Segmented SLAM in Three-Dimensional Environments

Nathaniel Fairfield, David Wettergreen, and George Kantor
The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
e-mail: than@cmu.edu, dsw@cmu.edu, kantor@cmu.edu

Received 12 January 2009; accepted 17 August 2009

Simultaneous localization and mapping (SLAM) has been shown to be feasible in many small, two-dimensional, structured domains. The next challenge is to develop real-time SLAM methods that enable robots to explore large, three-dimensional, unstructured environments and allow subsequent operation in these environments over long periods of time. To circumvent the scale limitations inherent in SLAM, the world can be divided up into more manageable pieces. SLAM can be formulated on these pieces by using a combination of metric submaps and a topological map of the relationships between submaps. The contribution of this paper is a real-time, three-dimensional SLAM approach that combines an evidence grid–based volumetric submap representation, a robust Rao–Blackwellized particle filter, and a topologically flexible submap segmentation framework and map representation. We present heuristic methods for deciding how to segment the world and for reconstructing large-scale metric maps for the purpose of closing loops. We demonstrate our method on a mobile robotic platform operating in large, three-dimensional environments. © 2009 Wiley Periodicals, Inc.

1. INTRODUCTION

The tasks of mapping and localization lie at the core of mobile robotics. 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. To a large degree, the SLAM problem has been solved for small, two-dimensional (2D), structured environments (Eliazar & Parr, 2006; Grisetti, Stachniss, & Burgard, 2005; Montemerlo, 2003). To make robots useful in the broader world, they need to move beyond such simple environments into large, three-dimensional (3D), unstructured environments. Accordingly, they need general algorithms for mapping and localizing that work just about anywhere: indoors and outdoors; subterranean, aerial, and underwater; natural and urban; flat and highly 3D. Such algorithms must fit within the computational constraints of real mobile robots: constant time in computation and linear in storage space. 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. On the other hand, whereas maps need to be locally accurate for path planning and obstacle avoidance, global accuracy can often be relaxed, again depending on the robot and its task.

Many online SLAM methods work well on a limited scale. In particular, they build useful 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 is a group of SLAM methods that explicitly exploit spatial sparsity by segmenting the world into independent submaps. Most of these methods use a combination of metric and topological maps (Bosse, Newman, Leonard, & Teller, 2004; Estrada, Neira, & Tardós, 2005; Jefferies, Cosgrove, Baker, & Yeap, 2004; Lisien et al., 2003; Newman, Leonard, & Rikoski, 2003; Schultz & Adams, 1998; Tardós, Neira, Newman, & Leonard, 2002; Yamauchi & Langley, 1996) in which the nodes of the graph are metric submaps and the relationships between submaps are represented by the edges of a graph. The submap segmentation is usually designed such that their scale is well within the capabilities of a particular SLAM approach. 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 submap, when to reenter an old submap, and how to represent different hypotheses about the topological relationships between submaps.

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 submap or reenter an old submap: weighting, resampling, and updating are still applied.

At the core of our approach is a sparse map representation that is based on 3D evidence grids and octrees (Fairfield, Kantor, & Wettergreen, 2007). This representation does not rely on features, which may be available only 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 Rao–Blackwellized particle filter (RBPF) SLAM (Doucet, de Freitas, Murphy, & Russell, 2000) method on this metric map representation (Fairfield, Kantor, Jonak, & Wettergreen, 2008). Owing to its RBPF foundations, our method has real-time performance 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. Owing to its

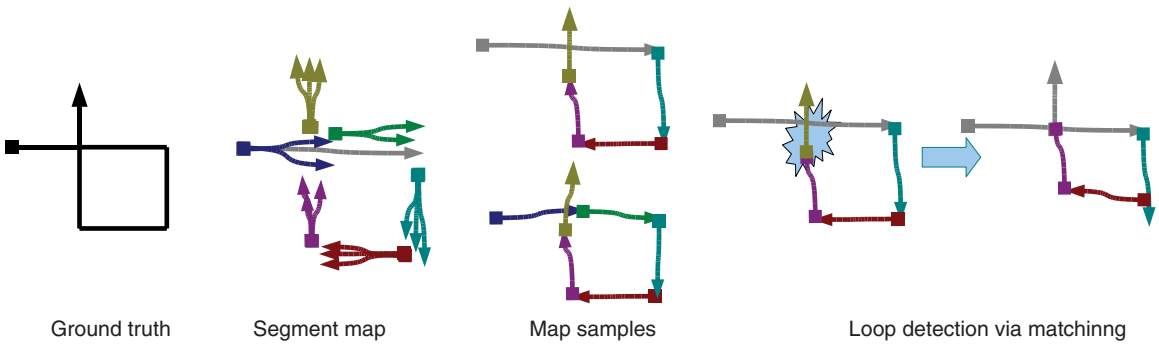

Figure 1. The process of segmentation, map sample generation, and matching. The SegMap stores the particle submaps (shown as different arrows) for each color-coded segment. The SegMap also stores the relationships between segments, loosely illustrated here by the segment placements relative to each other. Note that after matching, the breadth-first map sampling algorithm does not enforce global consistency between the red and turquoise (lower right) segments.

basis in evidence grids and particle filters, our method is robust to poor sensor data. Our SLAM algorithm has been used in several challenging environments, including flooded sinkholes (Fairfield et al, 2007) and the ocean floor (Fairfield & Wettergreen, 2008).

