γ, the particle poses in the segment are in the segment's coordinate frame.

To reconstruct the relative transforms between segments, we need to compute the transforms for each segment into some arbitrary frame. For example, if we are chaining backwards from the native frame of segment γ,

$$X_\gamma = [T_{\alpha'}^{-1} T_{\beta'}^{-1} x_\alpha, T_{\beta'}^{-1} x_\beta, x_\gamma] \tag{4.5.1}$$

or chaining forwards from the native frame of segment α

$$X_\alpha = [x_\alpha, T_{\alpha'}^{-1} x_\beta, T_{\beta'}^{-1} T_{\alpha'}^{-1} x_\gamma] \tag{4.5.2}$$

When searching either forwards or backwards through time, the series of segments forms a unique search order according to depth from the start segment. But if there are loops (due to matches), then we must do breadth-first expansion of the segments, randomly selecting the order in which we expand segments with the same search depth.

As segments are considered, it is important to remember that they are describing the vehicle's trajectory over time. This description may have temporal gaps, but should not have any overlaps. Thus each segment is checked for temporal compatibility before being expanded.

A related concept is *viability*: a segment is viable if it is compatible with at least one of the current segments. A segment can only fit into a reconstructed trajectory if there is a set of temporally sequential segments from the present to the segment (including the possibility of temporal jumps from matches). Due to the resampling process of the particle filter it is possible for a segment, or an entire ancestry of segments, to become non-viable, in which case they should be discarded because they can never be part of a reconstructed map that includes the present.

Recursively monitoring viability could become expensive, but is not in practice because a full segmentation or match across all the particles acts as a firewall – any particle that is viable at the time of the full segmentation will always be viable.

As shown in Figure 4.6, there are three trajectories that must be considered during the breadth-first expansion of the relative transform graph: forward and backward in time for the match, and the natural ancestor of the immediately prior segment. Further complicating the situation is the possibility that the match 5-1 can be discovered as the graph is grown either backwards in time (from segment 6 in the figure), or forwards in time, for example if somehow there were a match from a new segment 7 to segment 4, which was then itself expanded forwards. In cases when there are multiple segments at the same depth, the expansion order is randomized since choosing a segment excludes other segments that are temporally incompatible, and we want to generate a random sample from all such reconstructions. In a more nuanced best-first strategy, other factors, such as match score, could be used to break ties in depth – the motivation behind both breadth-first and best-first expansion being to more accurately sample from the underlying distribution of local metric maps. In fact, there are interesting connections between sampling from the segmented map and generating a random sample from a Generic Markov Chain, particularly generating a random spanning tree of a directed graph Propp and Wilson [1998].

4.6 Matching

Matching can be thought of as loop closure, overlap detection, or map re-entry. Fundamentally, it is the realization that that the current environment matches a place that has been seen before. In Section 2.5 we discussed methods for matching octree evidence maps together. In the previous section, we described how to sample local metric maps from the segmented map. Our approach for finding matches is then to periodically generate local metric maps, search them for likely match candidates, and then attempt to match to the candidates. Matches can be verified using the match score metric of Equation 2.5.2 and weighted accordingly. The algorithm then uses the weights to stochastically select from among the matches, and a match Segment is created as needed (see above).

4.6.1 Winnowing Match Candidates

We use a cascade of criteria to try to discard as many candidates as possible as quickly as possible. The first criteria, temporal separation, is intended to reduce hysteresis. This implies the assumption that the robot will not actually jump back and forth between segments very quickly. The second criteria, spatial proximity, queries several voxels near the current robot position in the candidate map (using the candidate transform) to see if they have any occupancy information, a quick check that the two maps overlap or are close to overlapping. After these two simple tests, there are rarely more than one or two candidates remaining.

4.6.2 Matching to Candidates

After winnowing the set of candidates, the robot's recent perceptions are matched to the candidates using one of the map matching methods from Section 2.5. Specifically, a map is built from the most recent few seconds of data (recall that SegSLAM runs a few seconds in the past), and

then this small map is matched with the candidate maps. The map matching method that we tended to use was ICP based on the octree-binned pointcloud, which is a very fast method that reduces the influence of point density. As state above, the weight for a particular match transform T_m can be estimated using the match score metric. Another, faster, method is to use the mean ICP nearest neighbor error. We also weight transforms with a smaller translational component τ_m to be more likely than large transforms:

$$w(T_m) \approx p(\text{err}_{\text{ICP}}|T_m)\ p(\tau_m^2)$$

To verify our map-based matching methods, we tested finding matches with particle filter localization. Since each transformation in the local map sampling process adds some uncertainty, we can estimate the uncertainty in the candidate map positions: nearby candidate positions will be fairly certain, but candidate positions at the end of long loops will be uncertain (this is also reflected in the variation between local metric map samples). This uncertainty estimate is used to initialize the variance in a particle cloud, centered around the vehicle's position in the candidate map coordinate frame. Then the past few seconds of robot perceptions (motion and range measurements) are used to localize the robot within the candidate submaps. The weight of a match transform derived in this way is a combination of the final variance of the particle cloud and the particle errors (as opposed to normalized weight), yielding a strong indicator when the particle filter converges to a good solution.

$$w_{\text{loc}}(T_m) \approx p(\text{err}_X)\ p(|\Sigma_X|)\ p(\tau_m^2)$$

where $\text{err}_X = (z - \hat{z})^2$ is the particle error and $|\Sigma_X|$ is the determinant of the covariance of the particle positions.

In this case, the matching particle filter is completely separate from the SegSLAM particle filter: it is created for the purpose of evaluating a single match and discarded afterward.

4.6.3 Discarding Segments

In the long-term case, after thousands of segments have been created, it may become desirable to throw some of them away. In particular, if the robot is travelling between two rooms, repeatedly matching back and forth between two families of Segments, there will come a time when the segment maps will be complete, and when all of the parent Segment maps (which record the intermediate states between a blank map and the complete map for each visit to the room), can be discarded. This is equivalent to saying that the robot has no uncertainty about the topology between the two rooms: the reason that it is important to keep around all the original parent segments is to allow for alternate topologies, for example the possibility that the robot really moved into a third room.

If the room maps are complete and there is very low topological uncertainty between them, the maps can be merged, particles switched to localization mode while it is in the merged area, and all the ancestral Segments can be discarded. Completeness can be estimated from the map entropy, and topological uncertainty from either the approximate match metrics or the gold standard match score.

4.7 SegSLAM Entropy

The distributed nature of the SegSLAM map makes entropy estimation even more difficult than RBPF entropy. We could imagine extending the idea of Blanco et al. [2008a] of estimating the entropy of the averaged map by drawing multiple map samples from the segmented map, and computing the expected total metric map entropy (either average of entropy or entropy of average). This would be computationally intensive: O(NSV) for N samples of S submaps of V voxels.

Alternatively, we could use the map factorization given in the derivation of SegSLAM together with the definition of conditional entropy to write

$$\begin{aligned} H(p(\Theta)) &= -E[\log(p(\Theta))] && (4.7.1)\\ &= -E_T[E[\log(\prod_{\theta\in\Theta} p(\theta|T))]] && (4.7.2)\\ &= -E_T[E[\sum_{\theta\in\Theta} \log(p(\theta|T))]] && (4.7.3)\\ &= -E_T[\sum_{\theta\in\Theta} E[\log(p(\theta|T))]] && (4.7.4)\\ &= E_T[\sum_{\theta\in\Theta} H[(p(\theta|T))]] && (4.7.5) \end{aligned}$$

where Θ is the entire segmented map and T are the transforms between the segments θ. We can approximate this with the segmented map in Monte Carlo fashion as

$$H(p(\Theta)) \approx \sum_{T_i^j\in\Theta} p(T_i^j) \sum_{\theta\in\Theta} H[p(\theta|T)]. \quad (4.7.6)$$

If we assume that $p(T_i^j) = 1/\#_{T\in\Theta}$, and observe that T_i^j only influences two submaps, then we can write

$$H(p(\Theta)) \approx \frac{1}{\#_{T_i^j\in\Theta}} \sum_{T_i^j\in\Theta} H[p(\theta_i,\theta_j|T)]. \quad (4.7.7)$$

The rightmost term can be decomposed as

$$H(p(\Theta)) \approx \frac{1}{\#_{T_i^j\in\Theta}} \sum_{T_i^j\in\Theta} H[p(\theta_i)] + H[p(\theta_j)] - H[p(\theta_i\cap\theta_j|T_i^j)], \quad (4.7.8)$$

where the entropy of each segment $H[p(\theta_i)]$ is the evidence grid entropy as described in Section 2.6, and $p(\theta_i\cap\theta_j|T_i^j) = \text{MatchScore}_{T_i^j}(\theta_i,\theta_j)$ is the match score after applying transform T to θ_j.

Using either of these methods, we are beginning to layer heuristics on heuristics to come up with a questionable estimate of entropy. In the next chapter, we will investigate methods for probing the SegSLAM entropy without explicitly carrying out these laborious (and ultimately inaccurate) entropy estimates.

4.7.1 Global Metric Maps Via Entropy Minimization

As we have asserted before, it is not necessary for the robot to resolve a globally metric reconstruction of all of its segments in real-time, but rather only a relatively metric reconstruction of a subset of the segments – a fact that we exploit to limit onboard computation. On the other hand, a globally metric map is useful and desirable, even though it may be computationally intensive to create.

There are many approaches to this problem, which is the offline SLAM problem, but they necessarily reprocess all the data from scratch. For example, our map matching methods are clearly related to Lu and Milios [1997]'s scan matching – we could generate a map sample by picking the most likely segments (based on particle weight) and transforms, and then solve the resulting nonlinear optimization problem to find an optimal global map. Putting that idea to one side, we would like to use the SegSLAM framework to reconstruct a good global map using only the matches that were detected at run-time. This approach is meant to take advantage of the computation that was done at run-time to produce a fast and yet reasonable global metric reconstruction.

One brute force method is to generate many global map samples (a complete trajectory built from the alternative segments), construct a global map for each sample by either merging the submaps or even building the map from scratch, and then evaluating the global map entropy and searching for the map with the lowest entropy. Merging together partially overlapping sub-maps is a simple matter of traversing one octree, transforming the coordinates of each leaf to the frame of reference of the other octree, and summing the appropriate cells. While fully traversing the source octree is computationally expensive, it is quicker than building the map from scratch.

The idea behind searching for the entropy minimizing global map is that entropy reflects the conciseness and certainty of the map, rewarding accurate alignment of the overlapping segments. For example, in cases with closed loops, the entropy will be significantly lower for map samples that properly include the loop.

We ran a simulated experiment in global metric map selection via entropy minimization. In this experiment, the simulated vehicle traveled around a box of four straight corridors. We ran SegSLAM with 10 particles and a large amount of motion model noise, with the result that some of the particles detected the loop closure while others did not. This yielded a wide variety of map topologies (Figure 4.7). Applying the brute force method, we generated 100 trajectory samples, merged them into a single global map (either by constructing the maps from scratch, or by merging the segment maps), which were searched according to the minimum entropy criteria, as summarized in Table 4.1.

Method	Runtime (s)	Min Entropy
Simulated ideal		1427
Average sample		1518
From scratch	384	1471
Map merging	120	1461

Table 4.1: Global map entropy after entropy minimization search. From a simulated corridor environment with 10 particles, 10 samples per particle.

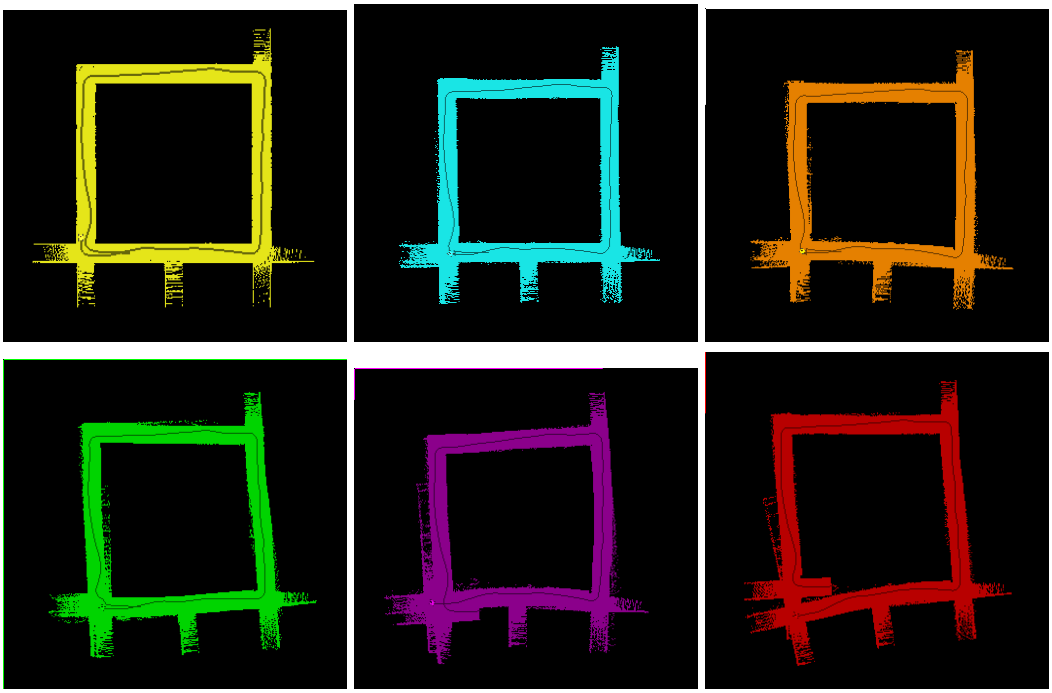

Figure 4.7: Example map samples from the segmented map. The top row from right to left consists of the ideal map (from the simulated data), the best map from the min entropy search building the maps from scratch, and the best map from min entropy search using map merging (entropies are given in Table 4.1). The second row consists of random representative samples.

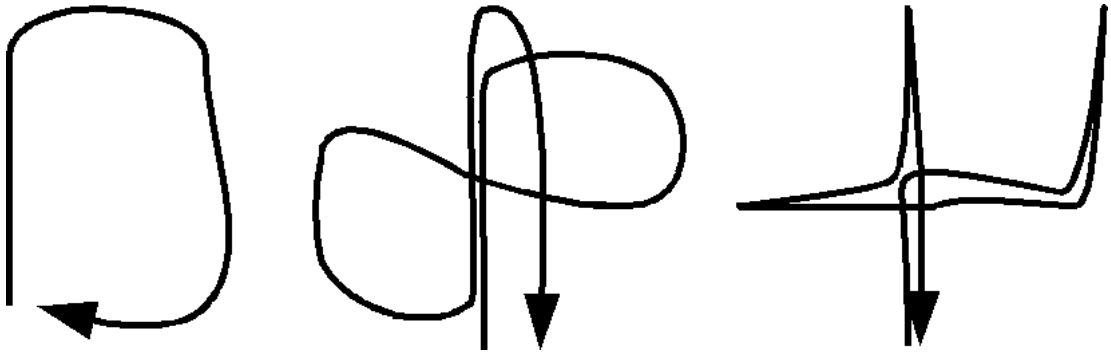

Figure 4.8: We investigate the application of different metrics to characterize SLAM performance in three broad topological classes: a single loop, multiple intersecting loops, and a star with multiple legs.

4.8 Subterranean Robot

4.8.1 Cave Crawler

Cave Crawler is an autonomous mobile robot that was designed to explore and map abandoned mines Morris et al. [2006] (Figure 4.9). Cave Crawler uses a Crossbow 400 IMU and wheel odometry as position measurements, and the mapping sensors are forward and backward-looking

Figure 4.9: The autonomous mine mapping robot Cave Crawler.

SICK LMS 200 laser range finders mounted on spinning jigs that rotate around the vehicle's forward-backward axis (roll). In many cases, only the front laser is used because the robot is followed by attendants who corrupt the rear-looking data. A crucial distinction between the Groundhog and Cave Crawler datasets described in Chapter 2 and the experiments in this chapter is that in the previous datasets, the vehicle came to a complete stop to collect its 3D data, whereas in this chapter all the datasets involve (almost) continuous movement. This significantly complicates the dead reckoning and sensor calibration problems, which we discuss in Appendix A.

By examining the laser data collected while the vehicle is stationary we can correct for roll skew and misalignment as described above. But when the vehicle is moving we must track the position and orientation of the entire robot in order to accurately register the laser data. The first step in that direction is dead reckoning: integrating an estimate of the vehicle's attitude and velocity to yield the full 6DOF pose – this is identical to the particle prediction step with no added error. Dead reckoning error is dominated by yaw error, which is significant because Cave Crawler lacks the KVH yaw gyro onboard Groundhog. In response, we developed a Extended Kalman Filter for vehicle attitude (derivation for a generic 6DOF IMU in Appendix B). Our EKF implementation exploits brief vehicle stops to estimate gyro bias rate corrections, as well as continous gravity-based angle corrections. Depending on the terrain, this led to position error on the order of $\sim 5\%$ of distance traveled.

4.9 Experiments

Characterizing SLAM performance is challenging, especially in situations without accurate ground-truth. We present three different methods for evaluating and comparing SLAM and SegSLAM,

Figure 4.10: A photo of the two parking garage entrances used: the vehicle exited the upper entrance and entered via the lower entrance.

and illustrate each method with an experiment with the Cave Crawler robot from a different site. The first method is to simply see if the algorithm properly detects a loop closure, which is illustrated by a multi-level loop from the Collaborative Innovation Center (CIC) parking garage. The second method is to subjectively examine a large map with many loops, to see if there are inconsistencies or obvious flaws, this is illustrated with a dataset from the Bruceton mine. The third method is to search for the minimum entropy map (as described above), and we illustrate this method with a dataset from a portion of the Gates building collected during its construction. These three datasets also correspond to different topological classes: a single loop, multiple intersecting loops, and a star of out and back legs (Figure 4.8).

4.9.1 Loop Closing Error – CIC Parking Garage

In this experiment, we examined the position error after the vehicle returned to (near) its start position – the loop closure error. The Collaborative Innovation Center parking garage on the CMU campus is a convenient, multi-level structure with three exits on different levels, which allows Cave Crawler to traverse 3D loops. In the dataset used for this experiment, Cave Crawler drove up a ramp from the first level to the second level, went around a tight loop at one end of the second level, and then drove out the second level exit and back into the first level entrance, for a total distance of 303 m (Figures 4.10, 4.11).

As discussed above, one method for evaluating SLAM and SegSLAM performance is to look at loop closure error. In our regular RBPF, we can generate a position estimate from the particle cloud by taking a weighted average of the particle positions. This approach usually won't work with SegSLAM: when the particles are in different segments they are in different coordinate frames. However, in simple cases, such as the short loop used here, all the particles do successfully and accurately match, meaning that they all return to the same coordinate frame.

To compare the performance of RBPF SLAM and SegSLAM, we ran each approach 20 times

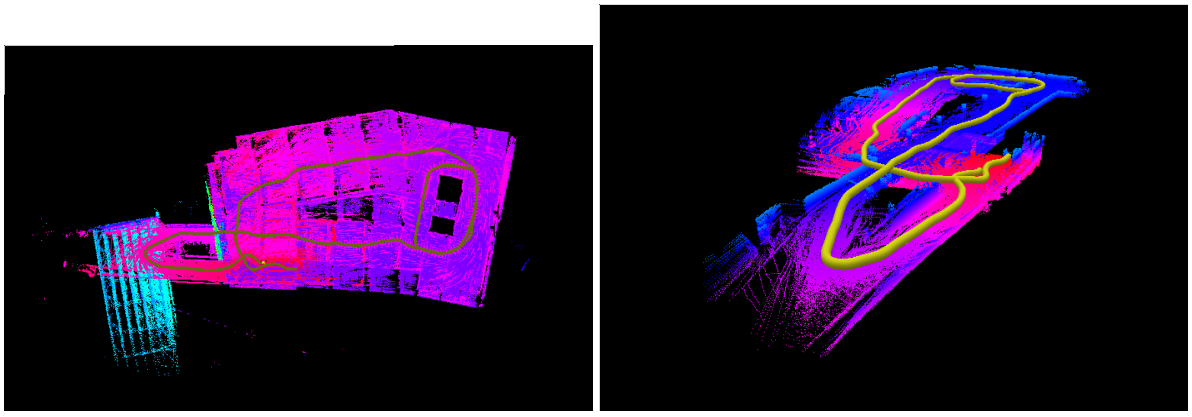

Figure 4.11: Left: a top isometric view of the parking garage dataset, showing the vehicle path, which includes two loops on two different levels. The prominent structure on the left is a bridge outside the parking garage. Right: a perspective view of the parking garage entrances on two levels, showing the vehicle path.

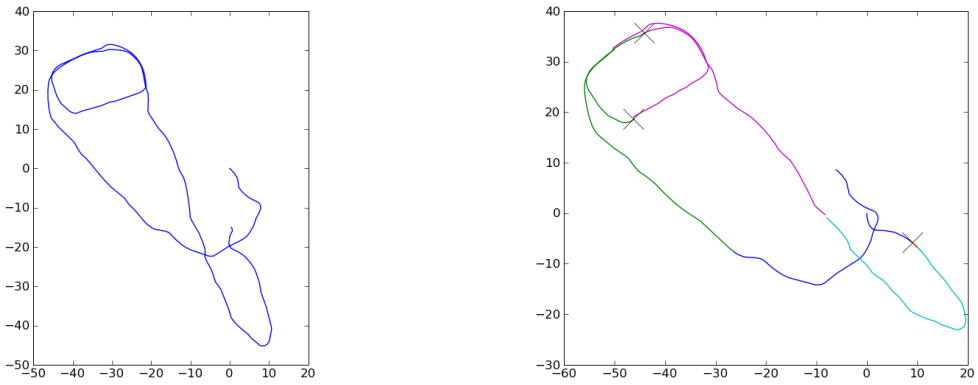

Figure 4.12: Left: dead reckoning for the parking garage dataset, showing the significant position error. Note that due to automobile traffic, the vehicle did not return exactly to its start position. Right: a reconstructed SegSLAM trajectory, showing how SegSLAM segmented the dataset and correctly detected matches (indicated by X's) and re-entered previously mapped segments (as indicated by the segment coloring).

on the dataset, each time computing the weighted average position from the final particle cloud. We then computed the mean and standard deviation of the final position error over the 20 runs, and repeated this process for a variety of particle counts. As shown in Table 4.2, the dead reckoning error of the original data is significant due to yaw bias (Figure 4.12), and the RBPF manages to close the loop after about 40 particles. But SegSLAM is able to reliably match very accurately even with just one particle. Surprisingly, the matching does come at some computational cost for low numbers of particles, as shown by the run times in Table 4.3, but for higher numbers of particles, SegSLAM is actually faster! Since SegSLAM adds the segmentation and matching steps to the RBPF, this may be explained by the fact that manipulating the segmented octree maps is

Particles	RBPF SLAM μ/σ	SegSLAM μ/σ
1	9.3 / –	0.7 / –
5	3.0 / 6.9	0.9 / 0.6
10	0.9 / 4.5	0.7 / 0.5
20	0.4 / 3.8	0.6 / 0.2
40	0.9 / 1.5	0.5 / 0.2
100	0.8 / 1.1	

Table 4.2: Mean and standard deviation of final particle filter pose (calculated as the weighted mean of the particle cloud) over 20 runs for different numbers of particles.