SegSLAM extends our RBPF SLAM method to yet larger scales by using a heuristic to find good segmentations between submaps, based on the ability of the current submap to predict future measurements. Mapping and localization within these submaps is performed using a particle filter similar to our basic RBPF SLAM method but generalized to allow particles to transition between submaps. Whereas the particles of a regular RBPF are samples from the distribution over poses and the metric maps, the particles of SegSLAM are samples from the distribution over poses and submaps, in which the poses are in the local coordinate frame of the submaps. Because each particle has its own copy of each submap, we use the noun *segment* to refer to the collection of particle submaps that are temporally compatible: 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*, or SegMap.

The SegMap is a probabilistic graph in which the particle pose transformations from one segment to another are the edges of the graph, but there may be many different metric submaps for each segment, one for each particle. A transformation between two segments can be used to stitch together any two particular submaps from the corresponding segments. Reconstructing a trajectory can be seen as drawing a sample from the SegMap probabilistic graph by stochastically picking edges and nodes in a breadth-first fashion to create a (partial or complete) metric map, which can then be used for loop detection and planning. It is important to note that the particles do not have to segment or reenter at the same time, but the resulting segments will not be temporally compatible and cannot form part of a reconstructed trajectory. In particular, the ability of different particles to independently segment and reenter is how SegSLAM can represent different topologies. Figure 1 provides an illustration of the SegSLAM algorithm.

Another way of interpreting sampling from the SegMap is in the context of particle depletion, which is the limiting factor on the scalability of RBPF SLAM approaches (Stachniss, 2006). 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: 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 addresses this situation in two ways. First, the SegMap represents an exponential number of trajectories (submap combinations), which can be sampled as needed. Second, topological relationships between submaps (including loop closures) are discovered and refined by using the map matching techniques described in Fairfield and Wettergreen (2009) to directly match the evidence grid submaps, meaning that loop closures can be detected even when there is significant error in the sampled trajectory.

Taken together, the components of our approach form a broadly applicable method that enables practical robotic operation in large, 3D environments. This approach contributes to the field of robotics by developing a compact and efficient map representation for 3D environments; developing algorithms for building, copying, and matching these maps, effectively treating maps themselves as macro features; and developing SegSLAM, a robust, real-time, multihypothesis, segmented SLAM approach that addresses the problems of segmentation, particle depletion, loop closure, and scalability in 3D environments.

The rest of the paper is organized as follows. Section 2 summarizes our relevant prior work, particularly our map representation and RBPF SLAM approach for large unstructured 3D environments. Section 3 then describes related work in the area of segmented SLAM. Section 4 introduces our SegSLAM method. Section 5 presents experimental results from using SegSLAM to map several 3D environments. Finally, Section 6 presents our conclusions and thoughts for future work.

2. SUMMARY OF PRIOR WORK

2.1. 3D Maps

We previously introduced the deferred reference count octree (DRCO) evidence grid data structure, an efficient 3D volumetric representation that exploits the spatial sparsity of many environments (Fairfield et al., 2007). In addition, the DRCO implicitly exploits the common map structure of resampled RBPF particles, meaning that only the differences between particle maps have to be stored.

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*. 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 that 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 is a drop-in replacement for uniform voxel arrays in memory: it is possible to convert losslessly between the two representations. Ray insertion and query can be done with a tree-traversing ray-tracing algorithm (see Havran, 1999, for an overview).

2.2. Rao–Blackwellized Particle Filter

In Fairfield et al. (2007), we described our approach to robust real-time 3D SLAM with a Rao–Blackwellized particle filter and DRCO map representation. Our approach efficiently exploits spatial *sparsity* because the DRCO compactly represents large unobservable regions and/or large homogeneous 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 the copy-on-write capability of the DRCO.

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. It uses range data rather than features: the 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 SLAM approach works in large 3D environments with arbitrary (though static) geometry, using sparse and noisy range sensors. As with all RBPF SLAM algorithms, this method is susceptible to particle depletion, which ultimately limits the scale of the algorithm to a few hundred meters, although we have shown that opportunistically using more particles ameliorates the problem (Fairfield et al., 2007).

In Section 4, we will show how to use our RBPF SLAM as a building block for a segmented SLAM approach that attempts to address particle depletion and large-scale loop closure, but we first discuss related work in submap SLAM.

3. RELATED WORK IN SUBMAPS AND LARGE-SCALE SLAM

Passing over the large body of work on SLAM and the many different approaches (see Thrun, Burgard, & Fox, 2005, for a survey), we focus here on related work in submap 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 updating only one submap at any given time. The difficulty with a submap approach is then deciding when to build a new submap, how to reenter old submaps, and how to estimate the transforms between submaps.

Submap representation. Submap methods usually combine both metric and topological representations, in which 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 (Choset & Nagatani, 2001; Kortenkamp & Weymouth, 1994; Remolina & Kuipers, 2004). This metric information is usually represented as feature-based maps, for example, Bosse et al. (2004), Estrada et al. (2005), Lisien et al. (2003), Newman et al. (2003), and Tardós et al. (2002), but evidence grid–based submaps are not uncommon (Jefferies, Cosgrove, Baker, & Yeap, 2004; Schultz & Adams, 1998; Yamauchi & Langley, 1996).

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. Statistical independence is often asserted by giving each submap its own reference frame; in an early approach, each landmark had its own frame (Bulata & Devy, 1996). Decoupled stochastic mapping (Leonard & Feder, 2001) is somewhat unique in that it divides the environment into overlapping submaps, but these maps share a global reference frame: as a result, this approach is fast but overconfident about transforms between submaps.

The best segmentations can be found only in retrospect, in postprocessing, 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 & Byun, 1991), based on estimates of accumulated localization error (Bosse et al., 2004; Chong & Kleeman, 1999), the maximum desired number of features in each submap (Estrada et al., 2005; Tardós et al., 2002), the detection of special feature-rich regions (Simhon & Dudek, 1998), or even in postprocessing for offline approaches (Friedman, Pasula, & Fox, 2007; Thrun, 1998). Frese (2006) describes the Treemap algorithm, an $O(\log n)$ approach that uses a hierarchically subdivided map. Lisien et al. (2003) use the generalized Voronoi graph as motive behind segmentation (and as the topological map), a simple distance criterion for creating new landmarks, and then combine local maps by aligning the landmarks along the edges between the maps. Blanco, González, and Fernández-Madrigal (2009) provide a probabilistically grounded method for segmenting based on normalized cuts.

We use limited look ahead and a predictive score metric that estimates how well the current map predicts future measurements, combined with a localization error metric, as our segmentation criterion.

Matching. Matching can be thought of as evaluating hypotheses about the metric relationship between two submaps. 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). The constant-time SLAM algorithm (Newman et al., 2003) opportunistically fuses the feature estimates from multiple submaps to improve the global feature estimate (and the relations between submaps). Grisettio, Tipaldi, Stachniss, Burgard, and Nardi (2007) use the implicit similarity between globally referenced submaps, called patches, to improve their proposal distribution, reducing the computational and memory requirements for an RBPF. Evidence grid–based matching methods use overlap (Jefferies et al., 2004) or evidence grid matching (Yamauchi & Langley, 1996) to detect matchings. In our approach to matching, we apply novel evidence grid matching techniques (Fairfield & Wettergreen, 2009) to register the submaps.

Topological hypotheses. If matching tests pairwise hypotheses about submap connections, topological hypotheses encompass all the submaps and how they fit together. Modayil, Beeson, and Kuipers (2004) discuss a framework for dealing with uncertainty and error between the local metric, global topological, and global metric levels, but multihypothesis topological methods usually do not impose loop consistency (global optimality) in the interests of speed.

Some submap methods support only 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 yields only a single topological hypothesis. The closely related compressed filter (Guivant & Nebot, 2001), local map sequencing (Tardós et al., 2002), and constrained local submap filter (Williams, Dissanayake, & Durrant-Whyte, 2002) methods 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, Jensfelt, Tardós, & Neira, 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 (Dudek, Freedman, & Hadjres, 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 hypotheses. The ATLAS criteria for selecting topologies are somewhat ad hoc, and Ranganathan and Dellaert (2004) apply more rigorous Bayesian inference to what they call probabilistic topological maps. They use a Markov-chain Monte Carlo (MCMC) approach to estimate the distribution over possible topologies by sampling from the space of partitions of landmark measurements. Similarly, the hybrid metric-topological SLAM of Blanco, Fernández-Madrigal, and González (2008) 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 SegMap data structure that underlies our SegSLAM approach is 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 map reconstruction is used to grow large-scale metric map samples from the current particles as needed, for example while searching for loop closures. Thus SegSLAM can consider an exponentially larger set of topologies than standard RBPF SLAM in an any-time fashion (see Figures 2 and 3). SegSLAM is thus a constant-time/any-time algorithm based on the RBPF but integrated with a probabilistic topological map.

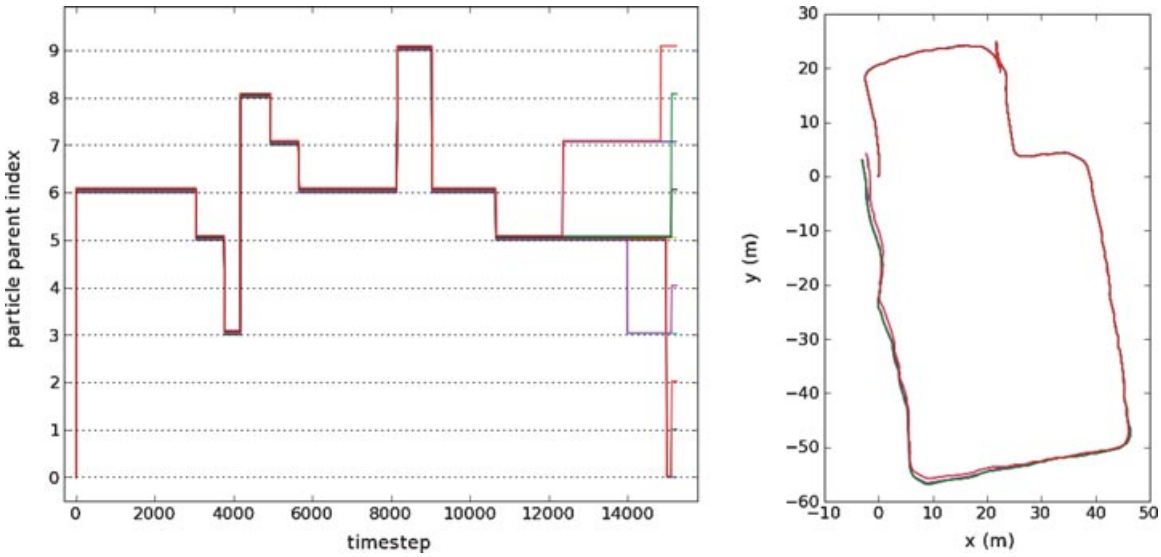

Figure 2. Trajectory depletion: Results from running our RBPF with 10 particles around a loop in Bruceton Mine. Left: The resampling ancestry of the particles shows that all the particles share a fairly recent common ancestor (particle 5). Right: The particle trajectories tell the same story, showing two competing hypotheses emerging near the lower right-hand corner. With only a single hypothesis about the vehicle position at the beginning of the loop, the particle filter fails to close the loop—no surprise with only 10 particles.