Particles	RBPF SLAM runtime (s)	SegSLAM runtime (s)
1	2.6	4.2
5	11	13
10	22	26
20	47	41
40	98	83
100	280	

Table 4.3: Runtime in seconds for the parking garage loop dataset.

faster than manipulating the full maps (although there is no difference in the octree depths, voxel dimensions, etc. between the maps).

4.9.2 Map Goodness – Bruceton Mine

Bruceton Mine is a research mine near Pittsburgh, PA, and a common location for Cave Crawler tests. The dataset used here was collected by the subterranean robotics team on May 14, 2007, and comprises a 1300 m traverse through the mine, including several loops (Figure 4.13). This site has also been described by Thrun et al. [2003].

As discussed above, one method for evaluating SLAM and SegSLAM performance is to just look at the maps and judge their "goodness". For a traditional RBPF, this is fairly simple, since even though there are many maps (one per particle), it is rare that the maps differ significantly except in the last few hundred meters, so if one succeeds they all will. For SegSLAM, the case is different because we can only draw samples from the segmented map. If only one sample out of a thousand is a good map (even if it is very good) can SegSLAM be considered to have succeeded? After all, the statistical likelihood of SegSLAM sampling that particular map are only one out of a thousand! At the same time, SegSLAM deliberately forgoes global metric accuracy in favor of speed and relative metric accuracy: it may be that although there is no map sample that looks "good", SegSLAM will properly matches between segments and never get lost – the true definition of success for a SLAM system.

RBPF SLAM never successfully closed all the loops, even with 1000 particles (taking 2.8

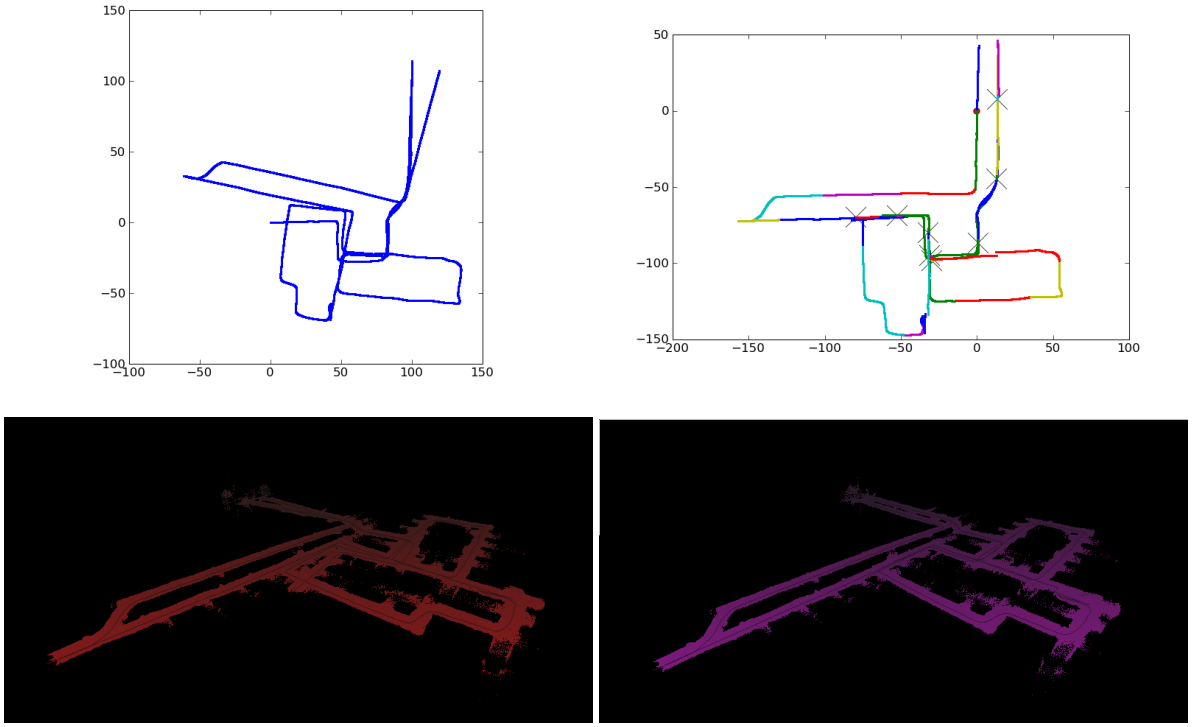

Figure 4.13: Comparison of the dead-reckoned path (left) with a sample SegSLAM path (right). Segments are color coded such that nearby lines with the same color indicate that a particle re-entered a prior segment (a limited palette means that distant lines may share the same color as well). X's mark matches/re-entries. The second row shows a perspective view of the resulting laser pointclouds.

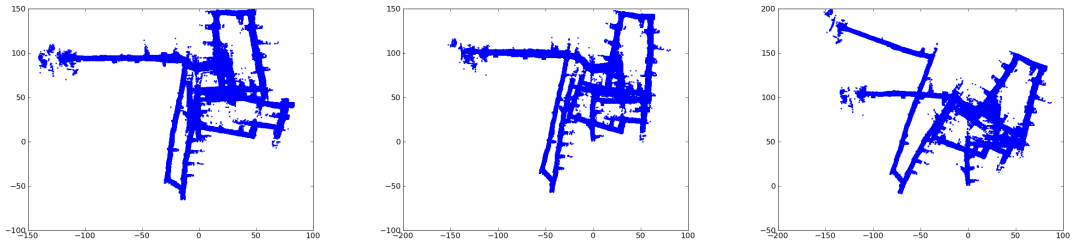

Figure 4.14: Example maps showing RBPF SLAM's failure on the Bruceton dataset, with 200, 500, and 1000 particles.

hours of computation): it simply could not deal with the many interlocking loops and even the best outcomes mistakenly merged two parallel tunnels (Figure 4.14). SegSLAM reliably yielded a good map with as few as 20 particles, largely because it could treat the tunnel segments as unique features (Figure 4.13) and was effectively doing feature detection in a sparse environment.

4.9.3 Minimum Entropy Map – Gates Building

We have discussed why standard metrics, such as average particle position and map "goodness" are difficult to apply to SegSLAM, at least in complex environments. The final method that we use for evaluating SLAM performance is to look at the minimum entropy global map. SegSLAM is at a distinct advantage to RBPF SLAM, since its segments encode an exponential number of possible maps whereas SLAM can only offer one map per particle. In a sense, we are searching to see if "good" maps have any support in the distribution over maps represented by the segmented map. But since generating and evaluating hundreds or thousands of map samples is computationally expensive, minimum entropy search is necessarily an offline operation.

Cave Crawler mapped out a portion of the Gates building, travelling 892 m over three different levels (Figure 4.15). This dataset was collecting during the construction of the building, and while it is clearly a man-made environment, there was a large amount of construction-related clutter, missing walls, etc., which gave it less structure than might be expected.

We ran RBPF SLAM on the Gates dataset with 20 to 800 particles, and found that 200 particles, which ran in just under three hours or about twice real-time, were sufficiently to generate consistent maps in which each leg of star topology of was aligned with itself and the SLAM position estimate accurately returned to the start position. However, because there were no real loops in the dataset, each of the legs of the star had some relative error to the other legs, yielding a gradual misalignment between the levels of the building (Figure 4.17).

SegSLAM with 40 particles ran in 693 seconds, and after an offline search for the minimum entropy map yielded the map shown in Figure 4.17. The SegSLAM map entropy was 10755, compared with the RBPF SLAM map entropy of 13561.

4.9.4 Closing Overview Loops

One of the reasons that SegSLAM outperformed RBPF SLAM on the Gates building dataset was that although the vehicle path was a star topology (Figure 4.8), there were several cases when the vehicle was able to look across between ramps or down from an overview to a region that it had previously explored. SegSLAM was able to correctly match these overview loops, which we illustrate with an example from the point in the Gates dataset (Figure 4.18).

In this excerpt from the full Gates dataset, the vehicle drove from the bottom, up and around a ramp to the left, and then looked down from the top. As shown in Figure 4.15 SegSLAM detected two matches, the first looking across between the ramps and the second looking down from the top of the ramp.

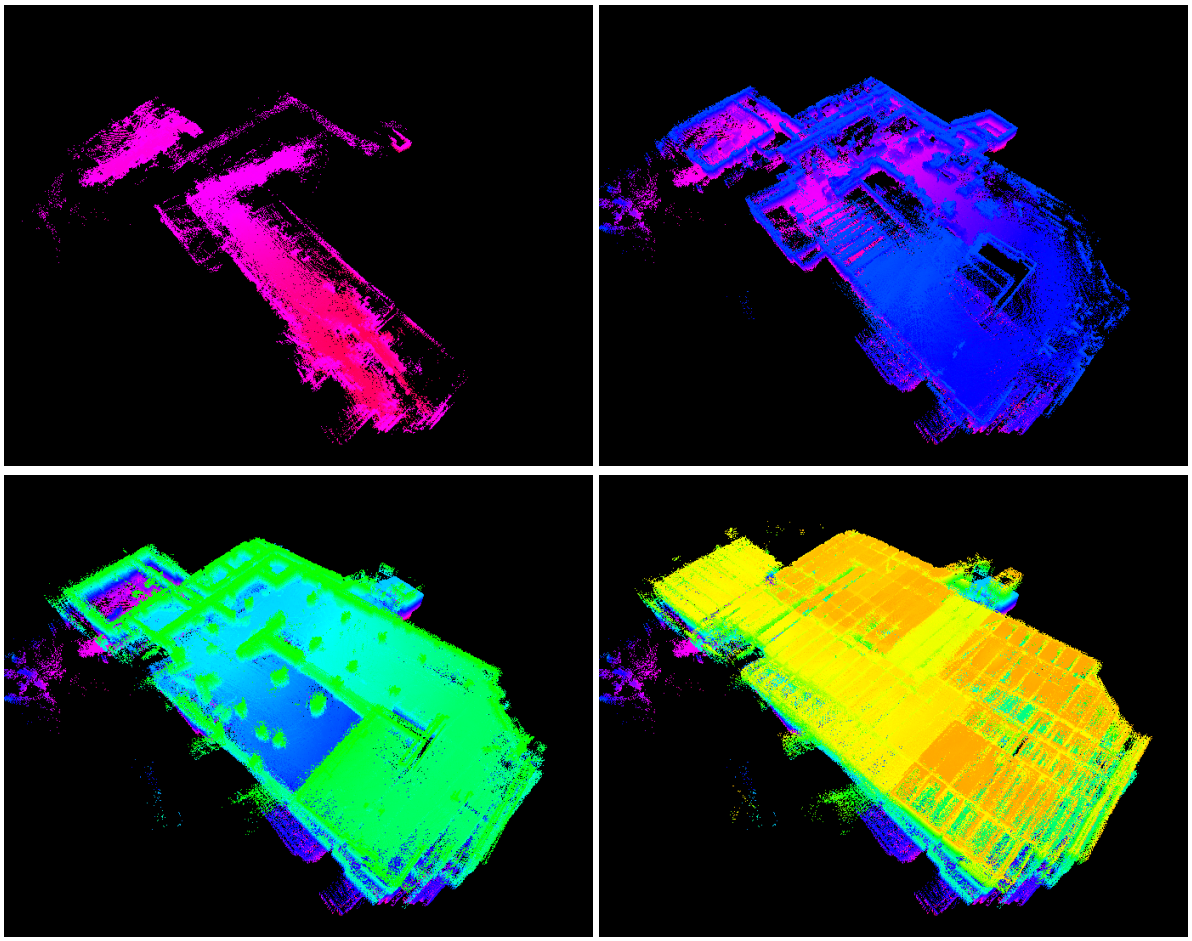

Figure 4.15: Slices through the Gates building, showing some of the internal structure of the three levels. The main axis of the building is about 80 m.

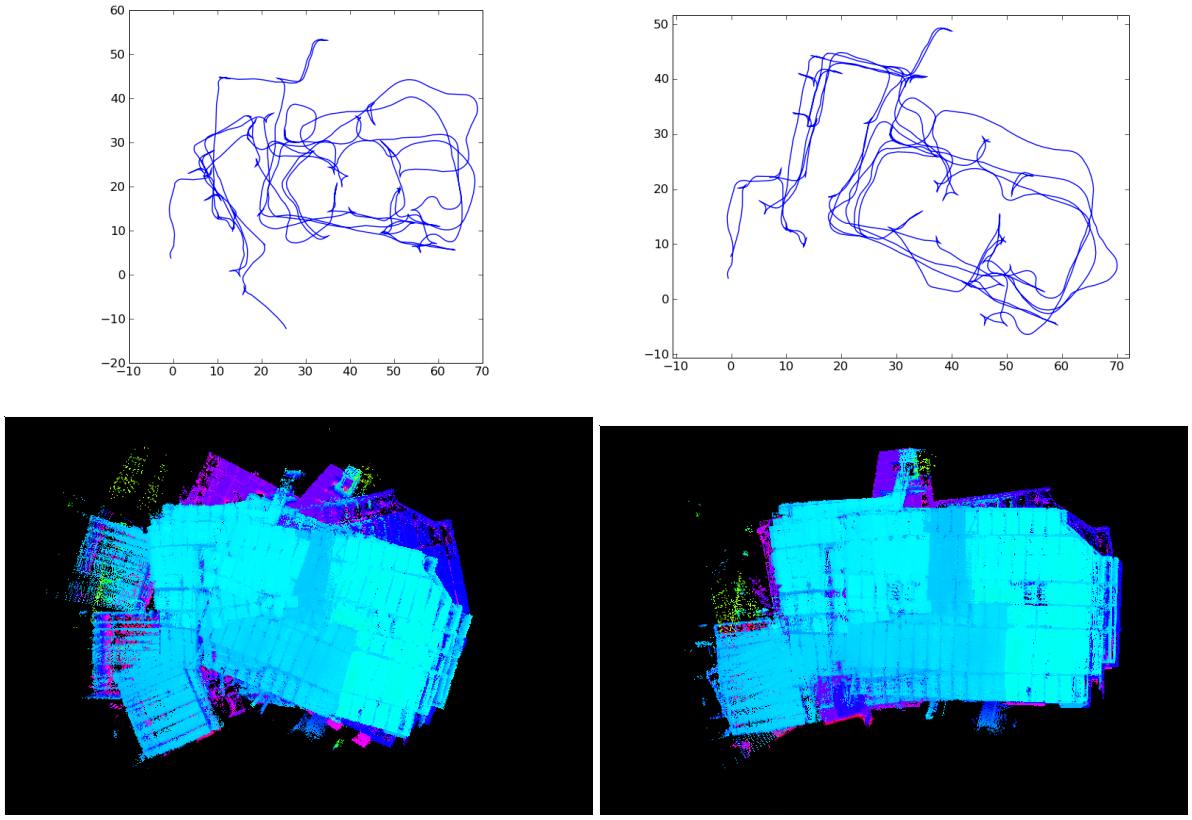

Figure 4.16: Raw dead reckoning (left) and calibrated dead reckoning using offline estimate of the heading bias (right) trajectories and pointcloud for the Gates dataset. Even with the optimal constant heading bias, there is still obvious misalignment between the three levels of the building.

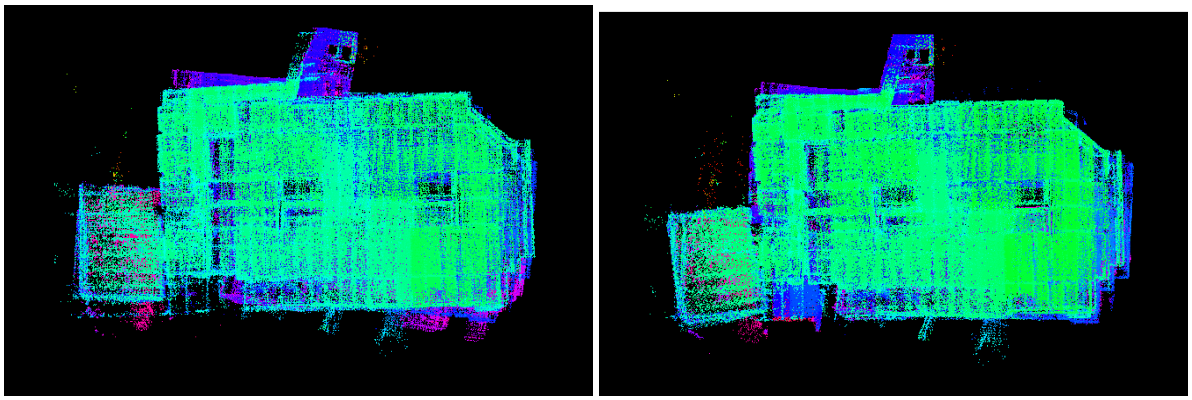

Figure 4.17: Left, the map from 200p RBPF SLAM, which took almost 3 hours to run. Right, the best map from an offline entropy minimizing search of the segmented map after running SegSLAM with 40 particles, which took under 12 minutes. Both maps show some misalignment between the levels of the building, but the SegSLAM map is distinctly better aligned, which is reflected in the respective map entropies of 13561 and 10755.

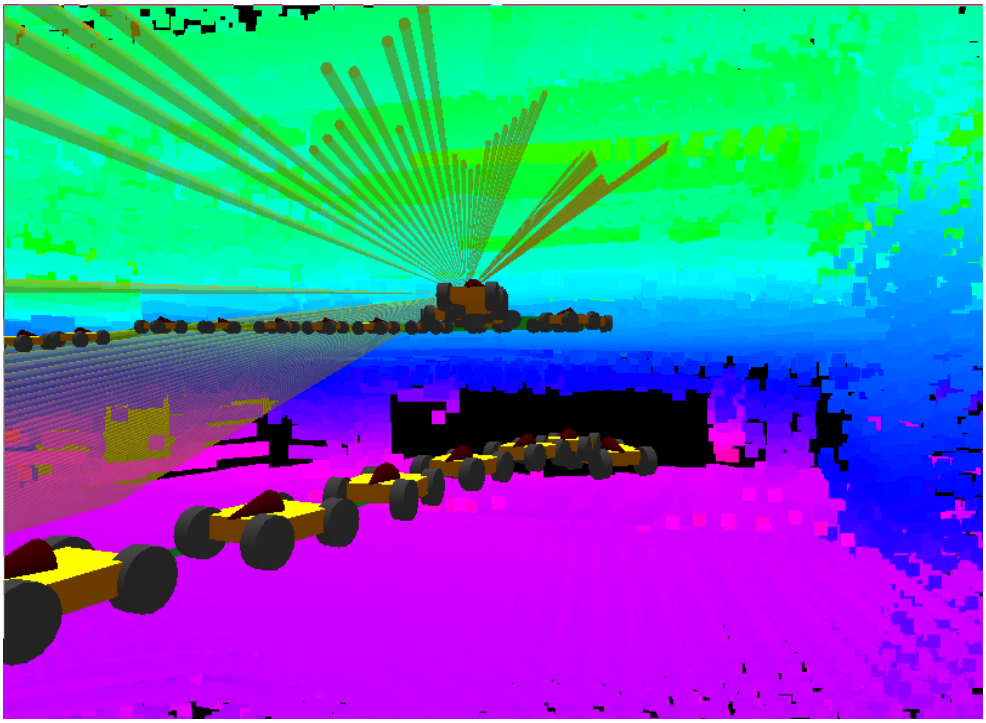

Figure 4.18: A rendering of the overview situation: the vehicle drove from the bottom, up and around a ramp to the left, and then looked down from the top. The laser beams from a single scan are rendered from the overview point.

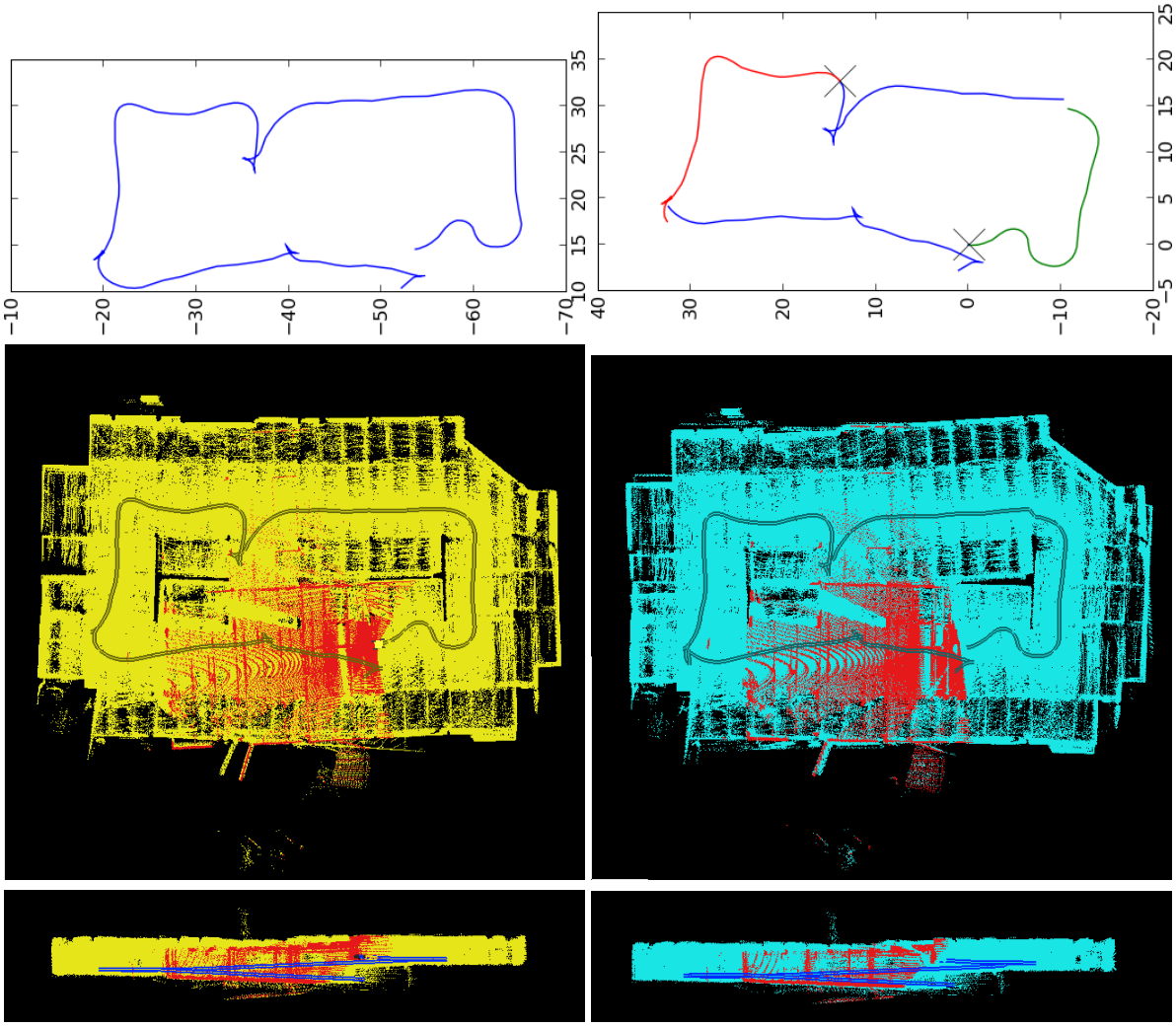

Figure 4.19: An instance of overview loop closure by SegSLAM, left is dead reckoning, right is SegSLAM. Top row: vehicle paths, with SegSLAM segments indicated by colors and matches indicated by X's. Note that SegSLAM detects two matches, the first looking across between the ramps and the second looking down from the top of the ramp. Middle row: top-down view of pointclouds, with the last few seconds of laser data rendered in red to highlight the misalignment in dead reckoning and the correct match in SegSLAM. Bottom row: side view of pointclouds and vehicle path, showing that the vehicle climbed about 4 m and was looking out over a previously mapped region when SegSLAM closed the loop.