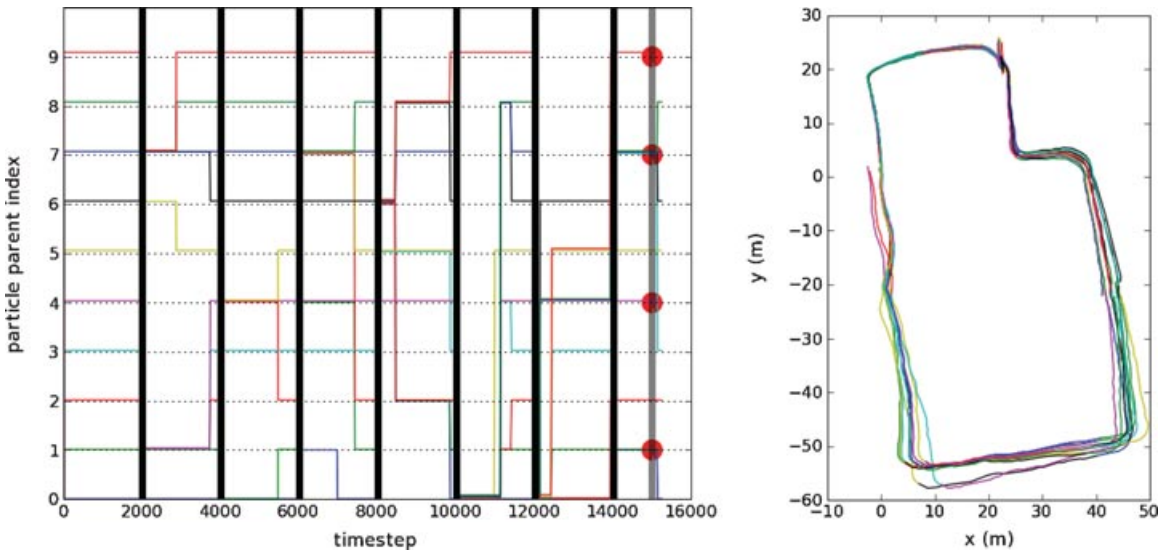

Figure 3. Maintaining trajectory variety: Results from running SegSLAM with 10 particles around a loop in Bruceton Mine. Left: The resampling ancestry of the particles shows how segmentation, illustrated by vertical black lines, maintains particle diversity, while still discarding unlikely hypotheses. 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 did not. In this plot, different colors indicate different segments as well as different particles.

4. SEGMENTED SLAM

4.1. Algorithm

SegSLAM extends the standard RBPF SLAM formulation (for example, see Montemerlo, Thrun, Koller, & Wegbreit, 2002) by extending the prediction step to include the selection of the current particle submap, in addition to the particle position within the submap. To create new submaps, SegSLAM applies a segmentation heuristic that uses an estimate of the gradual accumulation of motion error and 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 to have significant position error and that a submap should be able to predict measurements as long as the robot remains within the submap.

Finding a good moment to segment is always easier in retrospect, so the segmentation heuristic looks ahead a few seconds in time, meaning that in implementation SegSLAM runs a few seconds behind the current time. The precise time window depends on the speed, sensors, and dead-reckoning capabilities of the robot because the SegSLAM position estimate is brought up to the current time by appending the dead-reckoned trajectory. The resulting algorithm can be summarized as follows:

Initialize The SegSLAM particles $s_0^{(m)}$, $m = 1 : \#_s$ component poses $x_0^{(m)}$ and submaps θ_i, $i = 1 : \#_s$ are initialized according to a desired initial distribution. In the tabula rasa SLAM case, all the particle poses start at the origin and all the particle submaps are empty, and the particle weights are set to $1/\#_s$.

Predict

Predict motion The dead-reckoned position innovation u_t is computed using the navigation sensors (odometry, heading, etc). A new position x_t is predicted for each particle by sampling from

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

If we assume that u_t has zero mean Gaussian noise with standard deviation σ_u, then we can compute x_t by sampling from the navigation noise model $r_t \sim N(0, \sigma_u)$:

$$x_t = x_{t-1} + u_t + r_t. \tag{2}$$

Under motion prediction alone, the particles will gradually disperse according to the navigation sensor error model, representing the gradual accumulation of dead-reckoning error.

Predict submap In addition to predicting the particle poses, SegSLAM also predicts their submaps by sampling from

$$p\left(\theta = \{\theta_{\text{same}}, \theta_{\text{new}}, \theta_{\text{match}}\} \middle| s_{t-1}^{(m)}, \Theta\right). \tag{3}$$

There are three possibilities for each particle's predicted submap: first, that the vehicle is still in the same submap region and should keep the current submap; second, that the vehicle has entered a completely new region and should start a new submap; and third, that the vehicle has reentered a previously mapped region and should switch to a copy of an old submap that has been stored in the SegMap Θ. These last two cases are called segmentation (new submap) and matching (old submap), respectively, and they also involve a transformation of the vehicle pose into the coordinate system of the target submap.

Weight The weight w for each particle is computed using the measurement model and the range measurement:

$$w_t^{(m)} = \eta\ p\left(z \middle| s_t^{(m)}\right) w_{t-1}^{(m)}, \tag{4}$$

where η is a constant normalizing factor that can be ignored because the weights are always normalized before being used to resample (next step). In our implementation, the range measurement z_t is compared to a ray-traced range $\hat{z}_t$ using the particle pose and submap. If we assume that the measurements have a Gaussian noise model with standard deviation σ_z, then

$$p(z|x, \Theta) = \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{\frac{-(\hat{z}-z)^2}{2\sigma_z^2}}. \tag{5}$$

If the vehicle collects many measurements simultaneously, its weight is the product of many such probabilities and can be computed using logarithms for numerical stability.

Resample The $O(\#_s)$ algorithm described in Arulampalam, Maskell, Gordon, and Clapp (2002) is used to resample the set of particles according to the weights $w_t^{(m)}$ according to

$$p\left(s_t^{(m)}\right) = \frac{w_t^{(m)}}{\sum_n^{\#_s} w_t^{(n)}}, \tag{6}$$

such that particles with low weights are likely to be discarded and particles with high weights are likely to be duplicated. Resampling may not be performed at every timestep: a rule of thumb introduced by Doucet et al. (2000) based on a metric by Liu (1996) is used to decide whether to resample based on the effective number of particles:

$$\#_{\text{eff}} = \frac{1}{\sum_{i=1}^{\#_s} (w^{(i)})^2}, \tag{7}$$

so that resampling is performed only when the effective number of particles $\#_{\text{eff}}$ falls below half the number of particles, $N/2$. When resampling is performed, the weights are reset to $1/\#_s$.

Update The measurement z is inserted into the current submaps according to the sensor model and the particle position. This is when submaps must be copied and updated; owing to our DRCO map data structure, copies are fast but updates are linear in the ray-casting operations. We avoid duplicate updates by updating the maps *before* copying successfully resampled particles.

Estimate A position estimate (for example, the mean) is computed from the particles, and then the estimate is brought up-to-date by appending the recent dead-reckoned trajectory. In cases in which there are multiple topological hypotheses, it may be impossible to come

Table I. SegSLAM notation.

Symbol	Description
$x_t^{(m)}$	Vehicle pose of the mth particle at time t
θ_i	Submap i, which includes an evidence grid map as well as t_s, t_e, the start and end times for the submap; T_s, T_e, the start and end transforms; j a reference to the submap's parent (may be none)
$s_t^{(m)}$	SegSLAM particle, $s_t^{(m)} = \{x_t^{(m)}, \theta_i\}$
Θ	The segmented map contains all the submaps, $\Theta = \{\theta_0, \theta_1, \ldots\}$
$w_t^{(m)}$	Weight of mth particle at time t
$\#_s$	Number of particles
u_t	Vehicle dead-reckoned innovation at time t
z_t	Range measurement at time t

up with a meaningful point estimate: the robot must still be able to plan and act in the face of this ambiguity.

Repeat from Predict

We now provide a derivation for this algorithm.

4.2. Derivation

A SLAM algorithm estimates the SLAM posterior, the probability distribution at time t over vehicle poses x_t and world maps Θ_t using all the sensor measurements $\mathbf{z}_t = \{z_1, z_2, \ldots, z_t\}$ and navigation updates $\mathbf{u}_t = \{u_1, u_2, \ldots, u_t\}$ (for a complete notation, see Table I):

$$p(x_t, \Theta_t | \mathbf{u}_t, \mathbf{z}_t). \tag{8}$$

Bayesian filtering provides a probabilistic framework for recursively estimating the state of such a dynamical system. The recursive Bayesian filter formulation of the SLAM problem is straightforward to derive (Montemerlo et al., 2002) but is usually computationally intractable to solve in closed form. Particle filters are a sequential Monte Carlo approximation to the recursive Bayesian filter that maintain a discrete approximation of the SLAM posterior using a (large) set of samples from the state space, or *particles*. With a large number of particles, the filter can represent arbitrary distributions, and so particle filters provide an implementation of Bayesian filtering for systems whose belief state, process noise, and sensor noise are modeled by nonparametric probability density functions (Arulampalam et al., 2002).

The difficulty in applying particle filters to SLAM is that the state space is very large because it includes both pose x and map Θ. The key insight of Murphy (1999) is that the SLAM posterior can be factored into two parts, or marginals: the trajectory distribution and the map distribution

$$p(x_t, \Theta_t | \mathbf{u}_t, \mathbf{z}_t) = p(x_t | \mathbf{u}_t, \mathbf{z}_t) p(\Theta_t | x_t, \mathbf{u}_t, \mathbf{z}_t). \tag{9}$$

Furthermore, knowing the vehicle's trajectory $\mathbf{x}_t = \{x_1, x_2, \ldots, x_t\}$ makes the observations $\mathbf{z}_t = \{z_1, z_2, \ldots, z_t\}$ conditionally independent (Montemerlo et al., 2002), so that the map distribution $p(\Theta_t | x_t, \mathbf{u}_t, \mathbf{z}_t)$ can be computed in a computationally efficient closed form (often a Kalman filter or an evidence grid) from the poses and measurements, dramatically reducing the dimensionality of the space that the particles must be sampled from. The process of factoring a distribution such that one part can be computed analytically is known as Rao–Blackwell factorization (Doucet et al., 2000).