4.10 Statistical accuracy and consistency

Estimators, such as the SLAM estimate of position, can be evaluated using statistical tests to see if they are consistent with ground truth, when it is available. In particular, we would like to see if the SLAM algorithm accurately estimates its own uncertainty. One of the most common statistical tests is the Wald test [Wald, 1943] which considers the two hypotheses:

$$H_0 : x = x_0$$
$$H_1 : x \neq x_0$$

for some scalar parameter x. Assuming that $\hat{x}$, the estimate of x is asymptotically normal, then

$$\frac{(\hat{x} - x_0)}{\hat{\sigma}} \xrightarrow{D} N(0, 1),$$

where $\hat{\sigma}$ is the sample standard deviation and $\xrightarrow{D}$ indicates convergence in distribution. The Wald test defines

$$W = \frac{(\hat{x} - x_0)}{\hat{\sigma}},$$

and the size α Wald test rejects the null hypothesis H_0 when $|W| > \Phi^{-1}(1 - \alpha/2)$, where $\Phi^{-1}()$ is the inverse CDF of the standard normal distribution. The Wald test indicates whether the difference between an estimate and ground truth is statistically significant.

Using the Wald test is equivalent to checking whether the value x_0 is within the confidence interval $C = (\hat{x} - \hat{\sigma}\Phi^{-1}(1-\alpha/2), \hat{x} + \hat{\sigma}\Phi^{-1}(1-\alpha/2))$. This explains the standard convention of evaluating the consistency of an estimator by checking to see if its estimate is within a standard deviation of the true value.

When ground truth is available, we would like to consider the hypothesis that the particle filter estimate of the vehicle position is statistically equivalent to the ground truth position. In situations when the particle cloud is well described by a multivariate normal distribution, we could directly apply the confidence interval criteria or the Wald test to evaluate this hypothesis.

One problem with this approach is that particles interact (due to resampling) and so are not statistically independent. Hypothesis testing assumes independent and identically distributed random variables, and so can't properly be applied. Further, in cases where the particle distribution is significantly multimodal, the normality assumption falls apart.

Due to these problems, we shift from classical hypothesis testing and find a Bayesian *credible* interval or region (contrasting with the confidence interval discussed above), that is likely to contain the ground truth value. If we choose this region to be as small as possible, also called the highest posterior density (HPD) region, we will see that it yields equivalent test results to the confidence interval of the Wald test in the unimodal case, but that it generalizes to multimodal distributions.

As a first step to finding the HPD region, we estimate the continuous probability distribution from the set of particles using a standard kernel density estimation (KDE) approach. The probability distribution of a set of samples from a random variable (the particles), $x_1, x_2, \ldots, x_N \sim f$ is approximated as

$$\hat{f}_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x - x_i}{h})$$

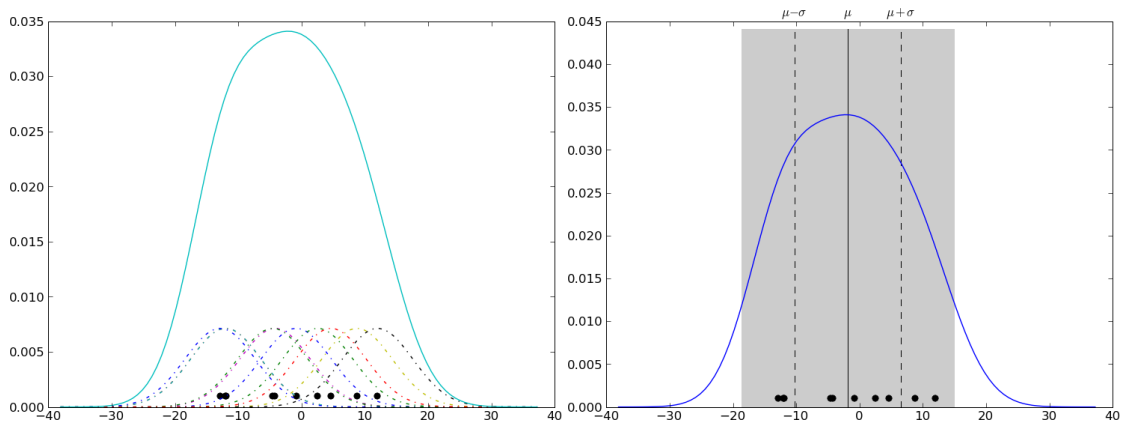

Figure 4.20: Left: Unimodal samples, each with a Gaussian kernel (dashed lines), which when summed together form the KDE of the continuous pdf. The right plot show the 2-σ confidence interval (1-σ interval shown with dotted lines).

where K is a kernel and h is the kernel bandwidth, or smoothing parameter. We use the standard normal kernel

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

The bandwidth h is chosen according to the rule of thumb proposed by Scott [1992]

$$h = N^{-\frac{1}{D+4}}$$

where N is the number of data points, and D is their dimensionality.

The KDE method for estimating f from the particles applies to both unimodal (Figure 4.20) and multimodal (Figure 4.21) particle distributions.

Now that we have an estimate $\hat{f}$ of the continuous probability distribution, we use the HPD region to approximate a Wald-like test for multimodal distributions. The $1-\alpha$ HPD region for x, $R(x)$ is defined as a set such that there exists a k such that $R = \{x : f(x) > k\}$, and

$$\int_{x \in R} f(x) = 1 - \alpha.$$

We can approximate the integral by evaluating $\hat{f}$ on a regular grid, and summing the maximum bins until we exceed $1-\alpha$ (being careful to normalize the total sum to 1). By taking the union of the areas of the summed bins, this procedure yields the desired HPD region. With a unimodal particle distribution, this will usually yield the same result as the Wald/confidence interval test, but is much more satisfactory in the multimodal case (Figure 4.22).

Normalizes Estimation Error Squared

Bailey et al. [2006] analyze the inconsistency of the FastSLAM algorithm of Montemerlo [2003] (and RBPF SLAM approaches in general) using a measurement of particle filter consistency based on the χ^2 goodness of fit test. The advantage of the χ^2 test is that it gives both upper

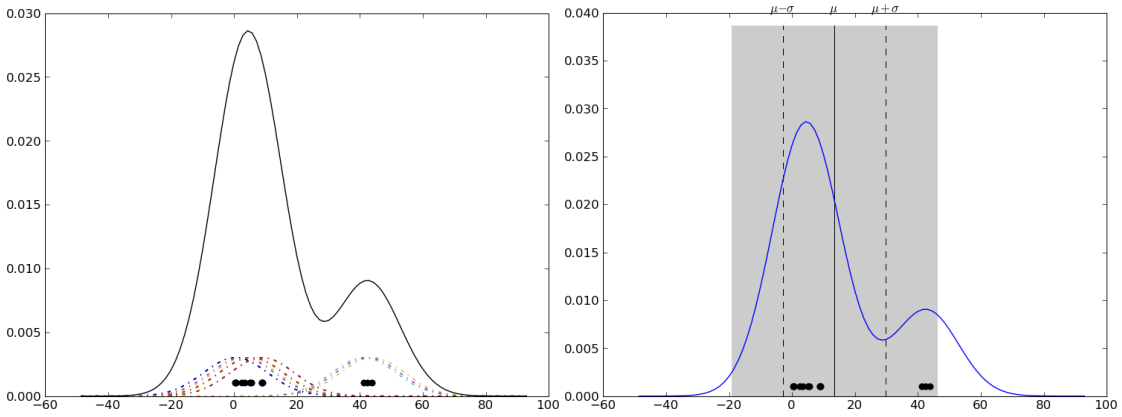

Figure 4.21: Multimodal samples, each with a Gaussian kernel (dashed lines), which when summed together form the KDE of the continuous pdf. The right plot shows the inappropriateness of the unimodal confidence interval.

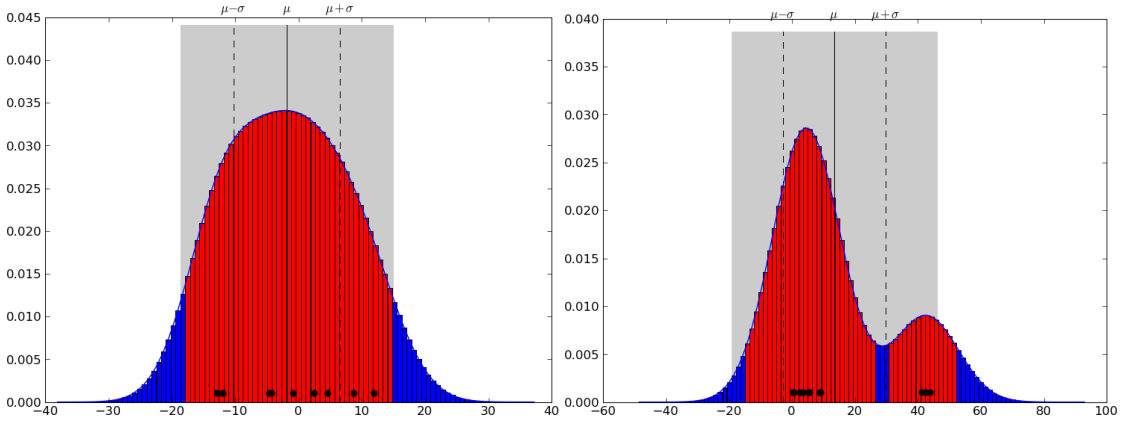

Figure 4.22: Left: unimodal samples, right: multimodal samples, showing the highest posterior density region (red bars) as compared with the unimodal confidence interval (grey box). For unimodal KDE's (left), the two regions coincide, but for multimodal KDE's (left), the HPD region is more satisfactory.

and lower bounds on the filter confidence, testing for overconfidence (high) or underconfidence (low).

The normalized estimation error squared (NEES) is defined as

$$\epsilon_t = (x_t - \hat{x}_t)^\top \hat{P}_t^{-1}(x_t - \hat{x}_t),$$

which is equivalent to the square of the Wald test value W given above. Under the assumptions of the χ^2 test, namely that the particle filter estimate $\hat{x}$ is normally distributed, ϵ_t is a χ^2_k random variable with degrees of freedom equal to the dimension of the state $k = dim(x)$. By performing N runs of the filter, or trials, we can take advantage of the fact that summing χ^2 random variables sums their degrees of freedom, so that if

$$N\bar{\epsilon}_t = N\frac{1}{N}\sum_{i=1}^{N}\epsilon_t^{(i)}$$

then $N\bar{\epsilon} \sim \chi^2_{dim(x)\times N}$. Since the expected value of a χ^2_k random variable with k degrees of freedom is k, the χ^2 test evaluates the validity of the hypothesis that the filter is consistent by constructing a confidence interval for $N\bar{\epsilon}$.

For our purposes, we consider only x, y, and yaw, discarding the other three degrees of freedom. With $dim(x) = k = 3$ degrees of freedom and $N = 50$ trials, the 95% confidence interval for $N\bar{\epsilon} \sim \chi^2_{150}$ is (116.7, 184.3), or dividing by N, we accept the hypothesis that the filter is consistent if $\bar{\epsilon} \in (2.33, 3.69)$. If $\bar{\epsilon}$ is below this interval then the filter is conservative, or underconfident. If it is above the interval then the filter is optimistic, or overconfident.

The conclusion of Bailey et al. [2006] is that FastSLAM rapidly becomes overconfident, due to exponential loss in particle diversity. We believe this is true of all RBPF SLAM approaches, of which FastSLAM is a well-known example, but not true of SegSLAM, since it can sample an exponential number of trajectories from the segmented map. We investigate this question in the following two experiments.

All of these consistency metrics avoid dealing with the particle maps by examining the particle trajectories: we believe that future work in consistency checks should directly incorporate the maps.

4.10.1 Consistency Experiment 1 – Ground truth in Bruceton Mine

Cave Crawler was deployed in the Bruceton Research Mine in December 2006, and traversed an L-shaped corridor for a total distance of about 190 m. Every few meters the vehicle stopped moving and collected a scan of the tunnel. At the same time, a Total Station was used to survey four points on the vehicle, giving highly accurate ground truth in all 6 DOF for each scan origin. In order to test the consistency properties of RBPF SLAM and SegSLAM, we excised all the intervals when the vehicle was stationary, yielding a continuous movement dataset.

We ran 50 trials of RBPF SLAM (with 50 particles) and SegSLAM (with 50 particles) on this continuous movement dataset. For each trial, we evaluated whether the ground truth end position (x,y,yaw) was within the 1 and 2 σ and 68% and 95% HPD credible regions. For this dataset, which has no loops, in most trials the final particle distribution was fairly unimodal, so the two tests often agree (Figure 4.23. To evaluate the SegSLAM particle distribution, we generated 50 trajectory samples in the global frame: there were 5 segments. The SegSLAM particle distributions (Figure 4.23) were usually more unimodal than the RBPF particle distributions (Figure 4.24), which were often clearly clustered due to resampling. SegSLAM avoids this particle depletion because the segmented map maintains particle diversity. The results over the 50 trials presented in Table 4.4 show that RBPF SLAM is overconfident and/or wrong (the main particle clusters are far from the true vehicle position) while SegSLAM is conservative in its confidence and (usually) has significant clusters of particles near the true position.

For the χ^2 test, the interval is (2.33, 3.69), and $\bar{\epsilon}$ was 30.2 for RBPF SLAM and 2.10 for SegSLAM. This supports the conclusions of the other tests, namely that RBPF SLAM is overconfident, and that SegSLAM is actually underconfident, if anything. It is important that while SegSLAM is conservative, the difference between the unimodal 1 σ test and the 68% HPD test show that this is not due to simply scattering particles around randomly (also see Figure 4.23).

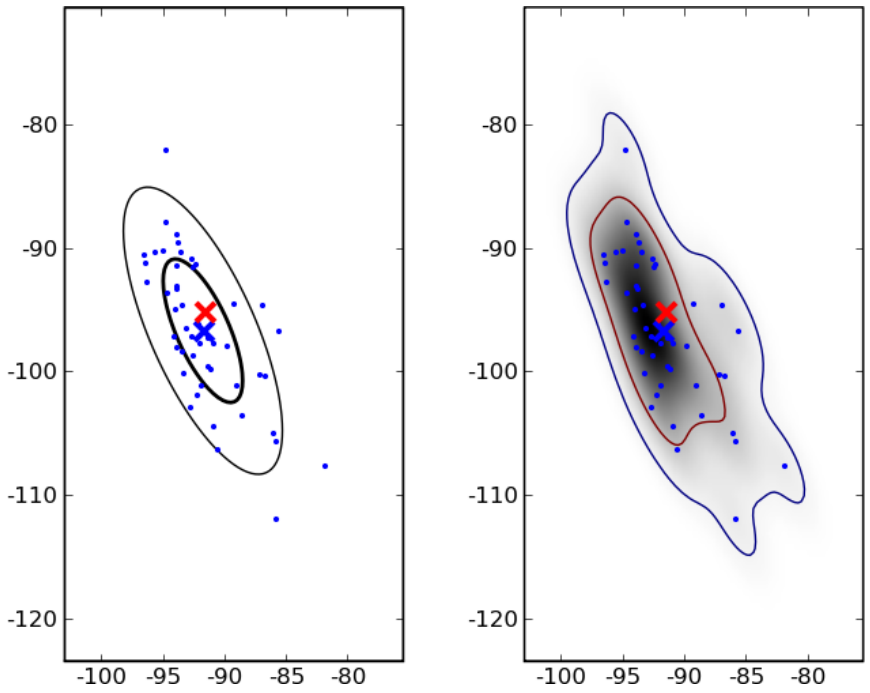

Figure 4.23: The final particle distribution of SegSLAM (with 50 particles) run on the Bruceton L dataset. 50 trajectory samples were generated from the segmented map: there were 5 segments. Blue dots indicate the particle positions, blue X the particle mean, and red X the ground truth position. Left, the 1 and 2 σ bounds. Right, the 68% and 95% MPD regions.

Test	RBPF SLAM (50p)	SegSLAM (50p)
1 σ interval	3 / 50	11 / 50
2 σ interval	4 / 50	27 / 50
68% HPD region	10 / 50	40 / 50
95% HPD region	21 / 50	50 / 50
$\bar{\epsilon}$	30.2	2.10

Table 4.4: Number of times surveyed ground truth was within the test regions over 50 trials.

4.10.2 Consistency Experiment 2 – Loop closure in Bruceton Mine

Cave Crawler was deployed in the Bruceton Research Mine in February 2007, and drove a loop through a series of corridors for a total distance of about 250 m (Figure 4.1). While there is no ground truth for this dataset, we manually registered the final position of the vehicle after it returned to near its initial starting location.

As in the previous experiment, we ran 50 trials of RBPF SLAM and SegSLAM on this dataset and evaluated the trials. We used 20 and 200 particles for RBPF SLAM, and 20 particles for SegSLAM. Due to the loop closure, the particle distributions are often far from unimodal, in which case the HPD region test gives a more satisfactory answer (Figure 4.25).

The results over the 50 trials presented in Table 4.5 again clearly show that RBPF SLAM with 20 particles is overconfident and/or wrong while SegSLAM is both correct and conserva-

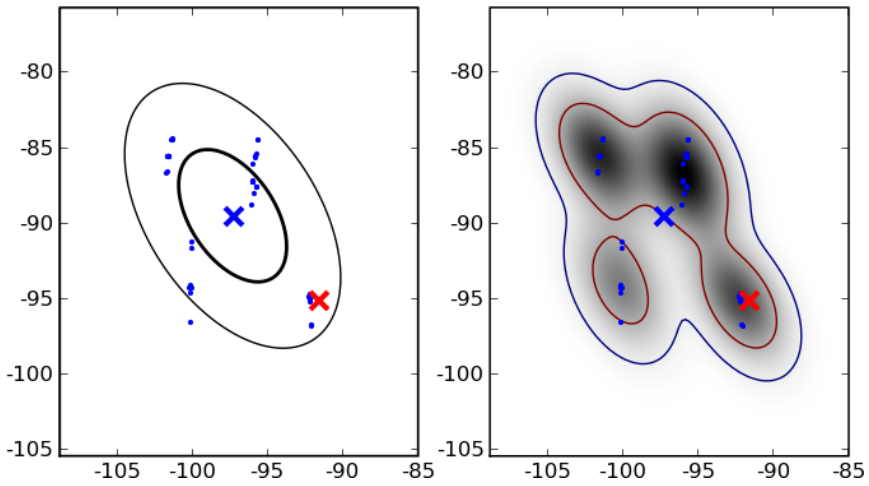

Figure 4.24: The final particle distribution of RBPF SLAM (with 50 particles) run on the Bruceton L dataset. See description of Figure 4.23.

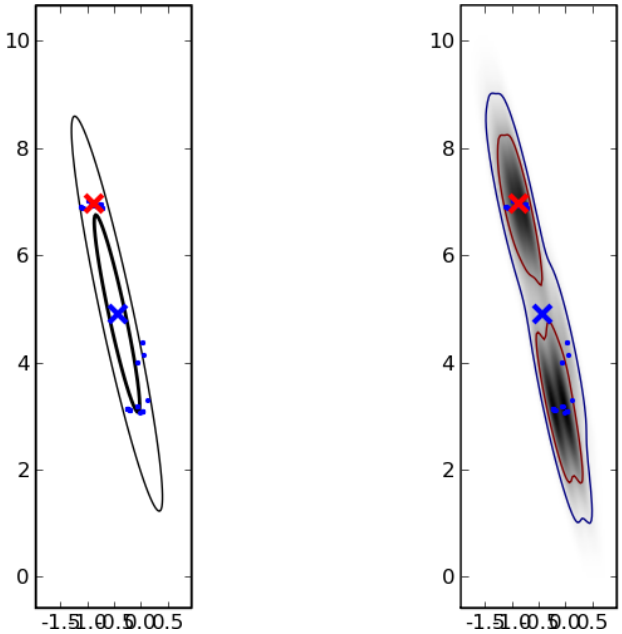

Figure 4.25: The final particle distribution of SegSLAM (with 20 particles) after closing the loop on the Bruceton loop dataset. 50 trajectory samples were generated from the segmented map: there were 9 segments. See description of Figure 4.23.

tive. Even though SegSLAM ran with only 20 particles, we generated 50 trajectory samples in the global frame: there were 9 segments. As a sanity check on our methods, RBPF SLAM with 200 particles is consistent for this loop, with $\bar{\epsilon}$ squarely in the (2.33, 3.69) χ^2 test interval, demonstrating that adding sufficient particles to RPBF SLAM can make it consistent for a given loop length.

Test	RBPF SLAM (20p)	RBPF SLAM (200p)	SegSLAM (20p)
1 σ interval	1 / 50	6 / 50	32 / 50
2 σ interval	2 / 50	22 / 50	43 / 50
68% HPD region	2 / 50	27 / 50	49 / 50
95% HPD region	2 / 50	43 / 50	50 / 50
$\bar{\epsilon}$	37800	3.16	1.18

Table 4.5: Number of times loop-closure ground truth was within the test region over 50 trials.

4.10.3 SegSLAM Experimental Summary

For segmentation, we described the predictive score heuristic, a real-time method for segmenting when the vehicle passes from one region to another, based on a particle map's ability to predict new measurements. The predictive score heuristic depends on two parameters: the length of the predictive window, which we set to about 10 seconds, and the segmentation threshold. We also described a motion error model-based heuristic for use in unstructured environments, where the predictive score was not informative.

The segmented map is a powerful map representation that encodes connections between compatible metric segments. This means that the segmented map can flexibly represent multiple hypotheses about both metric maps and their topological relations: a particular segment may be incorporated into multiple possible topologies. Generating map samples – large-scale metric map hypotheses that incorporate many segments – is implemented as a stochastic breadth-first search of the segmented map that exploits loop-closures detected by the matching process. Map sample generation may be truncated at a fixed graph depth to yield a local (rather than a global) metric map.