We can express the Rao–Blackwellized SLAM posterior over *trajectories* as

$$p(\mathbf{x}_t, \Theta_t | \mathbf{u}_t, \mathbf{z}_t) = p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{z}_t) p(\Theta_t | \mathbf{x}_t, \mathbf{u}_t, \mathbf{z}_t). \tag{10}$$

Instead of storing the entire particle trajectory $\mathbf{x}_t$ and reconstructing the particle map at each time step, a Rao–Blackwellized particle $s_t^{(m)} = \{x_t^{(m)}, \Theta_t^{(m)}\}$ consists of two parts: the current particle pose $x_t^{(m)}$ and a recursively updatable particle map $\Theta_t^{(m)}$ that is kept up-to-date at each time step, effectively encoding the particle's trajectory over time.

The SLAM problem can be formulated as a Bayesian graphical model that exploits the conditional independence of measurements given the vehicle poses. As shown in Figure 4, the world can be spatially segmented into potentially overlapping regions that are conditionally independent given the relative transform and the map match, or overlap, between the regions. SegSLAM makes the assumption (aided by its choice of segmentation points) that this overlap can be ignored.

SegSLAM divides each particle trajectory $\mathbf{x}_t^{(m)}$ into temporal intervals: $\{\mathbf{x}_{[1]}^{(m)}, \mathbf{x}_{[2]}^{(m)}, \mathbf{x}_{[3]}^{(m)}\}$. These intervals may be created when the segmentation heuristic decides to create a new submap, or when the matching heuristic finds the

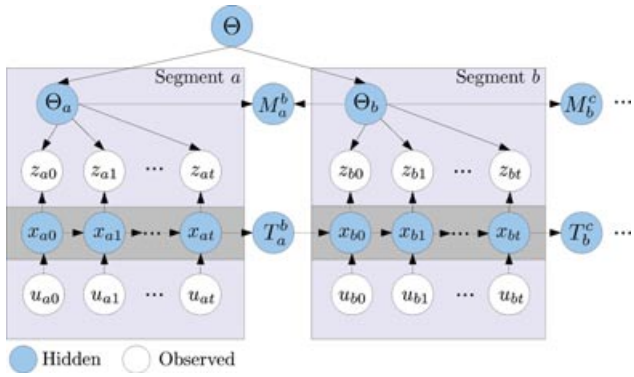

Figure 4. SegSLAM graphical model: Segments are related only by the relative transform T between their coordinate frames and the match M, or overlap, between them. For speed, SegSLAM ignores M, assuming independence between submaps.

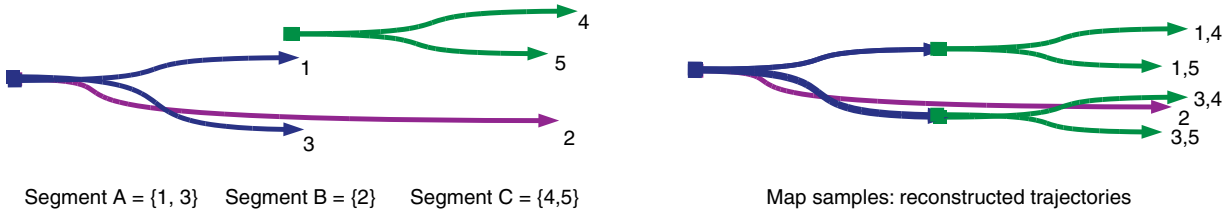

Figure 5. In this example, the particle trajectories are shown rather than the submaps that would be constructed from the trajectories. The particle filter has only three particles, and the left-hand figure shows that the particles segment halfway along the trajectory (in reality the two new trajectories start in a new coordinate frame). Because these two particles have the same start and end times, they are grouped together in Segment A, and likewise they are both in Segment C. SegSLAM can recover all of the trajectories shown on the right by map sampling.

particle has reentered a previous submap. Segmentation is like repeatedly restarting tabula rasa SLAM, except that all the particles do not necessarily have to restart at once (although the resampling process tends to narrow down the number of segmentation points that are represented by the set of particles). Each trajectory segment is in its own coordinate frame, so SegSLAM also maintains the transforms $T_{[i]}^{[j]\,(m)}$ between segments so that complete trajectories can be reconstructed.

SegSLAM decomposes the transform $T_{[i]}^{[j]\,(m)}$ into two pieces, $T_{[i]\rightarrow}^{(m)}$, the exit from submap θ_i, and $T^{\rightarrow[j]\,(m)}$, the entrance into submap θ_j. The exit $T_{[i]\rightarrow}^{(m)}$ encodes the particle's last position in submap θ_i and so depends only on interval i. In the case when submap j is a new submap (due to segmentation), then the entrance $T^{\rightarrow[j]\,(m)}$ is the identity matrix, because the particle starts at the origin of the reference frame of the new map. In the case of loop closures, however, a particle may reenter a previously constructed submap, in which case $T^{\rightarrow[j]\,(m)}$ arises from finding the registration, or match, into submap θ_j and transforming the particle pose accordingly. As will be described in more detail, SegSLAM finds this match using a short look-ahead time window that is not yet incorporated into submap θ_i. This means that $T^{\rightarrow[j]\,(m)}$ does not depend on interval i (or submap θ_i) but only on the previously constructed submap θ_j and the measurements in the look-ahead window.