Map samples are hypotheses about the local map that include a set of segments and the relationships between them. These relations are necessarily in a local coordinate frame, since the map sample may be truncated based on breadth-first search depth. The search for matches is focussed by the map sample by only attempting matches to segments within a certain metric distance threshold. There is no reliance on any global frame, but there is the possibility that many map samples will need to be generated before the true match falls within the metric distance threshold.

Out of all the map matching methods that we discussed in Chapter 2, the method that we preferred for segment matching was to generate pointclouds from the evidence grid maps by creating a point for each voxel above a threshold, and then using ICP to match the pointclouds. Similar to the predictive score segmentation metric, we used a brief (10 second) window of "future" data to build one of the maps, and matched this map to nearby candidate segments in a map sample. This DRCO binning method is faster than the isosurface extraction methods, and yet still robust to noise and point density issues.

In a sense, each particle can be considered as consuming a certain fraction of the robot's finite computational resources. Ultimately, the SegSLAM algorithm distributes the robot's finite computational resources (as particles) over multiple metric and topological hypotheses, resampling in order to focus on likely regions of the underlying probability distribution. SegSLAM uses the metric segment maps themselves as matchable features, and constrains the search for

matches using map samples drawn from the segmented map that maintains diversity over vehicle trajectories. This yields a scalable real-time SLAM approach that emphasizes local consistency and speed over global consistency.

4.11 Summary

In this chapter, we have shown how segmentation can be used to factorize the SLAM problem in such a way that our RBPF SLAM approach can be applied within each segment without running into the particle depletion problem. We call this new algorithm SegSLAM, and have addressed the new issues that segmentation raises, namely when to segment and how to detect matches.

For segmentation, we described the predictive score heuristic, a real-time method for segmenting when the vehicle passes from one region to another, based on a particle map's ability to predict new measurements. We also described a motion error model-based heuristic for use in unstructured environments, where the predictive score was not informative.

To detect matches, or loops back into old segments, we leveraged the map matching techniques of Chapter 2, and also described a method for using particle filter localization to see if the recent robot perceptions matched well with a target segment. To find candidate segments for matching, we showed how to generate metric map samples from the segmented map, and then discussed simple heuristics for winnowing down nearby candidates.

We demonstrated SegSLAM with the Cave Crawler robot in several environments, including a mine, a parking garage, and a multi-level partially constructed building. We showed that it is faster, more accurate, and handles larger scales than our previous RBPF SLAM. In particular, SegSLAM's topological flexibility allows it to excel precisely in the sparse, loopy 3D environments where RBPF SLAM fails.

One difficulty in working with SegSLAM is that it does not lend itself well to global error metrics: it emphasizes local consistency and speed over global optimality. This was a deliberate trade-off that is suitable for some applications and not for others.

An important caveat is that even though SegSLAM can close loops with a single particle, the resulting segmented map may not properly represent uncertainty: the filter may be overconfident. Our two experiments in consistency demonstrate that SegSLAM is conservative in its uncertainty, and also correctly maintains a cluster of particles near the true vehicle location (its easy to be underconfident and wrong by randomly scattering particles). This is in contrast to FastSLAM and other RPBF SLAM approaches in general, and our RBPF SLAM method in particular, which in agreement with the results of Bailey et al. [2006] become overconfident due to particle depletion. In other words, RPBF SLAM methods lose particle trajectory diversity at an exponential rate due to resampling, but SegSLAM maintains this diversity in the segmented map – although it only uses a subset of the trajectories due to real-time constraints.

Map samples are hypotheses about the metric relations between submaps. These relations are necessarily in a local coordinate frame, since the map sample may be truncated based on segment distance (expressed as breadth-first search depth). The search for matches is focussed by the map sample by only attempting matches to segments within a certain metric distance threshold. There is no reliance on any global frame, but there is the possibility that many map samples will need to be generated before the true match falls within the metric distance threshold.

This can be addressed in two ways, one we currently use and the other that will be future work. The one we use is to expand the search distance threshold based on the uncertainty in the chained segment transformations. In the worst case (in particular, if there was no dead reckoning and the segments were composed of dense scans from a single location), this could degenerate to having to search for matches with all segments. But cases where the dead reckoning is reasonable and linear chain of segments is not too long, the search space can be reduced to just one or two candidates for matching. The method that should be addressed in future work is to use other localization modalities (visual correlation, RFID, GPS, etc) to likewise constrain the segment search space. Again, in good conditions they could recommend a single candidate for matching.

In the next section, we will investigate how to combine planning and the segmented map to find actions that are expected to reduce SLAM uncertainty.

Chapter 5

Planning with Segmented Maps

5.1 Introduction

In previous chapters, the SLAM system has been a passive consumer of data. As the robot moves around and collects more data, SLAM attempts to build maps, localize, close loops, etc., but the robot's actions are generated by some other process. There are clear cases, such as when the robot has nearly closed a loop, when taking certain actions would significantly reduce the amount of uncertainty in the SLAM belief state. Even more, a policy of taking such SLAM-aiding actions can limit the overall complexity of the SLAM problem for a given environment. This process of allowing SLAM to actively control the robot's behavior is known as Active SLAM. In this chapter, we investigate a novel set of methods that allow SLAM to generate actions by probing the implicit uncertainty in its belief state.

Generally, planning is the process of creating a sequence of feasible actions, or a plan, which is expected to achieve a particular goal. With the goal of reducing the SLAM uncertainty, which includes both exploration and localization, the planning algorithm generates and evaluates candidate plans that balance the need to expeditiously explore new territory and to maintain an accurate localization solution, yielding a complete and accurate map in a reasonable amount of time.

A standard method for balancing the need to explore new regions and the need to know the location relative to previously mapped regions is to explicitly evaluate the expected information gain of different plans. Throughout the previous chapters, we have shown how the uncertainty or entropy of the SLAM belief state can be estimated. The planner can attempt to predict the expected entropy that will result from executing different plans, and select the plan that yields the lowest expected entropy or highest information gain. A common approach for making these predictions is to perform multiple simulations of the execution of the plan based on the current SLAM state, and then to evaluate the entropy of the resulting simulated SLAM state. However, it is computationally expensive to carry out the many accurate simulations needed, so they are often simplified and approximated, and for systems with large state spaces, such as SegSLAM, the approximations necessary for estimating the resulting entropy begin to approach outright hacks – and even the hacks become computationally intractable.

Rather than attempting to explicitly reason about expected information gain using crude sim-

ulations and loosely justified entropy metrics, we show how plans can be used to probe the SLAM belief state, and how the results of these probes can be then used to select plans that reduce the SLAM uncertainty. Exploiting the flexible world model encoded in the segmented map (described in the previous chapter), we apply basic Rapidly-exploring Random Trees (RRTs) planning to map samples from the segmented map, and then use simple statistical criteria on the success rate of the plan in other map samples to select plans that implicitly reduce the uncertainty in the underlying SLAM belief state.

Our approach extends to true loop prediction. A major goal of Active SLAM is to detect and close loops: when the robot returns to a place that it has previously mapped and recognizes that it is back in that place, it can correct for all of the error that has accumulated since it left. However, most Active SLAM methods have to perceive the loop before they can decide to close the loop. In other words, they need to have some sensor data that includes the previously mapped area before they can even consider the possibility of more closely investigating that area in order to precisely close the loop. But by using the segmented map as a predictive model, our method applies to the case of exploration or loop-closure through unknown regions. We extend map sample generation to include guesses, or hallucinations, about unknown regions using nearby map segments as templates. More importantly, we show how to get real use out of these hallucinations by using them to predict potential loops and generate plans for closing these loops.

We begin with a discussion of related work, then describe the motivation behind the information-gain, or entropy, formulation for Active SLAM, as well as some of the difficulties of explicitly reasoning about information gain. Next we use the example of active localization on the ocean floor to show how it is possible to select between plans without explicitly evaluating entropy. We then lay the ground-work for planning in the segmented map by discussing our method for using Rapidly-exploring Random Trees (RRTs) in 3D evidence octree maps. We then describe stochastic planning using the segmented map to find plans than implicitly reduce entropy, including the use of the segmented map as a predictive model for generating plans that extend into unknown regions. Finally, we demonstrate Active SLAM with predictive loop closure in a real world experiment.

5.2 Related Work in Planning

Exploration If location is known, the problem is map building. This is often formulated as a coverage problem, which is surveyed by Choset [2001]. Acar et al. [2003] applies probabilistic methods to the coverage problem, demonstrating the intersection of coverage with the area of exploration and action selection in uncertain environments [Thrun, 1993] [Koenig and Simmons, 1993] [Simmons and Koenig, 1995]. Singh [1995] used information entropy to optimize the speed of a mapping robot to maximize the mapping information gain. Generally, taking uncertain actions in an uncertain environment can be modelled as a partially observable Markov decision process (POMDP) [Cassandra et al., 1996, Kaelbling et al., 1996]. POMDPs are theoretically satisfactory and handle uncertainty well but are computationally intractable for any realistically complicated scenario, so heuristics (like greedy action selection) are used. More principled approximate methods include cases when there is a clear preference for the uncertain quantities [Likhachev and Stentz, 2006], which uses multiple searches in the underlying (uncer-

tain) environment, similar to our use of multiple map samples, or more generally reducing the dimensionality of the problem through belief compression Roy et al. [2005]. Our method of selecting macro-scale actions based on simplified sensor models may loosely be seen as ad-hoc dimensionality reduction.

Active Localization When a prior map is available, the problem is localization. Fox et al. [1998] describe a method for both controlling the robot motion and pointing the sensors to improve localization. The "Coastal Navigation" planner described in [Roy and Thrun, 1999] is so called because the planner uses a one-step look-ahead for belief state entropy, and thus tends to stick close to informative regions, like a ship staying within sight of the coast. Newman et al. [2003a] describe feature-based exploration within the ATLAS framework of Bosse et al. [2004], in which the features which make up the map are used to generate actions which will further expand and refine the map.

Many methods for active localization use motion-constraining control laws that simplify the correspondence problem by moving so that the robot will observe the landmark from the same position: Choset and Nagatani [2001] derive exploration control laws based on generalized Voronoi Graphs, Kuipers and Byun [1991] use their spatial semantic hierarchy and along with G. Dudek [1996] control exploration based on the tree of topological hypotheses, and Mataric [1990] use boundary-following.

Gonzalez and Stentz [2007] present a deterministic planner that takes into account partial and/or uncertain maps, as well as the vehicle's localization needs, to plan paths that minimize a combination of the traversability cost and localization uncertainty.

Active SLAM Balancing exploration and localization, Sim and Roy [2005] describe exploration trajectories that optimize the mean uncertainty in order to build more accurate maps. Kollar and Roy [2008] apply reinforcement learning to the problem of optimizing trajectories for map accuracy.

Burgard et al. [1997] distinguishes between active navigation, in which the entire robot moves, and active sensing, in which only the sensors are pointed or focussed on particular targets.

Using information-theoretic frameworks, Feder et al. [1999], Bourgault et al. [2002], and Burgard et al. [2005] describe complete systems for active SLAM, including active loop closure Stachniss [2006]. Kümmerle et al. [2007] do active sensing with a particle filter, clustering the particles into groups and calculating the total expected entropy for the particle filter by a weighted average of the expected entropy for each group. Thompson [2008] investigates information-gain based exploration of science-oriented models for science autonomy. Grocholsky et al. [2003] describe information-theoretic control of multiple platforms (an active area of research), even with nonlinear measurements.

Predicting unknown regions Nabbe et al. [2004] use "hallucination" to fill in gaps in a sparse map to assist in estimating path costs for navigation over large-scale terrains. Relying heavily on repetitive structure in the environment, Chang et al. [2007] match snippets from the current map to partially observed areas, with the idea of not exploring regions that can be predicted or of using the prediction to do localization, a questionable strategy. Although they don't use it

for prediction, Fox et al. [2003] have the most robust approach, learning a Dirichlet prior over structural models from a library of previously explored environments, and using samples from this distribution for estimating the probability that the robot is inside or outside the known map. Our segmented map is built as the robot explores the environment (which is both an advantage and a disadvantage), and we use it for predictions about unexplored areas.

5.3 Information Gain-Based Action Selection

Action selection gives the localization system the ability to influence the behavior of the vehicle. In particular, when the position estimate becomes uncertain, action selection can recommend actions that are expected to reduce this uncertainty. Since large regions of many environments are basically flat and featureless, simply travelling in a straight line may not be sufficient for quick and accurate position convergence and active localization is extremely useful, if not essential. The process of choosing actions in order to aide localization is known as active localization Burgard et al. [1997].

Many active localization methods use an information-theoretic framework, in which the entropy of the entire particle filter is estimated using a variety of heuristics, several of which have been described in Section 3.5. By simulating the effect of executing several different candidate actions on the particle filter, the action that is expected to lead to the lowest entropy can be selected. But this is difficult: not only is the simulation computationally expensive, but it must be repeated for several different simulated datasets: ideally one per candidate action per particle. To avoid this, Stachniss et al. [2004] only simulate the actions for a weighted subset of particles, while Kümmerle et al. [2007] attempt to cluster the particles into groups and run one simulation per cluster.

Following Stachniss [2006], we can evaluate a candidate action by simulating the measurements it expects to result from the action, and then examining the entropy of the SLAM belief state that results. In Chapter 3.5, we described how Roy and Thrun [1999] and Stachniss [2006] approximate the entropy of the belief state by decomposing it into the entropy of the particle cloud and the *expected* entropy of the particle maps, and how Blanco et al. [2008a] uses the average map entropy, which is approximated by computing a new evidence grid as the weighted average of all the particle evidence grids (in a single reference frame), and then computing the entropy of the resulting blurry map using the evidence grid entropy techniques discussed in Chapter 2. Finally, in Chapter 4 we described how we can generate many samples from the segmented map, merge them together, and then apply the approach of Blanco et al. [2008a] to estimate the SegSLAM entropy.

Considered as a whole, expected information-gain evaluation for action selection is based on multiple layered approximations and heuristics, and even so it is not computationally feasible. Our approach attempts to avoid all the expensive and approximate entropy estimates, and instead use stochastic planning to generate a large number of plans which implicitly reflect the SLAM uncertainty, and from which we can select a best plan using simple criteria.

In the next section, we introduce our approach to active localization that selects actions that are expected to generate data that maximally discriminates between the current position hypotheses.

5.4 Active Localization Using Most Discriminative Actions

When we are just localizing, rather than updating maps in full SLAM, we just want to find an action that allows the filter to discriminate between particles, yielding a unimodal particle cloud around the true vehicle position. We would like to find this action without explicitly simulating the particle filter state and then evaluating its entropy. We can do this by examining the simulated datasets from a subset of particles: the most discriminative action is that which generates the simulated datasets that are the most different. Since the real dataset which results from taking that action will only be similar to a few of the particles' simulated datasets, we know that all of the other particles will be discarded. Simulating datasets is fast, only requiring ray-casting and a simple kinematic vehicle motion model, and these datasets can be quickly compared using, for example, sum-squared difference.

Thus, if A is the set of candidate actions, M is the set of all particles, and $m \subseteq M$ is a subset of particles generated by sampling according to the particle weights, then for each action $a \in A$ and particle $q \in m$, we simulate the range measurements z_a^q using the particle pose, the simulated vehicle trajectory as a result of taking the action, and the map. We then find the most discriminative action:

$$\arg\max_{a \in A} \sum_{i \in m} \sum_{j \in m} (z_a^i - z_a^j)^2$$

By periodically repeating this action selection process, the vehicle will select actions which are expected to allow it to best discriminate amongst its current set of particles, thus reducing the particle filter entropy – without ever explicitly estimating the entropy. Due to its simplicity, this approach can be run faster, for longer action horizons, than full filter simulation and entropy estimation methods.

The set of candidate actions can be arbitrarily complicated, though the computational expense of evaluating them increases accordingly. In domains where the vehicle motion is constrained (by walls, for example), trajectory planners can applied to rapidly eliminate infeasible actions. In our experiments in the open underwater domain described here, we simply used straight-line motion for 30 m in one of the 8 cardinal and intercardinal directions as our set of candidate actions.

5.4.1 Justification of the Most Discriminative Action Criteria

We can verify the most discriminative action criteria by comparing it with the formal expected entropy criteria in a simplified 2D scenario. In this scenario, illustrated in Figure 5.1, the vehicle starts with some initial uncertainty in its position estimate, and must choose between two possible actions which are to move parallel or perpendicular to its primary axis of uncertainty. The only measurement is the perpendicular distance to the obstacle. Since everything in this simple scenario is linear, a Kalman filter can be employed with no approximations, and we can measure the entropy of the Kalman filter by using the differential entropy of the covariance matrix Σ:

$$H(KF) = \frac{1}{2} \log[(2\pi e)^N + \det(\Sigma)] \tag{5.4.1}$$

where N is the dimensionality of the state space: in this case $N = 2$.

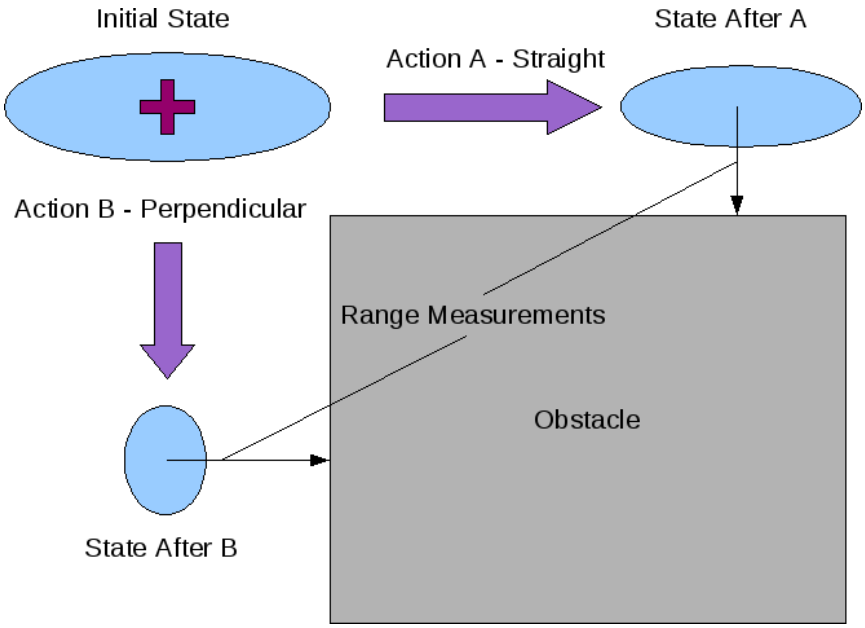

Figure 5.1: A simulated scenario for comparing the most discriminative action criteria with the expected entropy criteria.

We initialized the covariance to 1.0 in y and varied the covariance in x from 0.1 to 5.0. As shown in Figure 5.2, this caused the prior (initial) entropy to increase logarithmically. The same figure shows that when the covariance in x is below 1.0, the expected entropy *after* taking the straight action is less than taking the perpendicular action, but that the situation is reversed when the covariance in x increases beyond 1.0. Figure 5.3 shows the result of taking the standard deviation of 1000 simulated range measurements for each initial covariance, demonstrating that the most discriminative action criteria switches between the two actions at exactly the same point as the expected entropy criteria.

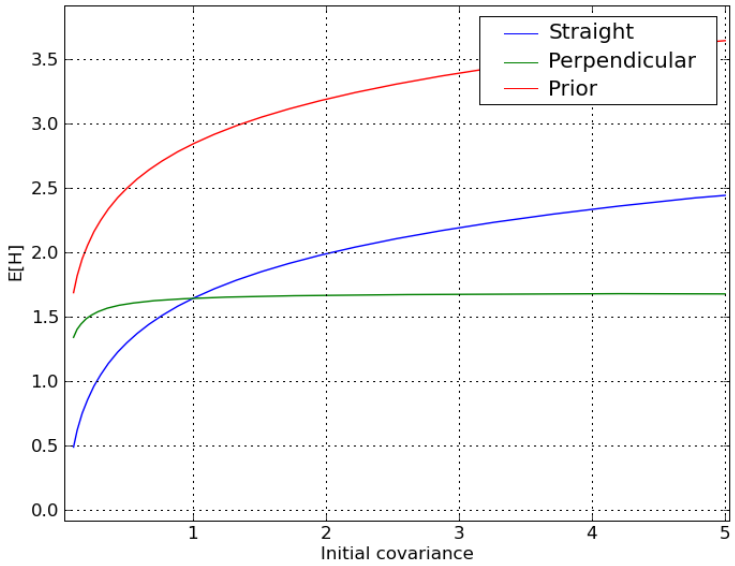

Figure 5.2: A comparison of the expected entropy (E[H]) for the straight and perpendicular actions, and the prior entropy before any action is taken, as the initial X covariance is increased from 0.1 to 5.0.

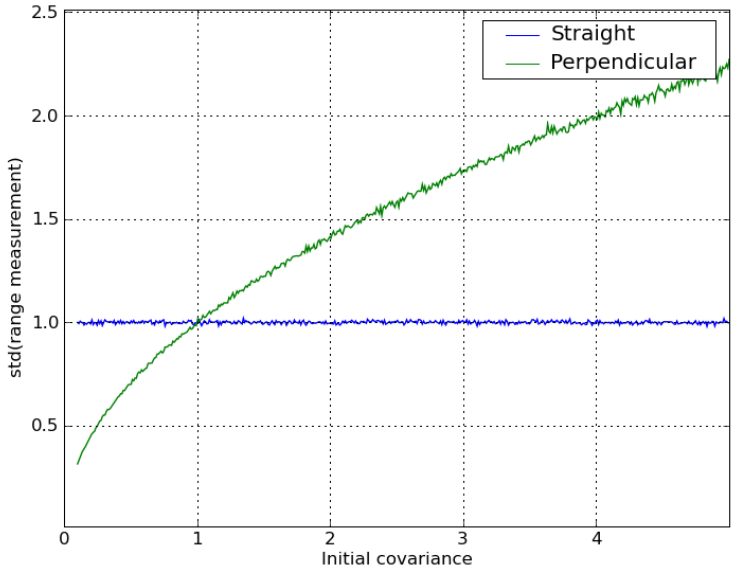

Figure 5.3: The standard deviation of 1000 simulated range values shows that the most discriminative action criteria (which picks the action with the highest standard deviation) will switch its choice between the two actions at exactly the same point as the expected entropy criteria, as the initial X covariance is increased from 0.1 to 5.0.