Given the trajectory segments $\{\mathbf{x}_{[1]}^{(m)}, \mathbf{x}_{[2]}^{(m)}, \ldots\}$ and the corresponding transforms $\{T_{[i]\rightarrow}^{(m)}, T^{\rightarrow[j]\,(m)}, \ldots\}$, SegSLAM can reconstruct particle trajectories that are equivalent to the RBPF trajectories:

$$\mathbf{x}_t^{(m)} = \text{concatenate}\left(\mathbf{x}_{[1]}^{(m)}, T_{[1]\rightarrow}^{(m)} T^{\rightarrow[2]\,(m)} \mathbf{x}_{[2]}^{(m)}, \ldots\right). \quad (11)$$

Thus, in the same way that the set of RBPF particles are used as a discrete approximation to the SLAM posterior, the SegSLAM particles, after reconstruction, represent the same distribution. But in addition, there may be (and usually are) several particles, each with a slightly different trajectory, that share the exact same interval, starting and ending at the same time. We say that these particle trajectories share the same segmentation interval and are thus part of the same segment and are temporally compatible with other segments that do not temporally overlap with their interval. Submaps from compatible segments can be combined interchangeably to produce new trajectories. For example, if particle trajectories $\mathbf{x}_{[1]}^{(k)}$ and $\mathbf{x}_{[1]}^{(m)}$ are in segment 1 and particle trajectory $\mathbf{x}_{[2]}^{(n)}$ is in the compatible segment 2, then we can join $\mathbf{x}_{[1]}^{(k)}$ and $\mathbf{x}_{[2]}^{(n)}$:

$$\mathbf{x}_t^{(*)} = \text{concatenate}\left(\mathbf{x}_{[1]}^{(k)}, T_{[1]\rightarrow}^{(k)} T^{\rightarrow[2]\,(n)} \mathbf{x}_{[2]}^{(n)}, \ldots\right). \quad (12)$$

See Figure 5 for an illustration of this example.

After several segmentations, there is a very large number of possible reconstructed trajectories. Rather than exhaustively enumerating these trajectories, SegSLAM generates samples by growing random combinations of submaps and segment-to-segment transforms. SegSLAM can quickly generate a large number of these map samples so that its discrete estimate of the SLAM posterior is much better than that of an RBPF.

In the next two subsections, we describe our methods for segmentation and matching submaps.

4.3. Segmentation

Intuitively, segmentation should divide 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 should not 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 be plausible for a structured environment, such as a series of rooms connected by narrow doorways, but will obviously unravel in unstructured environments.

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 that lie within the contiguous region and unlikely to collect measurements in different regions. This

property is also called simultaneous visibility or overlap (Blanco, González, & Fernández-Madrigal, 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, because relatively few measurements span 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 such as doorways, rooms, or hallways.

To broaden the intuition, Estrada et al. (2005), Paz and Neira (2006), and Roman (2005) 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.

4.3.1. Motion Error Segmentation

A segment should have minimal internal position error. This is a straightforward proposition if we consider only the dead-reckoning error: a position error model that accounts for the uncertainty incurred by different maneuvers can be used to segment when the estimated 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:

$$\text{position}_{\text{err}} \propto (\alpha_1 v_{\text{err}} + \alpha_2 u_{\text{err}})t \tag{13}$$

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 frequently be worse for a wheeled vehicle at high accelerations due to wheel slippage:

$$\text{position}_{\text{err}} \propto [\alpha_1 v_{\text{err}}(x) + \alpha_2 u_{\text{err}}(x)]t. \tag{14}$$

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

Directly applying the dead-reckoning error model would result in near-periodic segmentation, which ignores the SLAM corrections to short-term dead-reckoning error. Although 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 (Fairfield, 2009). As a result, 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.

4.3.2. 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:

$$p(z_a|x_a, \theta_a) \gg p(z_a|x_a, \theta_b) \tag{15}$$

and

$$p(z_b|x_b, \theta_b) \gg p(z_b|x_b, \theta_a). \tag{16}$$

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

$$p(z'|x', \theta) \propto \text{predictiveScore}(z', x', \theta) = \prod_{n=1}^{\#_{z'}} p(z'^n|x', \theta). \tag{17}$$

To use the "future" measurements z', SegSLAM must be run a few seconds in the past, so that its current maps are a few seconds old. The future pose data x' are 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