Figure 5.4: The MBAUV aboard the R/V Zehpyr. *Image: Duane Thompson (c) 2005 MBARI*

5.5 Active Localization Experiment – Axial Seamount

The Monterey Bay Aquarium Research Institute (MBARI) has developed the 6000 m rated Multibeam Mapping AUV (MBAUV) (Figure 5.4), based on the modular 21" diameter Dorado AUV design Kirkwood et al. [2004]. The MBAUV navigation system is built around a Kearfott SEADeViL integrated IMU/DVL and provides dead reckoning with error on the order of 0.05% of distance traveled, so long as the vehicle is within DVL bottom-tracking range of the bottom Henthorn et al. [2006]. The primary mapping system is a Reson 7125 200 kHz multibeam sonar. The multibeam sonar array provides an array of 256 1° by 1° beams spread in a downward facing 150° fan perpendicular to the AUV's direction of travel. The update rate depends on the range to the bottom, but is generally about 2 Hz. The mapping AUV has demonstrated very successful survey operations Henthorn et al. [2006], including a survey of Axial Seamount in 2006. The MBARI mapping team has generously provided this dataset, which includes both the navigation data from the onboard IMU/DVL and sonar data from the multibeam sonar system.

Test Data

Our test dataset is from Axial and Monterey Canyon (Figure 5.5), a 23.3 km survey in 5 legs of about 4 km, plus two cross-track legs, in about 4.4 hours (Figure 5.5). After examining the dataset, we discarded the last 27 out of 256 beams, which seemed to be very noisy. We divided the dataset into two pieces: the trackline legs and the cross-track legs. We then built a map with the trackline legs, and used the cross-track legs to test our localization methods.

Map Construction

We used the onboard dead reckoning, together with the sonar range measurements, to construct an octree evidence grid map at 1 m resolution. The octmap dimensions were 8192^3, and although

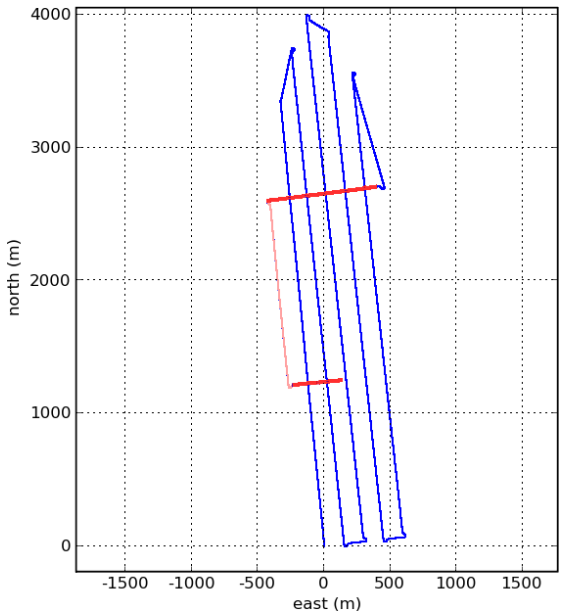

Figure 5.5: Axial survey tracklines – the long tracklines were used to create the map and random segments of the cross-track lines were used to verify localization.

only a small fraction of this volume ended up being used, the large size meant that we didn't have to worry about the vehicle driving off the edge of the map – this scalability is essential for a real-world system.

It took 22 minutes to construct the map on a 3 GHz P4 computer using the 20 km of trackline legs, a total of 6.5 million range measurements. Since this portion of the mission took 225 minutes to run, this is about 10 times realtime.

Un-compacted, the octree map size was 530 MB. After running lossy compaction, in which the octree map was compacted by thresholding into {empty, occupied} and homogeneous regions were consolidated, the map was just 78 MB. As an upper bound, a uniform array of 4096 x 4096 x 4096 would be 65536 MB. As a rough lower bound, a minimal 1 m resolution uniform array tightly fit to to the survey area would be about 4096 x 1024 x 20 = 100 MB; the compact octree efficiently represents the volume.

5.5.1 Baseline Localization Results

Before discussing the active localization results in this environment, we summarize the localization results. Although the dataset was collected at about 1500 m, we would like to demonstrate that our method can converge from the initial descent error that would result from a descent of 1000 and 6000 m. The vehicle travels at about 1.5 m/s and dives at about 30 degrees pitch, yielding a 0.75 m/s dive rate. Brokloff [1998] reports performance of 0.16% of distance traveled with bottom tracking and 0.35% with only water tracking. Since the MBAUV has much better bottom tracking performance, 0.05% of distance traveled Henthorn et al. [2006], we hope that assuming 0.35% during descent is a suitably conservative estimate. Thus, if the vehicle needs to dive 1000 m to bring the DVL into range of the bottom, the final position error on the bottom is 1000 m ÷

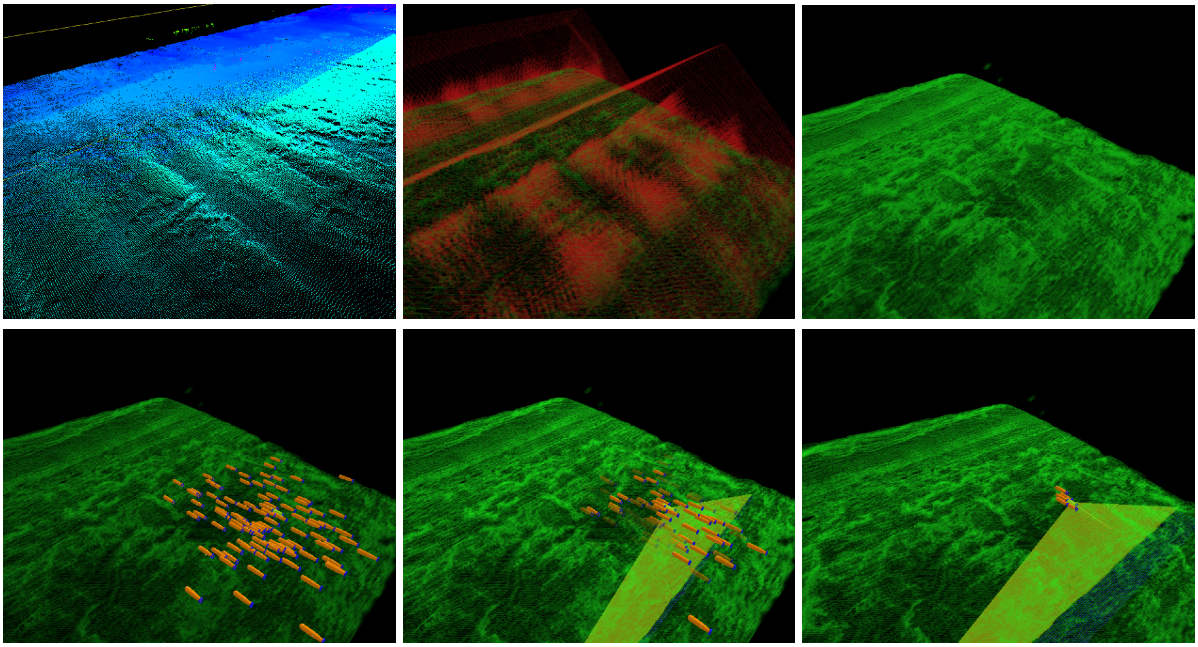

Figure 5.6: Localization example showing 1) sonar pointcloud from two tracklines, 2) a portion of the evidence grid constructed from all the tracklines, red is known empty space and green is known occupied space, 3) the occupied portion of the evidence grid, 4) the true vehicle position (in yellow) at the beginning of a cross-track line (not used to construct the evidence grid) and the particle cloud of possible positions, 5 & 6) the iterative convergence of the particle cloud to the true position over the course of a few seconds.

0.75 m/s $\times$ 0.35% = 4.7 m 1 σ. The same calculation for diving 6000 m yields a position error of 28 m 1 σ.

To evaluate basic particle filter localization, we constructed a map as described above, and attempted to localize using data from the as-yet unused cross-track legs. This process is summarized in Figure 5.6. We used several different segments of the cross-track legs for testing. For a given segment of one of these legs, we first perturbed the initial position according to a 5 or 28 m 1 σ Gaussian (depending on whether we simulating a 1000 or 6000 m dive), and then distributed the particles around this perturbed mean according to a 10 or 35 m 1 σ Gaussian distribution. The distribution of the particles was larger than the dive-length derived position perturbation to ensure that there were particles near the true position. Repeating this entire process, we evaluated the particle filter performance over multiple runs according to how quickly and accurately it converged to the "ground truth" position. Convergence, indicated by low particle cloud variance, is an important metric because it indicates when the filter is confident in a unique position estimate. In the broader context of AUV operation, convergence indicates that the AUV is well localized and can begin to perform its other tasks on the sea floor.

The particle filter did not always converge, as indicated by low particle position variance: Figure 5.7 shows a failure to converge because of multiple modes, one of which is the correct one, so the filter could converge given more data.

Convergence was consistent for the 1000 m test (Figure 5.8), but sometimes failed (detectably) for the 6000 m test (Figure 5.9). Since the variance remained high for these failed

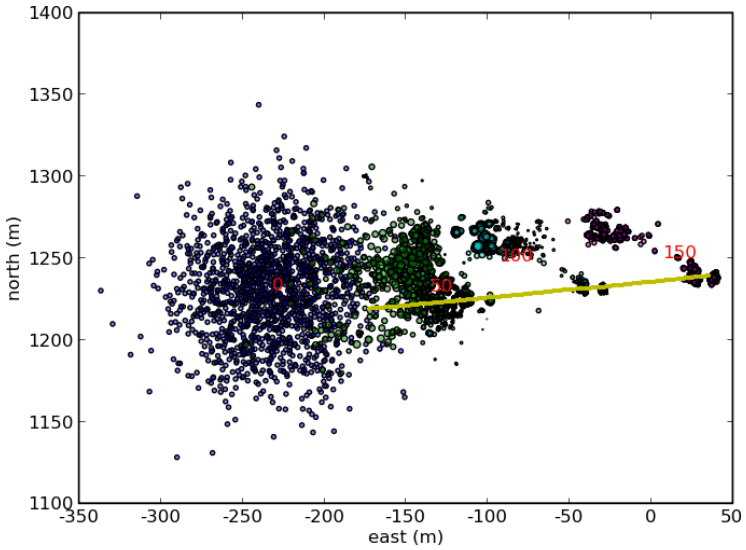

Figure 5.7: Failed particle filter convergence for a simulated 6000m dive: there are still multiple particle modes, one of which is the correct position, so the particle filter could still converge to the correct position given more data. Note that the mean of the initial particle distribution is significantly perturbed from the true initial position shown by the left end of the yellow line.

runs, the vehicle could choose to restart the entire localization process (with perhaps a different initial position bias), repeating the procedure until it successfully localized.

5.5.2 Active Localization Results

Using the map constructed above, we simulated vehicle motion and range data so that we could allow the simulated vehicle to take different actions. We compared three methods for choosing the heading for the next 30 m leg of vehicle travel: a constant heading, a random heading, and an actively selected heading in which the vehicle chose between 8 different headings (at 45 degree increments) by finding the most discriminative action (Figure 5.10). Note that, as shown by the localization results with real data, sometimes traveling on a constant heading can yield excellent convergence: in order to distinguish whether active localization has an effect, we selected a starting location and direction in which constant heading was known to fail due to the lack of terrain variation. But even when constant heading fails we also needed to show that our action selection process was better than simply choosing a random heading. Localization convergence (Figure 5.12) shows that active heading selection significantly reliably converges better than either keeping a constant heading or randomly selecting the next heading.

Active Localization Summary

We have demonstrated active localization with multibeam sonar data, and shown that by selecting the most discriminative action, active localization can yield accurate convergence in situations where the terrain variation is sparse and convergence does not necessary occur at all.

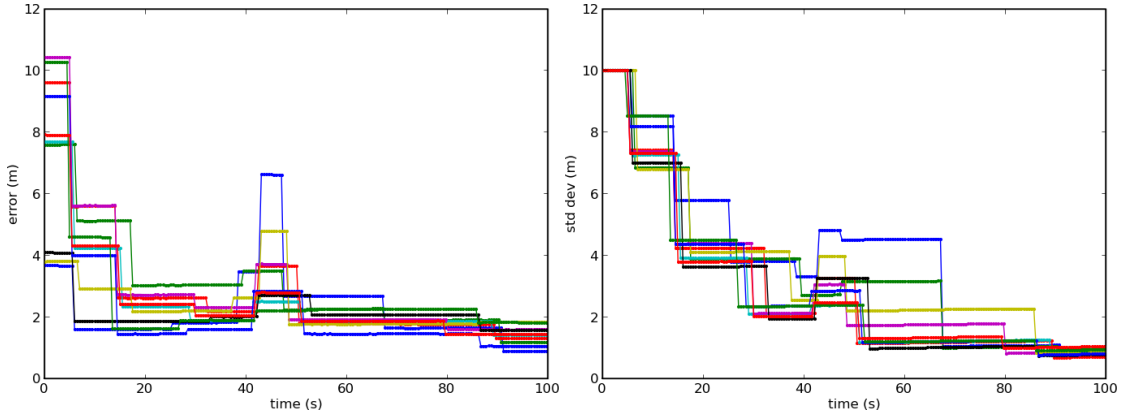

Figure 5.8: Convergence plot for 10 different runs of the 1000 m test: on the left is the position error over time, and on the right is the standard deviation of the particle positions over time. The fact that all the particles converge to a position error of around 1.5 may indicate that our ground truth is slightly incorrect.

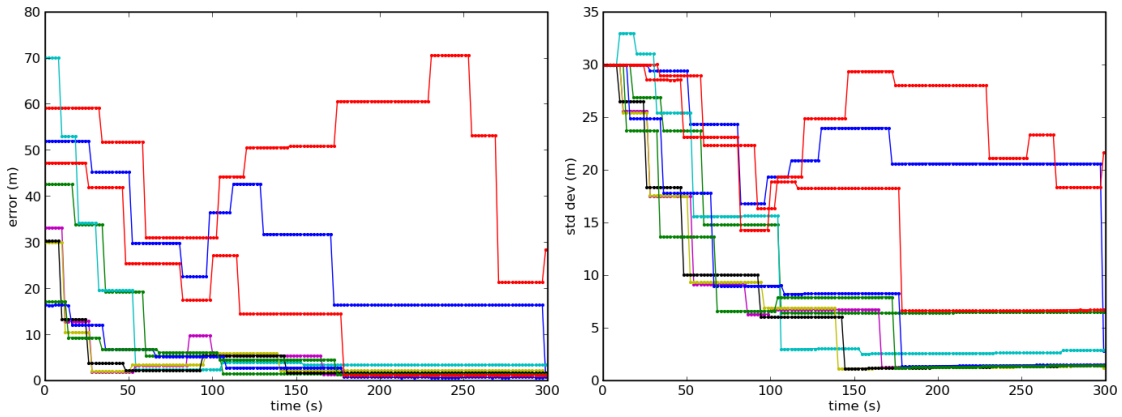

Figure 5.9: Convergence plot for 10 different runs of the 6000 m test: on the left is the position error over time, and on the right is the standard deviation of the particle positions over time. Note that several runs failed to converge, but this can be detected by observing that the standard deviation stays high (indicating a disperse or multimodal particle distribution).

At the same time, this experiment has glossed over two important problems: path planning and SLAM. In the underwater environment, the AUV didn't need path planning because it could swim straight toward its goal. This is not the case for the subterranean and urban environments that we consider next. Further, we assumed that we could build a map and then perform active localization – we will consider doing both at once in a later section on Active SLAM.

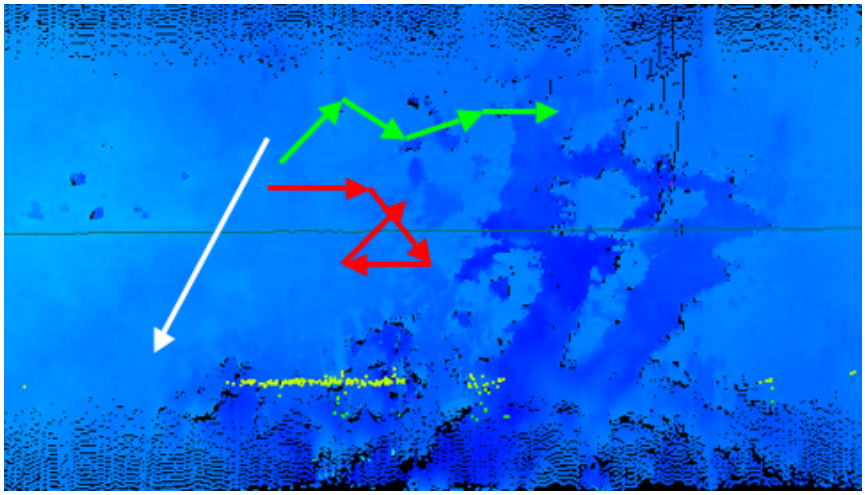

Figure 5.10: Examples of three different action policies over a fairly flat and featureless region of the sea floor: constant heading (white), random heading changes (red), and actively selected heading (green).

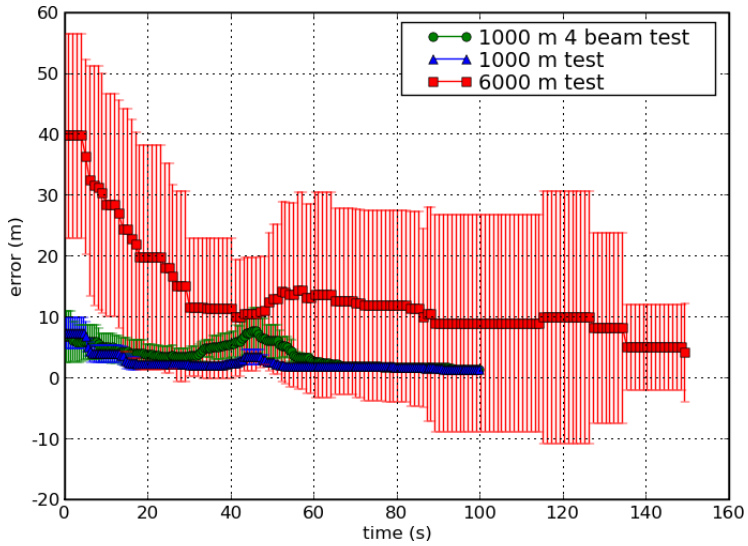

Figure 5.11: Comparison of the average performance, with error bars, of 10 runs of the three types of tests. Note that the final error of the 6000 m test is so poor because several runs (detectably) failed to converge (see Figures 5.8 5.9).

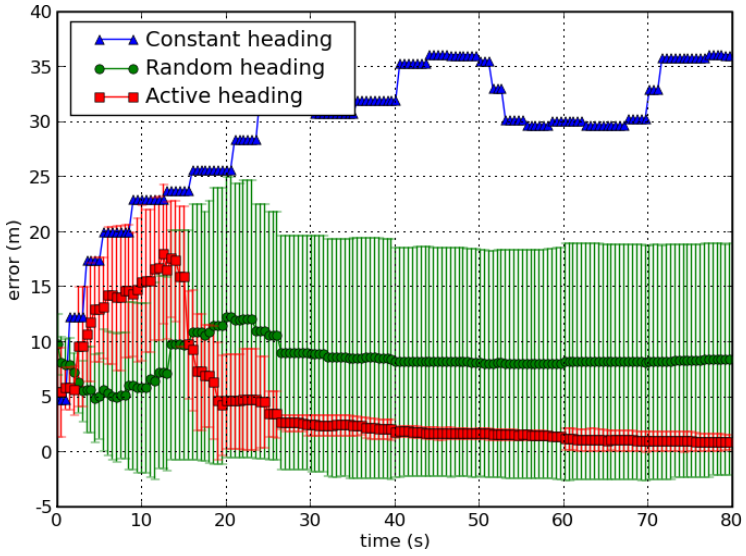

Figure 5.12: Comparison if the average performance, with error bars, of several action selection strategies. The error bars indicate the standard deviation of the position error across 5 runs (except for constant heading which always has nearly the same result).

5.6 RRT Path Planning in 3D Octree Evidence Maps

Although many different methodologies could be employed to path-plan within our 3D octree evidence grid maps, the fairly high 6 DOF dimensionality of the state space, the need for speed instead of completeness, and the complexity of the vehicle traversability model caused us to lean towards the use of Rapidly-exploring Random Trees (RRTs) LaValle and Kuffner [2000]. Further, we don't have specific goal states, but are interested in finding actions that further the implicit goals of mapping and localization.

In our RRT implementation, we build the tree by randomly selecting a node to expand (initializing the set of nodes with the current vehicle pose), and then pick a random action, parameterized by distance and curvature. We simulate the vehicle's kinodynamic motion in small steps, and check the resulting state by attempting to place the vehicle in the map. This placement check involves using ray tracing in the evidence grid map to find four wheel contact points, fitting a plane to these contact points to find vehicle attitude, and using the contacts and attitude to check body clearance. The placement check will fail if the wheel placement exceeds reasonable constraints, the four point plane-fit has high error or indicates excessive vehicle pitch or roll, or if the body clearance check indicates a collision. If all goes well with the placement check, the kinodynamic simulation is updated with the resulting vehicle pose. If the entire action succeeds, the terminal pose is added to the set of RRT nodes, and the process begins again. The tree continues to grow until either a goal is reached or a certain number of actions have been tested (note that some or all of them may fail).

When attempting to choose actions that reduce SLAM entropy, it may appear at first that there are an unlimited number of possible actions. However, the robot is very constrained in the actions that it can take, both because its controls are limited to velocity and horizontal curvature, and because many actions will result in collisions with obstacles. Thus we can restrict evaluation to candidates from the set of non-colliding actions that are reachable from the current robot state. An optimal planner, such as A*, would need to consider the effect on the SLAM uncertainty of all such candidate actions at a fine granularity.

In order to satisfy real-time constraints, we need to consider the effect on the SLAM uncertainty at a coarse, or macro, granularity; for example after the robot drives along a particular arc for a few meters. The RRT path planner generates reachable, non-colliding macro actions that rapidly explore the space of usable candidates, which we then evaluate in terms of SLAM uncertainty in an any-time fashion to find the best plan.

A more thorough approach would be to quantize the space of possible actions based on curvature and distance, and then to evaluate all possible actions, first checking traversability, and then evaluating the expected SLAM entropy. Sequences of actions could then be evaluated by repeating this process for every possible combination of actions. Since the SLAM entropy is generally difficult to predict, especially in the vicinity of loop closures, we believe it will be fruitless to apply best-first search algorithms, such as A* [Hart et al., 1968] (but see Gonzalez and Stentz [2007] for an admissible localization entropy heuristic under special conditions).

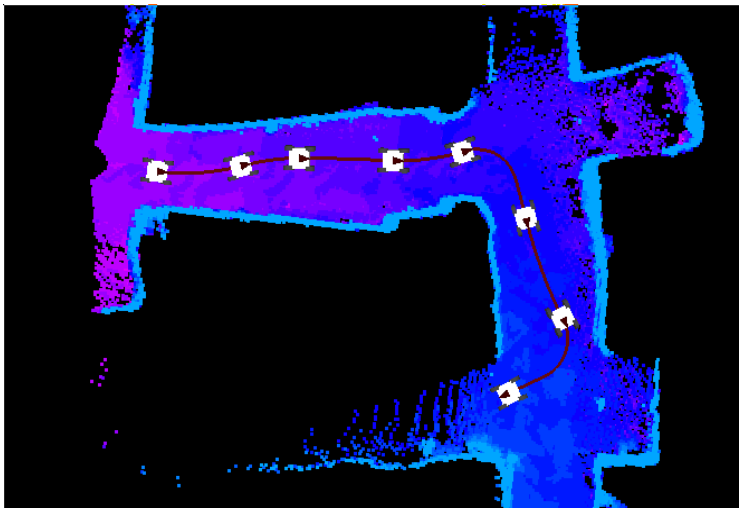

Figure 5.13: An RRT-planned trajectory through the narrow corridors of the Bruceton mine.

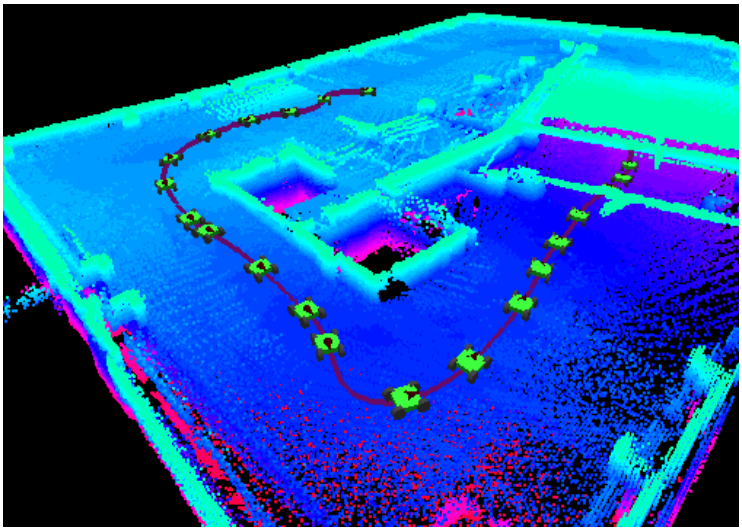

Figure 5.14: An RRT-planned trajectory extends down the parking garage ramp to a lower floor.

Planning Example

We introduced the Cave Crawler vehicle and the Bruceton mine and CIC parking garage in Chapter 4. Since Cave Crawler can't initially sense the space directly under the vehicle, we initialize the map with a small patch of floor under the vehicle. Figure 5.13 shows that the RRT planner can be used in relatively constricted and narrow tunnels, while Figure 5.14 shows that the RRT planner can handle slopes and multiple 3D levels.

Exploration Example

A simple metric to evaluate the exploration value of an RRT-generated plan state is to count the number of range beams that pass beyond the current map without encountering an obstacle.

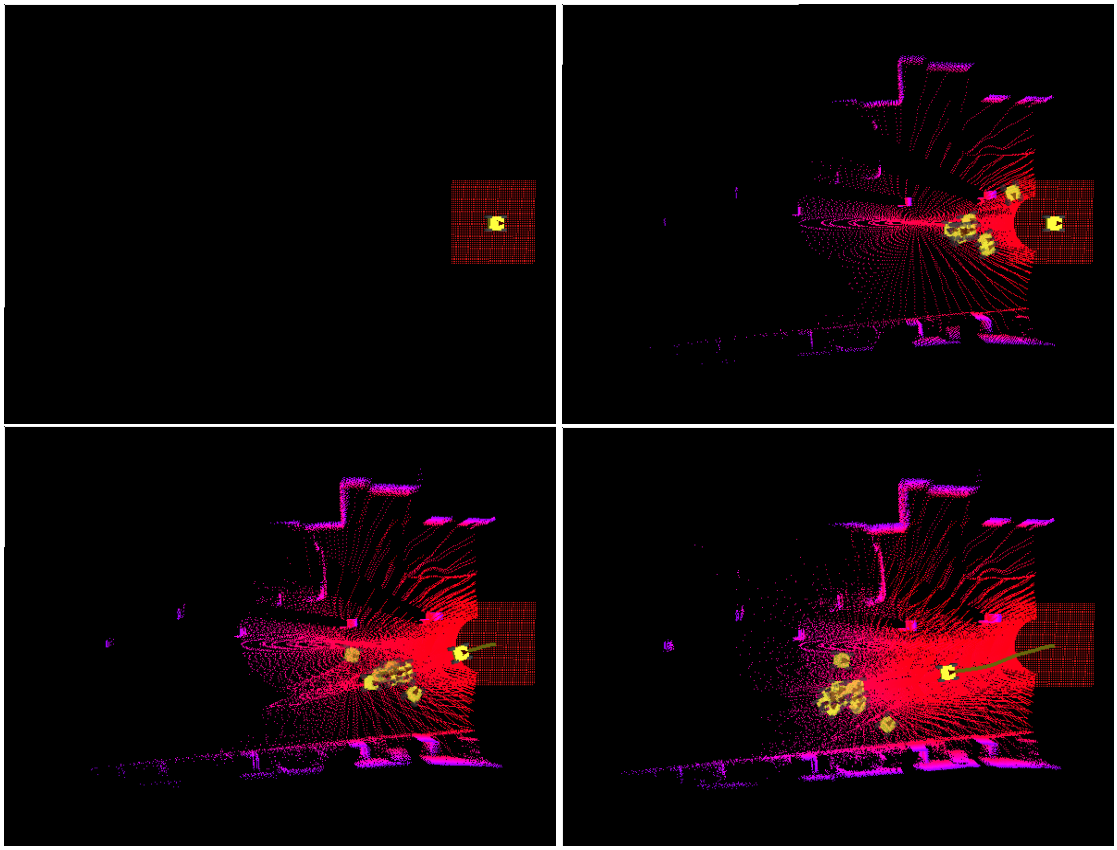

Figure 5.15: RRT exploration trajectories seek out final positions that are reachable and also are expected to yield lots of new map information. This sequence of visualizations show Cave Crawler's first few seconds of exploration in the CIC Garage, using the RRT planner and the unpredicted-beam metric to select exploration actions as the evidence grid map gradually grows.

These unpredictable beams are indicative of both map uncertainty and of frontiers, both of which are excellent goals for exploration. Figure 5.15 shows the initial stages of Cave Crawler's exploration of the CIC parking garage using the unpredictable beams metric. To select the next action, the RRT planner was used to generated a large number of possible actions with no particular goal. Each action was then weighted by applying the unpredictable beams metric to the action's terminal pose, and the planner selected the action with the highest weight (most unpredicted beams).

5.7 Active SLAM with the Segmented Map

What if we want to plan using the belief state encoded in the segmented map? We have already seen how to use the segmented map to generate hypothesized arrangements of segments, or map samples. Each map sample is a collection of segment maps and the relative transforms between them. To use the RRT planner in the full map sample (not just a single segment), the map sample can be fused, or merged, into a single octree evidence grid. This is done by creating a new "merged map" evidence grid, traversing each segment octree and transforming each cell into a common coordinate frame according to the map sample transforms, and adding the value of the cell to the merged map.

The RRT planner can then be applied fused map to find actions that explore, drive to a goal, etc. But even if the RRT planner successfully generates such a plan, all that really means is that the plan will succeed in that particular map sample. To more fully probe the SegSLAM belief state, as encoded by the segmented map, it is necessary to generate more map samples and evaluate the plan in those samples. We can then draw conclusions from the success rate of the plan in multiple samples, an approach that is generally called "Bayesian robustness" [Bagnell, 2004]. Broadly, we can imagine three categories of success rates: very high, very low, or evenly balanced.

Robust Plans

By evaluating a plan in multiple samples from the segmented map, we can get an idea of the robustness of the plan: if the plan succeeds in almost all of the samples, then the SegSLAM belief state (as represented by the segmented map) is certain about the part of the world that the plan passes through. If we imagine taking all of the fused maps from the map samples, and merging *them* into a single map, that "master map" would be sharp and clear in the region that a plan passes through. If we trust the SegSLAM belief state, then a plan with a high success rate is robust – it will (almost) surely succeed.

Discriminative Plans

We can use the same idea to instead find discriminative plans that probe regions of the world that the segmented map is *not* very certain about. If a plan works in a few map samples, but fails in the majority, that indicates that the SegSLAM belief state is uncertain in the region of the plan. The master map would be blurry or blocked in that region. According to the SegSLAM belief state, a discriminative plan has a high risk of failure, but it is also likely to yield some new information that will allow SegSLAM to discover which set of samples is correct. However, the discriminative plan may not result in net information gain – what new information is gained by exploring into the uncertain area may be canceled out by the fact that SegSLAM is uncertain about its position while exploring.

Information Gain Plans

Balanced between robust plans and discriminative plans are plans which succeed in about half of the map samples. These plans are somewhat robust because they succeed in several map samples,

and they are also somewhat discriminative because they fail in several map samples. As a result, they are likely to result in information gain, either by entering into new regions, or by returning to old uncertain regions. In either case, the robustness criteria ensures that SegSLAM knows enough about the region to localize well, while the discriminative criteria ensures that SegSLAM will still learn something new about the map.

This heuristic method for selecting actions that attempt to balance the mapping and localization uncertainty may prove to be a coarse real-time relative of the Bayesian Q-Learning exploration strategies of Dearden et al. [1998], particularly the myopic value of perfect information.

We somewhat inaccurately call our heuristic the *median successful action criteria*, and illustrate it with a real-world experiment after introducing a method of predicting unknown regions using the segmented map.

5.7.1 Prediction with the Segmented Map

Our overall goal is to plan for Active SLAM, finding actions that reduce the uncertainty in both mapping and localization. The planner needs to predict the outcomes of actions, which will generally limit the planning horizon to the borders of the current map. As a result, the most audacious action that horizon-constrained planner can recommend is to approach a frontier, or to more thoroughly investigate the site of a possible loop closure – but a possible loop closure can only be detected if some of the particles have already closed the loop! It would be far more useful if the planner could generate non-trivial actions that extend into or through unexplored regions. But by definition, the map is incomplete in these regions and information gain metrics will be very inaccurate, unless there is some good predictive model available.

SegSLAM provides such a model. The segmented map is an excellent model of the explored environment: it is a large collection of connected map segments that can be used as a predictive model to generate locally appropriate hypotheses about unexplored regions. We use the same process as for generating map samples, with the addition of a step that randomly selects a nearby segment, and grafts its start position onto the current position with some random perturbation or refinement from map matching. We will call this grafted segment a "guess," because although it is in fact a hypothesis about the local map structure, it is a much weaker hypotheses than the regular segments.

The grafting process is different than the usual map merging method used to fuse together segments. In map merging, the evidence values at each point in one evidence grid octree are summed with the evidence values at the transformed point in another octree. In the case of grafting, *positive* log-likelihood occupancy values are summed at full value, but *negative* log-likelihood occupancy values are clipped to a value of -1 before summing. Effectively, the obstacles in the grafted map are fused at full value, but the empty space is fused very tentatively. This means that the grafted segment can't carve holes in obstacles in the other segments, and that the empty space due to the grafted segment, which has a log-likelihood value of -1, is easily distinguishable from the empty space of the other segments.

The question is then how to make use of the "guess" yielded by the grafted segment. If the guess conflicts with the "known" portions of the map sample due to the regular segments, it is useless. If the guess extends into unmapped regions, it can be used to generate informed plans (for example allowing plans to extend down a hallway). But the real value of a guess is when

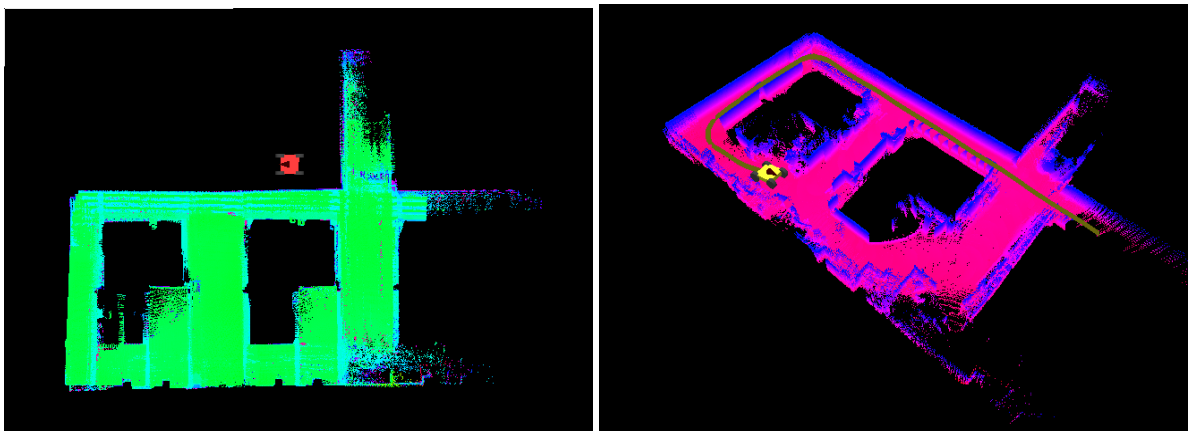

Figure 5.16: Overview of the catacombs environment – the long axis of the figure eight is about 20 m, and the short axis is about 10 m. In several places Cave Crawler can barely fit through the narrow tunnels.

it connects known regions while bridging across unknown regions: this allows Active SLAM to close *predicted* loops.

We have described the general procedure for grafting guesses onto a map sample. To accelerate the process of finding good plans, our planner also uses the following simplifications. As above, it first generates a local map sample and fuses the submaps into a single evidence grid map. It then generates a guess by selecting a random segment to append to the vehicle's current position, as well as a small perturbation. This guess segment includes the vehicle's original trajectory through the segment, so to quickly check the plausibility of the guess, our planner queries the fused map along the (transformed) trajectory to verify that the guessed trajectory starts in known empty space, goes into unknown space, and returns to known empty space. If the guess passes this quick check, it is grafted into the fused map, and the planner uses the full RRT vehicle model to see if the vehicle can pass through the resulting map. If the RRT model can go from known empty space to grafted empty space and back to known empty space without collision, the plan is considered a success for the map sample.

In the next section, we demonstrate this approach to Active SLAM with loop prediction using the median successful action criteria in a real-world experiment,

5.8 Active SLAM Experiment – FRC Catacombs

Beneath the Field Robotics Center highbay is a network of tunnels affectionately known as the catacombs. The navigable area of the catacombs forms a square figure eight shape (Figure 5.16). The tunnels of the catacombs are wide enough for Cave Crawler, but in some cases the wheels rub on both sides. While we have shown that our RRT-based planner is adroit at planning through narrow corridors, it failed it this extremely constrained environment. To allow the planner to function, we reduced the size of the virtual vehicle, and then manually drove the vehicle in a close approximation of the planned trajectory. As a second compromise, we clipped the laser ranges to a maximum of 4 m so that the vehicle could not see the end of the tunnel segments.

We started the vehicle at one end of the figure eight, and used the RRT planner to find explo-

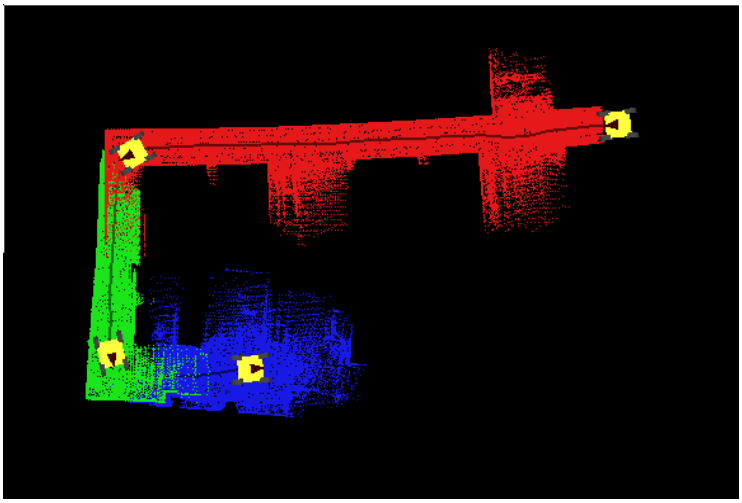

Figure 5.17: In both experiments, Cave Crawler started at one end of the figure eight (red) and then proceeded down one side and around the bottom.

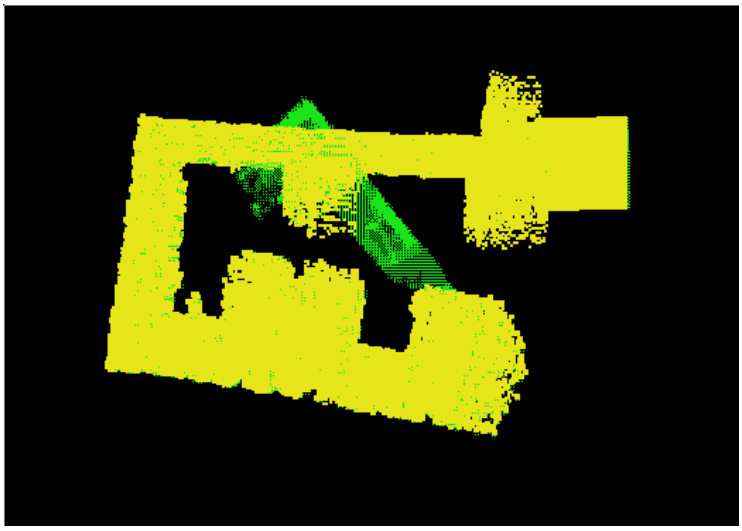

Figure 5.18: An example of a bad grafted segment guess that didn't work out – in this structured environment, most guesses fail.

ration actions with a bias toward moving in straight lines. After driving down one side, past the crossing tunnel and around the bottom of the loop, the vehicle encountered the other end of the crossing tunnel (Figure 5.17). At this point, it had three or four segments (varying between different runs) with which to make guesses. In this experiment, we evaluated guesses as described above, and checked their success in 10 map samples, considering the median success criteria to be satisfied if the action succeeded in 4-7 samples. Of course, most guesses result in collisions, as shown in Figure 5.18, and rarely have success rates above 2/10. But some guesses successfully bridged the middle segment of the figure eight, and the success rate of actions across these bridges depended on the SegSLAM uncertainty.

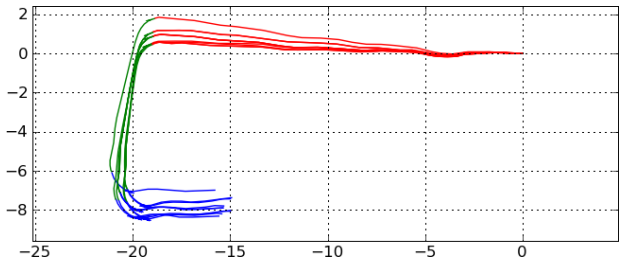

Figure 5.19: High uncertainty run: Trajectory samples from a segmented map with high uncertainty, as the vehicle approaches the crossing tunnel of the figure eight. Colors indicate segments.

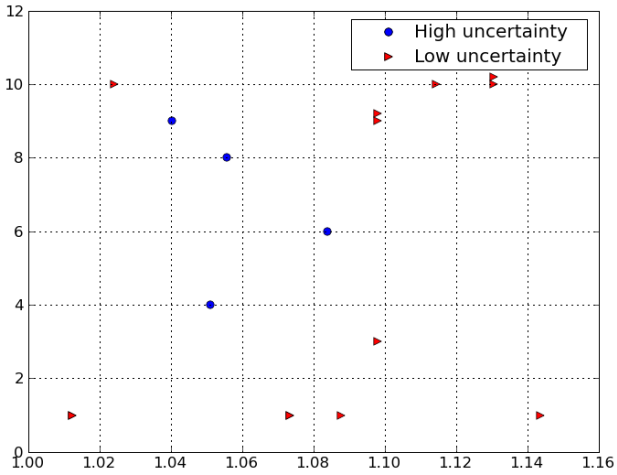

Figure 5.20: Action votes over the period of time when the vehicle approached the crossing tunnel, for both the high and low uncertainty runs.

To control the uncertainty, we artificially increased the motion model noise. When the SegSLAM uncertainty was high as illustrated by Figure 5.19, the bridging actions had success rates of 4/10 or 6/10 (Figure 5.20), and following the median successful action criteria we took the action and closed the loop (Figure 5.21). In the runs with low SegSLAM uncertainty, the bridging actions had either high or low success rates (Figure 5.20), and we did not cross the center tunnel, continuing to follow the usual exploration actions up and around the figure eight to close the loop (Figure 5.22).

This experiment provides some empirical justification for the media successful action criteria in a structured environment. We would like to investigate a more rigorous justification in the future.

5.8.1 Active Planning Experimental Summary

Our approach to Active SLAM can be summarized as using simplified models to heuristically estimate the entropy of the segmented map. In particular, the planner probes the segmented map

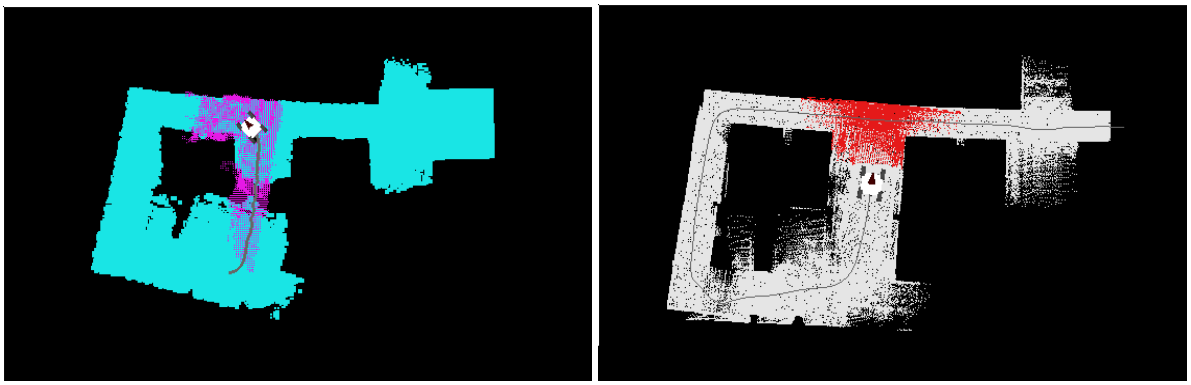

Figure 5.21: High uncertainty run: the guessed path through the center tunnel (left) had a success rate of 6/10, which satisfied the median successful action criteria. After taking this action, SegSLAM closed the inner loop (right).

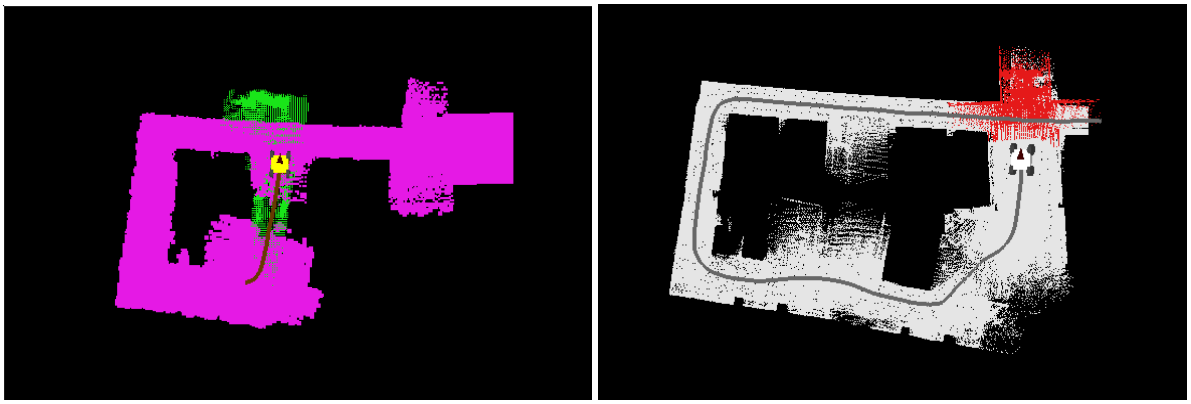

Figure 5.22: Low uncertainty run: the guessed path through the center tunnel (left) had a success rate of 9/10, which as too high to satisfy the median successful action criteria. The vehicle continued to follow the outside of the figure eight, and SegSLAM closed the outer loop (right).

by applying a simple traversability model to multiple map samples. The traversability model meshes well with our choice of an RRT approach to low-level path planning.

Extending the active localization case, we described how to do active SLAM with stochastic planning by generating map samples from the segmented map and selecting actions based on their success rate in multiple maps, preferring the median successful action. We then showed how to use the segmented map as a predictive model that can hypothesize about unseen regions by using nearby segments as priors.

Our approach depends on, but is not sensitive to, the RRT branching rate, determined by how new actions are added to the tree, and the RRT random action generator, so long as it covered the vehicle's range of motion. Nor is sensitive to the number of map samples that plans were evaluated as part of the voting process, though it was important to select actions with success rates near half of the total votes (median successful action).

5.9 Summary

In this chapter, we have investigated methods for selecting actions that balance exploration and localization, also known as active SLAM, without explicitly evaluating the SLAM entropy.

Using multibeam sonar data collected with the MBAUV, we have shown how we can use the most discriminative action criteria to do active localization without full-blown particle filter simulation or explicit entropy estimation. In simulation, we showed that the most discriminative action criteria clearly has a close relationship with the ideal expected entropy criteria.

Extending the active localization case, we described how to do active SLAM with stochastic planning by generating map samples from the segmented map and selecting actions based on their success rate in multiple maps, preferring the median successful action. We then showed how to use the segmented map as a predictive model that can hypothesize about unseen regions by using nearby segments as priors.

Integrating all these ideas, we have demonstrated active SLAM in an on-line experiment with Cave Crawler, which incorporates all of the ideas presented in this thesis, including octree evidence grids, SegSLAM, segmentation and RBPF SLAM within segments, map matching, predictions from the segmented map, and median successful action planning.

Chapter 6

Conclusions

In this thesis, we developed robust, real-time methods that allow robots to explore large, three dimensional, unstructured environments, and to allow subsequent operation over long periods of time. To avoid dependence on any particular set of environement-specific features, we used a featureless map representation based on evidence grids. Using a variety of robots with active range sensors, we demonstrated exploration and mapping of large three dimensional environments by factoring spatial uncertainty using a combined metric/topological map representation, and by integrating the tasks of localization, mapping, and planning so that the exploration strategy is informed both by the structure of the environment and by the ongoing uncertainty of the explorer.

6.1 Contributions

This thesis has made significant contributions to the field of robotics by developing a broadly-applicable approach for exploring and mapping large, unknown 3D environments.

We developed the Deferred Reference Count Octree (DRCO), an efficient data structure for building 3D metric maps using range measurements. The DRCO exploits spatial sparsity and homogeneity, allowing the construction of high-resolution kilometer-scale maps in real-time. We also created new methods for quickly comparing and matching these maps, effectively treating the maps themselves as macro features.

The DRCO is the innovation that allows the efficient implementation of a Rao-Blackwellized particle filter for 3D SLAM, described in chapter 3. We used our real-time RBPF SLAM implementation to produce the first metric 3D maps of large-scale underwater environments. This approach is particularly robust to low quality range sensor measurements, such as sonar, although we also demonstrate it with dense, high quality lidar data.

To address the problems of scalability and loop closure, we developed SegSLAM, a flexible and robust segmented SLAM method. As described in chapter 4, SegSLAM's maintains real-time particle filter performance, while the segmented map representation allows the particles to represent different metric and topological hypotheses. We have developed a simple segmentation heuristic based on the current map's ability to predict future measurements, and we demonstrated a robust method for detecting loop closures by using map samples from the segmented map to

narrow the search for nearby segments and matching segments using our DRCO map matching techniques. We believe that SegSLAM is the first segmented metric map approach to handle general 3D environments, including ramps, multiple superimposed levels, and complex 3D geometry.

Finally, we developed novel methods for integrating a stochastic planner with SegSLAM in order to perform Active SLAM. We described how to do action selection with stochastic planning by evaluating the success rate of plans in multiple map samples from the segmented map. This approach to stochastic planning allowed us to perform real-time Active Localization and Active SLAM. We also showed how segmented map is a useful model of the local environment that can be used to make predictions about unseen regions. This is the first approach to allow 3D predictive loop detection in real-time.

Throughout this document, we have empirically demonstrated the generality of our approach using four different robots in a variety of challenging environments, summarized here:

Robot	Sensor Type	Sensor Quality	Dead Reckoning Quality
DEPTHX	sonar array	poor	excellent / poor very near walls
Groundhog	nodding laser	excellent	good
Cave Crawler	spinning laser	excellent	fair
MBAUV	multibeam sonar	good	excellent / poor during dive & ascent

Location	Type	Scale	Structure
Dakota mine	subterranean	large (1000 m)	grid of tunnels
Bruceton mine	subterranean	large (1000 m)	tunnels with long loops
Test tank	underwater	small (15 m)	cylindrical, smooth
Flooded quarry	underwater	medium (200 m)	irregular, shallow
Zacatón	underwater	large (300 m)	irregular, chimney-shaped
La Pilita	underwater	medium (100 m)	irregular, jug-shaped
Caracol	underwater	medium (100 m)	irregular, jug-shaped
Verde	underwater	medium (100 m)	irregular, shallow
CIC Parking Garage	urban	medium (60 m)	3D ramps and loops
Gates building	urban	medium (80 m)	cluttered 3D ramps, passages
Axial Seamount	underwater	large (4000 m)	irregular ocean floor
FRC Catacombs	indoor	small (20 m)	tight steam tunnels

6.2 Future Work

There are two specific areas of future work that will follow from this dissertation. The first is an in-depth investigation of the long-term performance of SegSLAM, particularly with regard to dealing with low-probability regions of the segmented map. We have alluded to methods for discarding non-viable segments and for merging together well-registered segments, but we would like to go further into the matter.

The second specific area is a more formal characterization of our proposed stochatistic planning methods. Planning in multiple map samples is clearly a powerful approach, but we would

like to more carefully investigate the relationship of our statistical heuristics to information entropy, and place them in context with the broader POMDP-based planning formalism.

More generally, we would like to investigate the consistency properties of SegSLAM. In particular, SegSLAM can have very low uncertainty in the coordinate frame of the current segment, while having much higher uncertainty about the position of distant segments. In many task domains, this distance-related decay in certainty may align well with the needs of short and long-range planning.

The broad goal of our future work is to push robots out into the real world by developing robust, general methods for robotic exploration and operation in both exotic and mundane environments.

Appendix A

Calibration of a Spinning Laser

Laser beams are almost perfect rays, with very little beam spreading. The SICK LMS 200 beam geometry is a 180 degree fan of 361 beams at 0.5 degree steps. Accuracy is very high (1 cm) and noise is generally low (5 mm 1 σ). We find that the effective range is limited to about 30 m. In addition to range noise inherent in the SICK sensor, there are two sources of error which are specific to the rotating laser configuration: roll skew and misalignment.

Roll skew correction The SICK sweeps out a line of points by spinning a mirror about the instrument Z axis. The mirror spins at 75 Hz, and must spin twice in order to generate 361 0.5 degree resolution points, yielding a 37.5 Hz or 0.026 s update rate. In addition there are two roll rates, slow at $\sim 6.4^\circ/s$ and fast at $\sim 64^\circ/s$, ten times faster. This means that at the slow roll rate, the SICK rolls by $6.4 \times 0.026 = 0.17^\circ$ during the sweep, and $64 \times 0.026 = 1.6^\circ$ at the fast roll rate. Without compensating for this roll skew, a 30 m transverse beam (at the beginning or the end of the SICK sweep) would be offset by $30 \times \sin(1.7^\circ)$ = 0.9 m at the slow rate, or 9 cm at the slow rate. Fortunately, as long as the SICK is rolling at a smooth rate, it is simple to pre-compute the rolled beam geometry for each beam, although it must be remembered that the SICK mirror spins twice: first for the integer degrees and second for the half degrees, although these measurements are interleaved in the outgoing data stream.

Misalignment correction The misalignment of the SICK with the rotating jig will cause the same error regardless of the spin rate. In particular, it is inevitable that the optical axis of the SICK will not be perfectly aligned with the axis of roll. This misalignment can be parameterized as offsets in roll, pitch, yaw, x, y, and z – which describe a full 6DOF transformation. So long as the jig is rigid, the transformation does not depend on the spin angle.

We use nonlinear optimization to find the 6 misalignment parameters. To generate an error signal for the optimization process, we observed that the misalignment is clearly apparent when a single 360 degree roll rotation of the SICK is split according to whether the roll angle is less than or greater than 180 into two overlapping point clouds: during a 360 rotation, the line of the laser ranges sweeps each point twice (left half of Figure A.1). We then take several slices through this split cloud, perpendicular to the axis of roll, and compute the optimization error as the total nearest neighbor distance between the angle-split halves of each slice – and divide by

the square root of the number of points, so that the error from dense slices (close to the robot) doesn't dominate the error from sparse slices (further from the robot).

$$\text{err} = \sum_{\text{slices}} \frac{\text{nnDist}(\text{points}_1, \text{points}_2)}{\sqrt{\#_{\text{points}_1} + \#_{\text{points}_2}}}$$

After optimization, the split slices overlap closely (right half of Figure A.1). Note that we use at least two or three slices at 5m intervals in order to avoid local minima during the optimization. We then verify the parameters by looking at the alignment of other slices which were not used during the optimization.

A final source of error unique to Cave Crawler arises from slack in its differential suspension, which averages the angle between left and right side-frames. This slack can cause the pitch offset between the sensor platform and the wheels to change as the vehicle ascends or descends a slope, or even in response to bumps. We attempt to reduce this source of error by adding weight to one end of the sensor platform, and by calibrating the resulting pitch bias by measuring the platform pitch on reciprocal headings for a given location:

$$\beta_{\text{pitch}} = \frac{\text{pitch}_{0^\circ} + \text{pitch}_{180^\circ}}{2}.$$

Accurate pitch information is essential when the vehicle is exploring 3D environments, since it has no altitude sensor.

We assume that the entire laser sweep is collected from a single robot position. As noted above, the laser scan takes 0.026 s. If the vehicle is moving its standard 0.4 m/s, this would yield an error of 0.4 m/s * 0.026 s = 1 cm along the axis of motion. More significantly, the vehicle's attitude rates appear to peak around 10°/s (Figure A.2), yielding a heading error of 10°/s * 0.026 s = 0.26°. However, this error is in heading, such there are frequently 30 m beams pointing down the corridor that are deflected by 30 m * sin(0.26°) = 14 cm.

One possibility would be to exclude laser data collected during large attitude rate excursions. Unfortunately, the attitude excursions appear to be closely associated with *any* vehicle motion. Attempting to compromise, we could use laser data collected during times of relatively low angular rates, say $< 2^\circ$/s. If we also only use laser ranges out to 10 m, then the endpoint deflection is only 1 cm. In fact, we simply use all the data (under 30 m) and trust the evidence grid map representation to cope with any angular displacement in the long-range measurements.

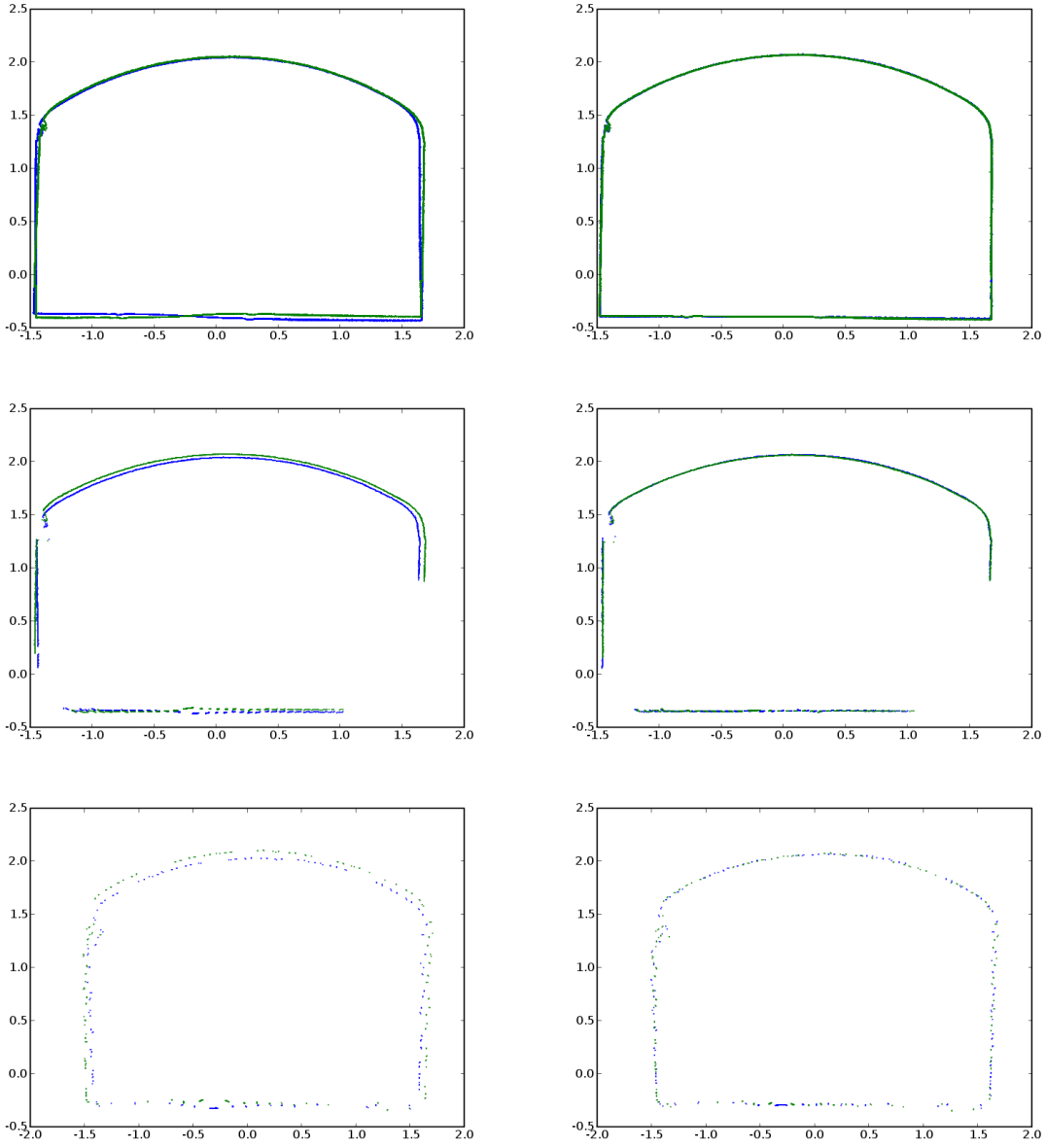

Figure A.1: On the left, slices at 5 m, 10 m, and 15 m of the data collected during a single 360 rotation of the SICK while the vehicle was stationary, showing the typical bread loaf tunnel cross section. The effects of the misalignment of the SICK are clear – gaps in the tunnel profile are due to shadowing effects from nearer obstacles. On the right, the same data showing the improvement resulting from correcting for the SICK misalignment on the rolling jig. The bottom row (15 m) was constructed using slices from fast roll rate data, showing that the misalignment correction computed from the slow data also applies to the fast data.

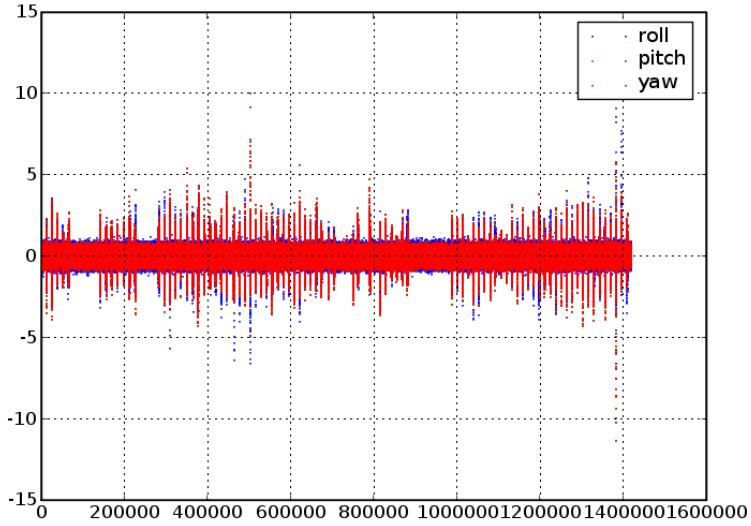

Figure A.2: Angular rates in °/s for Cave Crawler during typical driving conditions. Heading rate spikes to 10°/s.

Appendix B

Extended Kalman Filter for a 6-DOF IMU

Sources

Welch, G. and Gary, B. An introduction to the Kalman Filter. TR 95-041. 2006.
Titterton, D. and Weston, J. Strapdown Inertial Navigation Technology – 2nd Edition. The Institution of Electrical Engineers. 2004.
Diebel, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, 2006.

Generic EKF Algorithm

Generic EKF Notation

$g(\mu, u, w)$	State transition function: predicts state
G	State transition Jacobian w.r.t. state $= \delta g/\delta\mu\|_{\hat{\mu}_{t-1},u}$
W	State transition Jacobian w.r.t. state noise $= \delta g/\delta w\|_{\hat{\mu}_{t-1},u}$
$h(\mu, v)$	Measurement function: predicts measurement from state
H	Measurement function Jacobian w.r.t. state $= \delta h/\delta\mu\|_{\hat{\mu}_t}$
V	Measurement function Jacobian w.r.t. measurement noise $= \delta h/\delta v\|_{\hat{\mu}_t}$

EKF PREDICT: processing a new u

$$\begin{aligned} \mu &= g(\mu, u, 0) \\ \Sigma &= G\Sigma G^T + WQW^T \end{aligned}$$

EKF CORRECT: processing a new z

$$\begin{aligned} y &= z - h(\mu, 0) \\ S &= H\Sigma H^T + VRV^T \\ K &= \Sigma H^T S^{-1} \\ \mu &= \mu + Ky \\ \Sigma &= (I - KH)\Sigma \end{aligned}$$

a	Accelerations $= [a_x, a_y, a_z]^T$
ω	Angular rates $= [\omega_x, \omega_x, \omega_x]^T$
β	Gyro biases $= [\beta_x, \beta_y, \beta_z]^T$
α	Euler angles, 123 convention [roll, pitch, yaw]$^T = [\phi, \theta, \psi]^T$
$\dot{\alpha}$	Angle rates $= [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$
q	Quaternion $= [q_0, q_1, q_2, q_3]^T$
dt	Elapsed time

Table B.1: IMU EKF Notation

IMU EKF with Quaternion State

There are many representations for the state of the IMU. We track the angular rate bias of each gyro, and the IMU attitude as represented by a quaternion. Other common attitude representations include Euler angles and direction cosine matrices or rotation matrix, but the quaternion has several advantages: no singularities, overparameterization by just one value, and straightforward linearization (see the Ω matrix, below).

INITIALIZE Shorthand notation

$$s_\psi = \sin(\psi/2)$$
$$c_\psi = \cos(\psi/2)$$

Initialize quaternion with Euler angles

$$q = \begin{bmatrix} 2c_\phi c_\theta c_\psi + s_\phi s_\theta s_\psi \\ -c_\phi s_\theta s_\psi + s_\phi c_\theta c_\psi \\ c_\phi s_\theta c_\psi + s_\phi c_\theta s_\psi \\ c_\phi c_\theta s_\psi - s_\phi s_\theta c_\psi \end{bmatrix}$$

Initialize biases with zero

$$\beta = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Concatenate quaternion and biases to form initial state vector

$$\mu = \begin{bmatrix} q \\ \beta \end{bmatrix}$$

Initial covariance matrix is somewhat arbitrary

$$\Sigma = 0.000001 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

NOTE: To increase the stability of the yaw bias estimate (rotation about z axis), which is poorly measured, we set the last entry $\Sigma[6,6] = 0$.

Typical specifications for a Crossbow 400 IX IMU are: gyro random walk (σ_ω) 0.005 radians per second, 1 σ, and accelerometer bias (σ_β) 0.012 g per second, 1 σ.

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} (\sigma_\omega dt)^2$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sigma_\beta$$

PREDICT Controls are the IMU angular rates:

$$u = \omega = [\omega_x, \omega_y, \omega_z]^T$$

The Ω matrix is used to approximate the propagation of quaternions using angular rates

$$\dot{q} = \frac{1}{2}\Omega q$$

$$q = \left(I + \frac{1}{2}\Omega \Delta t\right) q$$

$$\Omega = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}$$

So, remembering to apply the biases

$$g(\mu, \omega) = [I_{4\times 4} + \frac{1}{2}\Omega \Delta t, I_{4\times 3}]\mu$$

$$g(\mu, \omega) = \frac{1}{2} \begin{bmatrix} +w2q_0 - q_1(\omega_x - \beta_x)\Delta t - q_2(\omega_y - \beta_y)\Delta t - q_3(\omega_z - \beta_z)\Delta t \\ 2q_1 + q_0(\omega_x - \beta_x)\Delta t + q_2(\omega_z - \beta_z)\Delta t - q_3(\omega_y - \beta_y)\Delta t \\ 2q_2 + q_0(\omega_y - \beta_y)\Delta t - q_1(\omega_z - \beta_z)\Delta t + q_3(\omega_x - \beta_x)\Delta t \\ 2q_3 + q_0(\omega_z - \beta_z)\Delta t + q_1(\omega_y - \beta_y)\Delta t - q_2(\omega_x - \beta_x)\Delta t \\ 2\beta_x \\ 2\beta_y \\ 2\beta_z \end{bmatrix}$$

and the Jacobian of g with respect to μ

$$G = \frac{1}{2} \begin{bmatrix} 2 & -(\omega_x - \beta_x)\Delta t & -(\omega_y - \beta_y)\Delta t & -(\omega_z - \beta_z)\Delta t & q_1\Delta t & q_2\Delta t & q_3\Delta t \\ (\omega_x - \beta_x)\Delta t & 2 & (\omega_z - \beta_z)\Delta t & -(\omega_y - \beta_y)\Delta t & -q_0\Delta t & q_3\Delta t & -q_2\Delta t \\ (\omega_y - \beta_y)\Delta t & -(\omega_z - \beta_z)\Delta t & 2 & (\omega_x - \beta_x)\Delta t & -q_3\Delta t & -q_0\Delta t & q_1\Delta t \\ (\omega_z - \beta_z)\Delta t & (\omega_y - \beta_y)\Delta t & -(\omega_x - \beta_x)\Delta t & 2 & q_2\Delta t & -q_1\Delta t & -q_0\Delta t \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

and the Jacobian of g with respect to w, or β

$$W = \frac{1}{2}\begin{bmatrix} q_1\Delta t & q_2\Delta t & q_3\Delta t \\ -q_0\Delta t & q_3\Delta t & -q_2\Delta t \\ -q_3\Delta t & -q_0\Delta t & q_1\Delta t \\ q_2\Delta t & -q_1\Delta t & -q_0\Delta t \\ 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

NOTE: We normalize the quaternion portion of the state vector after the prediction, in order to enforce that it is a unit quaternion.

$$q = |q| = \frac{q}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}}$$

CORRECT Measurement is the z-axis, the normalized acceleration

$$z = |a|$$

NOTE: By checking the magnitude of the acceleration vector, we can tell when the vehicle is accelerating. In that case, we do not perform the correction step, since the estimate of the z vector is perturbed.

To derive h, note that we can convert a quaternion to a rotation matrix as follows

$$M_q = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

and that we can extract the z-axis from a rotation matrix as

$$z_q = M_q[2,:]^T = [2(q_1q_3 - q_0q_2), 2(q_2q_3 + q_0q_1), q_0^2 - q_1^2 - q_2^2 + q_3^2]^T$$

So now we have

$$h(\mu) = \begin{bmatrix} 2(q_1q_3 + q_0q_2) \\ 2(q_2q_3 - q_0q_1) \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} + v$$

And so the Jacobian H is

$$H = \frac{\delta h}{\delta \mu}|_{\mu,0} = \begin{bmatrix} 2q_2 & 2q_3 & 2q_0 & 2q_1 & 1 & 0 & 0 \\ -2q_1 & -2q_0 & 2q_3 & 2q_2 & 0 & 1 & 0 \\ 2q_0 & -2q_1 & -2q_2 & 2q_3 & 0 & 0 & 1 \end{bmatrix}$$

and the Jacobian of h with respect to v is

$$V = \frac{\delta h}{\delta v}|_{\mu,0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

NOTE: Again, we normalize the quaternion portion of the state vector after the correction.

STATIONARY CORRECT When we know that the vehicle is stationary, we can begin to estimate the biases directly, basically by taking the average of the measurements.

The measurement of the biases are rates themselves – thus we have a linear measurement model and can use a Kalman Filter formulation for the correction (rather than an EKF formulation).

$$z = \omega$$

and

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and now we can plug this into the KF correct equations given above, with R the same as for the EKF.

NOTE: Again, we normalize the quaternion portion of the state vector after the correction. Technically, this should not be necessary since the stationary correct shouldn't affect the quaternion, but a little caution doesn't hurt.

OUTPUT We convert the quaternion state estimate as Euler angles

$$\hat{a} = \begin{bmatrix} arctan2(2(q_2q_3 + q_0q_1), 1 - 2(q_1^2 + q_2^2)) \\ -arcsin(2(q_1q_3 - q_0q_2)) \\ arctan2(2(q_1q_2 + q_0q_3), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

We also output the corrected Euler angle rates (see above for conversion from angular rates to Euler angle rates).

Bibliography

E. Acar, H. Choset, Y. Zhang, and M. Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International Journal of Robotics Research*, 22(1):441 – 466, July 2003.

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2): 174–188, February 2002.

A. Bachmann and S. Williams. Terrain aided underwater navigation - a deeper insight into generic monte carlo localization. In *Proc. of the Australasian Conf. on Robotics and Automation*, 2003.

J. Bagnell. *Learning Decisions: Robustness, Uncertainty, and Approximation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2004.

T. Bailey, J. Nieto, and E. Nebot. Consistency of the fastslam algorithm. In *Proc. of Intl. Conf. on Robotics and Automation*, pages 424–429, May 2006.

H. Baker. Minimizing reference count updating with deferred and anchored pointers for functional data structures. *ACM SIGPLAN Notices*, 29(9):38–43, 1994. ISSN 0362-1340.

S. Baker, R. Patil, K. Cheung, and I. Matthews. Lucas-kanade 20 years on: Part 5. Technical Report CMU-RI-TR-04-64, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, November 2004.

P. Besl and N. Mckay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal. Consistent observation grouping for generating metric-topological maps that improves robot localization. In *IEEE International Conference on Robotics and Automation (ICRA'06)*, May 2006.

J.-L. Blanco, J.-A. Fernández-Madrigal, and J. González. A novel measure of uncertainty for mobile robot slam with rao-blackwellized particle filters. *International Journal of Robotics Research*, 27(1):73–81, 2008a. ISSN 0278-3649.

J.-L. Blanco, J.-A. Fernández-Madrigal, and J. González. Towards a unified bayesian approach to hybrid metric-topological slam. *IEEE Transactions on Robotics*, 24(2):259–270, 2008b. ISSN 1552-3098.

J. L. Blanco, J. González, and J. A. Fernández-Madrigal. Subjective local maps for hybrid metric-topological slam. *Robot. Auton. Syst.*, 57(1):64–74, 2009. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2008.02.002.

M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework. *The Intl. Journal of Robotics Research*, 23(12):1113–1139, 2004.

F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, September 30 - October 4 2002.

D. Bradley, D. Silver, and S. Thayer. A regional point descriptor for global localization in subterranean environments. In *IEEE Conf. on Robotics Automation and Mechatronics*, December 2004.

J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1): 25–30, 1965.

N.A. Brokloff. Dead reckoning with an adcp. *Sea Technology*, pages 72–75, Dec 1998.

H. Bulata and M. Devy. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *Intl. Conf. Robotics and Automation*, volume 2, pages 1054–1060 vol.2, Apr 1996. doi: 10.1109/ROBOT.1996.506848.

W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization by entropy minimization. In *Advanced Mobile Robots, 1997. Proceedings., Second EUROMICRO workshop on*, pages 155–162, Oct 1997. doi: 10.1109/EURBOT.1997.633623.

W. Burgard, C. Stachniss, and G. Grisetti. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of the Learning Workshop (Snowbird)*, 2005.

A.R. Cassandra, L.P. Kaelbling, and J.A. Kurien. Acting under uncertainty: discrete bayesian models for mobile-robot navigation. *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, 2:963–972 vol.2, Nov 1996. doi: 10.1109/IROS.1996.571080.

H. Jacky Chang, C. S. George Lee, Yung-Hsiang Lu, and Y. Charlie Hu. P-slam: Simultaneous localization and mapping with environmental-structure prediction. *IEEE Transactions on Robotics*, 23(2):281–293, 2007.

Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2724–2729, 1991.

K. Chong and L. Kleeman. Feature-based mapping in real, large scale environments using an ultrasonic array. *I. J. Robotic Res.*, 18(1):3–19, 1999.

H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113 – 126, 2001.

H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(1):125 – 137, April 2001.

C. Connolly. Cumulative generation of octree models from range data. In *Proc. of IEEE Intl. Conf. Robotics and Automation*, volume 1, pages 25–32, Mar 1984.

R. Dearden, N. Friedman, and S. Russell. Bayesian q-learning. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 761–768, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7.

F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *Int. J. Rob. Res.*, 25(12):1181–1203, 2006. ISSN 0278-3649.

J. Diebel, K. Reuterswärd, S. Thrun, J. Davis, and R. Gupta. Simultaneous localization and mapping with active stereo vision. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 4, pages 3436– 3443, Sendai, Japan, 2004.

J. Djugash and S. Singh. A robust method of localization and mapping using only range. In *International Symposium on Experimental Robotics*, July 2008.

A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proc. of the Sixteenth Conf. on Uncertainty in AI*, pages 176–183, 2000.

T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Auton. Robots*, 12(3):287–300, 2002. ISSN 0929-5593. doi: http://dx.doi.org/10.1023/A:1015269615729.

A. Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1989.

A. Eliazar and R. Parr. Hierarchical linear/constant time slam using particle filters for dense maps. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 339–346. MIT Press, 2006.

C. Estrada, J. Neira, and J. Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, August 2005.

R. Eustice, H. Singh, and J. Leonard. Exactly sparse delayed-state filters. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pages 2428–2435, Barcelona, Spain, April 2005a.

R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Proc. of Robotics Science and Systems*, June 2005b.

R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: insights into sparsification. In *Intelligent Robots and Systems*, pages 3281–3288, Aug. 2005c. doi: 10.1109/IROS.2005.1545053.

N. Fairfield and D. Wettergreen. Active localization on the ocean floor with multibeam sonar. In *Proceedings of MTS/IEEE OCEANS*, 2008.

N. Fairfield, G. Kantor, and D. Wettergreen. Three dimensional evidence grids for SLAM in complex underwater environments. In *Proc. of the 14th Intl. Symposium of Unmanned Untethered Submersible Technology*, August 2005.

N. Fairfield, G. Kantor, and D. Wettergreen. Towards particle filter slam with three dimensional evidence grids in a flooded subterranean environment. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, May 2006.

H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *Int. J. Rob. Res.*, 18(7):650–668, 1999.

G. Ferri, M. Jakuba, E. Caselli, V. Mattoli, B. Mazzolai, D. Yoerger, and P. Dario. Localizing multiple gas/odor sources in an indoor environment using bayesian occupancy grid mapping. In *IROS*, pages 566–571, 2007.

B. Ferris, D. Hähnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Robotics: Science and Systems*, 2006.

M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. ISSN 0001-0782.

E. Fong, W. Adams, F. Crabbe, and A. Schultz. Representing a 3-d environment with a 2 1/2-d map structure. In *Proc. of the Intl. Conf. on Intelligent Robotics and Systems*, 2003.

D. Fox. Adapting the sample size in particle filters through KLD-sampling. *I. J. Robotic Res.*, 22(12):985–1004, 2003.

D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. In *Robotics and Autonomous Systems*, volume 25, pages 195–207, 1998.

D. Fox, J. Ko, K. Konolige, and B. Stewart. A hierarchical bayesian approach to the revisiting problem in mobile robot map building. *Intl. Symp. of Robotic Research*, 2003.

J. Freidman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977. ISSN 0098-3500.

U. Frese. A proof for the approximate sparsity of slam information matrices. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 329–335, April 2005.

U. Frese. Treemap: An o(log n) algorithm for indoor simultaneous localization and mapping. *Auton. Robots*, 21(2):103–122, 2006. ISSN 0929-5593.

U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *Robotics, IEEE Transactions on*, 21(2):196–207, April 2005. ISSN 1552-3098. doi: 10.1109/TRO.2004.839220.

S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *IJCAI*, pages 2109–2114, 2007.

P. Freedman G. Dudek, S. Hadres. Using local information in a nonlocal way for mapping graph-like works. In *Intl. Joint Conf. Artif. Intell.*, pages 1639–1645, 1993.

S. Hadjres G. Dudek, P. Freedman. Using multiple models for environmental mapping. *Robotic Systems*, 13(8):539–559, August 1996.

M. Gary. Understanding Zacatón: Exploration and initial interpretation of the world's deepest known phreatic sinkhole and related karst features, southern Tamaulipas, Mexico. *Karst Frontiers, Karst Waters Institute Special Publication*, 7:141–145, 2002.

N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Proc. of Fourth Intl Conf on 3-D Digital Imaging and Modeling*, pages 260–267, Oct. 2003.

J.-P. Gonzalez and A. Stentz. Planning with uncertainty in position using high-resolution maps. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, April 2007.

N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Proc. Inst. Elect. Eng. F*, volume 140, pages 107–113, April 1993.

M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. In *Proc. Fourth Intl. Conf. on 3-D Digital Imaging and Modeling, 2003*, pages 442–448, 2003.

G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 2443–2448, 2005.

B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. *Intl. Conf. on Robotics and Automation, 2003*, 1:1521–1526 vol.1, Sept. 2003. ISSN 1050-4729. doi: 10.1109/ROBOT.2003.1241807.

J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotic and Automation*, 2001.

J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation*, pages 318–325, 1999. doi: 10.1109/CIRA.1999.810068.

D. Hähnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient fastslam algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proc. of the Conf. on Intelligent Robots and Systems*, pages 27–31, 2003.

D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard. Towards lazy data association in slam. In Paolo Dario and Raja Chatila, editors, *ISRR*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 421–431. Springer, 2003.

C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

V. Havran. A summary of octree ray traversal algorithms. *Ray Tracing News*, 12(2), 1999.

A. Healey, P. An, and D. Marco. On line compensation of heading sensor bias for low cost AUV's. In *Proc. of the IEEE Workshop on Autonomous Underwater Vehicles*, pages 35–42, August 1998.

R. Henthorn, D. Caress, H. Thomas, W. Kirkwood, R. McEwen, C. Paull, and R. Keaten. High-resolution multibeam and subbottom surveys of submarine canyons and gas seeps using the mbari mapping auv. In *Proc of MTS/IEEE Oceans*, Boston, Mass., September 2006.

D. Huber and N. Vandapel. Automatic 3d underground mine mapping. In *International Conference on Field and Service Robotics*, July 2003.

M. Jefferies, M. Cosgrove, J. Baker, and W. Yeap. The correspondence problem in topological metric mapping - using absolute metric maps to close cycles. In *KES*, pages 232–239, 2004.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. Technical report, Brown University, Providence, RI, USA, 1996.

G. Kantor, N. Fairfield, D. Jonak, and D. Wettergreen. Experiments in navigation and mapping with a hovering auv. In *Intl. Conf. on Field and Service Robotics*, 2007.

James C. Kinsey, Ryan M. Eustice, and Louis L. Whitcomb. A survey of underwater vehicle navigation: Recent advances and new challenges. In *IFAC Conference of Manoeuvering and Control of Marine Craft*, Lisbon, Portugal, September 2006. Invited paper.

W. Kirkwood, D. Caress, H. Thomas, M. Sibenac, R. McEwen, F. Shane, R. Henthorn, and P. McGill. Mapping payload development for mbari's dorado-class auvs. *OCEANS '04 MTS/IEEE*, 3:1580–1585 Vol.3, Nov. 2004.

J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. In *IROS*, pages 3471–3476, 2008.

S. Koenig and R. Simmons. Exploration with and without a map. In *Proc. of the Workshop on Learning Action Models, National Conference on Artificial Intelligence*, Washington, DC, July 1993.

T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. *Int. J. Rob. Res.*, 27(2):175–196, 2008. ISSN 0278-3649. doi: http://dx.doi.org/10.1177/0278364907087426.

D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *AAAI'94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 979–984, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3.

B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robot. Auton. Syst.*, pages 47–63, 1991.

R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard. Active monte carlo localization in outdoor terrains using multi-level surface maps. In *Proc. of Field and Service Robotics*, Chamonix, France, 2007.

M. Larsen. High performance doppler-inertial navigation – experimental results. In *Proc. of IEEE/MTS OCEANS*, pages 1449–1456, 2000.

S. LaValle and J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.

J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, Jun 1991. ISSN 1042-296X. doi: 10.1109/70.88147.

J. Leonard, A. Bennett, C. Smith, and H. Feder. Autonomous underwater vehicle navigation. Technical report, MIT Marine Robotics Laboratory, 1998.

J.J. Leonard and H.J.S. Feder. Decoupled stochastic mapping. *Oceanic Engineering, IEEE Journal of*, 26(4):561–571, Oct 2001. ISSN 0364-9059. doi: 10.1109/48.972094.

M. Likhachev and A. Stentz. PPCP: Efficient Probabilistic Planning with Clear Preferences in Partially-Known Environments. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, volume 1, pages 448 – 453, October 2003.

J. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, June 1996.

W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. ISSN 0097-8930.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robotics*, 4:333–349, April 1997.

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

R. Madhavan, M. Dissanayake, and H. Durrant-Whyte. Autonomous underground navigation of an LHD using a combined ICP-EKF approach. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, volume 4, pages 3703–3708, May 1998.

M. Magnusson. *3D Scan Matching for Mobile Robots with Application to Mine Mapping*. Licentiate thesis, Örebro University, September 2006.

I. Mahon and S. Williams. Three-dimensional robotic mapping. In *Proc. of the Australasian Conf. on Robotics and Automation*, Dec. 1-3 2003.

A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, Lausane, Switzerland, October 2-4 2002.

M. Martin and H. Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

R. Martinez-Cantin and J.A. Castellanos. Unscented slam for large-scale outdoor environments. In *Intl. Conf. on Intelligent Robots and Systems*, pages 3427–3432, Aug. 2005. doi: 10.1109/IROS.2005.1545002.

R. Martinez-Cantin, N. de Freitas, and J. Castellanos. Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-slam. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

M. Mataric. A distributed model for mobile robot environment-learning and navigation. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1990.

J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 1530–1536, 2004.

M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of the AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121, March 1985.

A. Morris, D. Ferguson, Z. Omohundro, D. Bradley, D. Silver, C. Baker, S. Thayer, W. Whittaker, and W. Whittaker. Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57, January 2006.

D. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. In *CGC 2nd Annual Fall Workshop on Computational Geometry*, 1997. available at *http://www.cs.umd.edu/mount/ANN*.

P. Moutarlier and R. Chatila. An experimental system for incremental environment modeling by an autonomous mobile robot. In *ISER*, pages 327–346, 1989.

K. Murphy. Bayesian map learning in dynamic environments. In *Neural Information Processing Systems*, pages 1015–1021, 1999.

B. Nabbe, S. Kumar, and M. Hebert. Path planning with hallucinated worlds. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, volume 4, pages 3123 – 3130, October 2004.

P. Newman, M. Bosse, and J. Leonard. Autonomous feature-based exploration. In *ICRA*, pages 1234–1240, 2003a.

P. Newman, J. Leonard, and R. Rikoski. Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *Eleventh Intl. Symposium of Robotics Research*, pages 409–420, 2003b.

A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam with approximate data association. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, pages 242–249, Seattle, USA, 2005.

M. Paskin. Thin junction tree filters for simultaneous localization and mapping. Technical Report UCB/CSD-02-1198, EECS Department, University of California, Berkeley, Sep 2002.

L. Paz and J. Neira. Optimal local map size for ekf-based slam. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 5019–5025, Beijing, China, 2006.

L.M. Paz, P. Jensfelt, J.D. Tardós, and J. Neira. EKF SLAM updates in O(n) with Divide and Conquer SLAM. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'07)*, Rome,Italy, April 2007.

J. Propp and D. Wilson. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27(2):170–217, 1998. ISSN 0196-6774.

F. Ramos, D. Fox, and H. Durrant-Whyte. Crf-matching: Conditional random fields for feature-based scan matching. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

A. Ranganathan and F. Dellaert. Inference in the space of topological maps: an mcmc-based approach. In *Intelligent Robots and Systems*, volume 2, pages 1518–1523 vol.2, Sept.-2 Oct. 2004. doi: 10.1109/IROS.2004.1389611.

E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artif. Intell.*, 152 (1):47–104, 2004. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(03)00114-0.

C. Roman. *Self Consistent Bathymetric Mapping from Robotic Vehicles in the Deep Ocean*. PhD thesis, Massachusetts Institute of Technology & Woods Hole Oceanographic Institution, May 2005.

N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12, pages 1043–1049, 1999.

N. Roy, G. Gordon, and S. Thrun. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

F. Savelli and B. Kuipers. Loop-closing and planarity in topological map-building. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems.*, Sendai, Japan, 2004.

A. Schultz and W. Adams. Continuous localization using evidence grids. In *Proc of Intl. Conf. on Robotics and Automation*, volume 4, pages 2833–2839, 1998.

D. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.

C. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, july, october 1948.

H.-Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *Int. J. Comput. Vision*, 36(2):101–130, 2000. ISSN 0920-5691.

R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

S. Simhon and G. Dudek. A global topological map formed by local metric maps. In *IROS*, volume 3, pages 1708–1714, Victoria, Canada, October 1998.

Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

H. Singh. *An Entropic Framework for AUV Sensor Modelling*. PhD thesis, Woods Hole Oceanographic Institution and Massachussetts Institute of Technology, May 1995.

R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Intl. Symp. Robot. Res.*, pages 467–474, 1988.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehicles*, pages 167–193, 1990.

C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, 2006.

C. Stachniss, G. Grisetti, D. Hähnel, and W. Burgard. Improved rao-blackwellized mapping by adaptive sampling and active loop-closure. In *Proc. of the Workshop on Self-Organization of AdaptiVE behavior*, pages 1–15, 2004.

W. Stewart. Multisensor modeling underwater with uncertain information. Technical Report AITR-1143, Massachusetts Institute of Technology, Cambridge, MA, USA, July 1988.

W. Stone, B. am Ende, F. Wefer, and N. Jones. Automated 3d mapping of submarine tunnels. In *Robotics 2000*, pages 148–157, 2000.

J. Tardós, J. Neira, P. M. Newman, and J. J. Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4):311–330, April 2002.

D. Thompson. *Intelligent Mapping for Autonomous Robotic Survey*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2008.

S. Thrun. Exploration and model building in mobile robot domains. In *In Proceedings of the IEEE International Conference on Neural Networks*, 1993. IEEE Neural Network Council.

S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artif. Intell.*, 99 (1):21–71, 1998. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00078-7.

S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15 (2):111–127, 2003.

S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

S. Thrun, D. Haehnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. L. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, May 2003.

S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 2004.

R. Urick. *Principles of Underwater Sound*. McGraw-Hill, New York, NY, 3rd edition, 1983.

A. Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 54:426–482, 1943.

M. Walter, R. Eustice, and J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *Intl. J. Robotics Research*, 26(4):335–359, April 2007.

J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *Proc. of Intelligent Robots and Systems*, pages 3834–3839, 2005.

L. Whitcomb, D. R. Yoerger, and H. Singh. Combined doppler/LBL based navigation of underwater vehicles. In *Proc. of the 11th Intl. Symposium on Unmanned Untethered Submersible Technology*, August 1999.

W. Whittaker and S. Thayer. Dakota mine no. 2 robotic survey, Bob White, West Virginia. Technical report, Robotics Institute, Carnegie Mellon University, August 2005.

S. Williams. A terrain aided tracking algorithm for marine systems. In *Intl. Conf. on Field And Service Robotics*, pages 55–60, July 14-16 2003.

S. Williams and I. Mahon. Simultaneous localisation and mapping on the great barrier reef. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 1771–1776, April 26-May 1 2004.

S. Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pages 22–28, April 2000.

S. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *IEEE Intl. Conf. Robotics and Automation*, volume 1, pages 406–411 vol.1, 2002. doi: 10.1109/ROBOT.2002.1013394.

B. Yamauchi and P. Langley. Place learning in dynamic real-world environments. In *Proc. of RoboLearn 96*, pages 123–129, 1996.