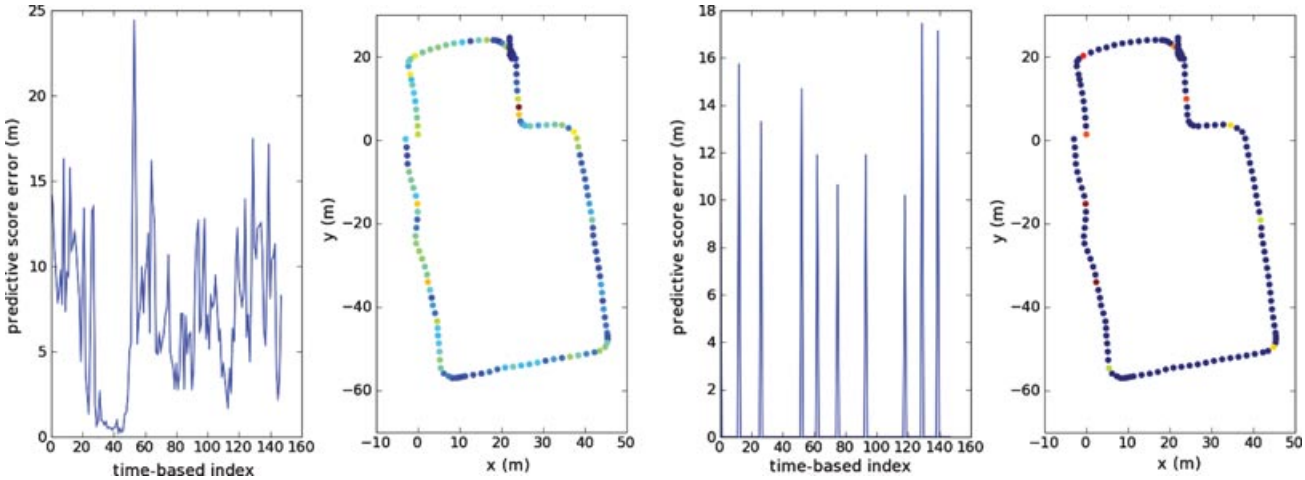

Figure 6. A portion of the Bruceton Mine, showing the predictive score as a plot and a scatter plot (left) and the resulting predictive score and segmentation points after a minimum segment size criterion has been enforced.

SegSLAM should segment. The predictive score heuristic depends on two parameters: the length of the predictive window and the segmentation threshold. See Figure 6.

Blanco et al. (2008) has a more exact segmentation method using graph cuts but then needs to reconstruct the maps, a slow procedure in two dimensions and an intractable one in three dimensions. We take the penalty of suboptimal segmentation in exchange for real-time speed.

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

4.4. Generating Local Metric Map Samples

Different combinations of temporally compatible segments from the SegMap can be stitched together to form a complete trajectory—this is like sampling from the distribution of all metric maps that are encoded in the SegMap (Figure 5). As with RBPF SLAM, it is computationally intensive to use the entire particle trajectories (or reconstructed trajectories) to assemble the maps at every timestep. So, like RBPF particles, SegSLAM particles $s_t^{(m)} = \{x_t^{(m)}, \theta_i^{(m)}\}$ instead store a pose and a reference to a submap. Submaps, in turn, store the interval (t_s, t_e), entry and exit transforms T_s, T_e, an octree-based evidence grid map, and a reference a parent submap (Figure 7). New submaps, created by segmentations, do not have parents and have empty evidence grids. But when a particle closes a loop and reenters a previous submap, the particle copies the previous submap evidence grid (a free operation for the DRCO map structure), sets the submap parent to the previous submap, updates the interval and transforms, and begins to update the evidence grid. This is efficient in storage because the DRCO copy-on-write map structure also has the concept of inheritance: a child map stores only modifications to the parent map. The parent map stores its own interval, transforms, and parent, so that by traversing up the parent links all the time intervals that were used to construct the map can be reconstructed, as well as the particle entrance and exit points for each of those intervals (Figure 7).

SegSLAM relies on the assumption that submaps do not significantly overlap each other: if they do, then the assumption of independence fails. The two operations of map segmentation and map matching are intended to minimize the degree to which the independence assumption is violated. Segmentation attempts to minimize overlap between sequential submaps. Matching overlapping maps together can recover most of the joint information and is essential for finding loops.

If there are loops (due to matches), then ties are broken by randomly selecting the order in which we expand segments with the same search depth, because choosing a segment excludes other segments that are temporally incompatible, and we want to generate a random sample from all such reconstructions. If many map samples are generated, they will break these ties differently. This is the root reason that SegSLAM is locally consistent but not globally consistent: the map sampling algorithm uses consistent transforms to expand segments, but when the breadth-first expansion must break a tie, SegSLAM does not attempt to globally optimize the transforms around loops. However, the breadth-first search does push global inconsistencies away from the current position, meaning that the local area is still consistent (Figure 7).

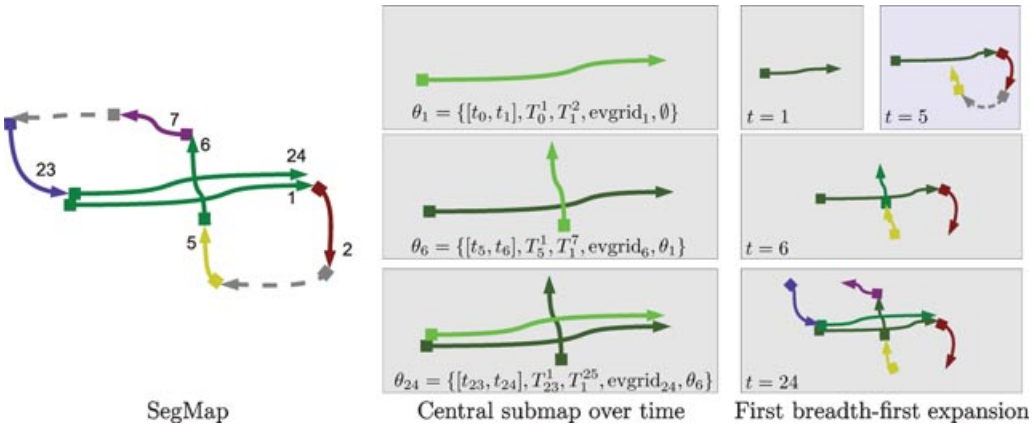

Figure 7. In this example, we consider the case of generating map samples by breadth-first expansion from the current submap. For simplicity, consider SegSLAM with a single particle. On the left is the segmented path of the vehicle around a figure-eight loop. Dashed arrows indicate a number of segments that have been omitted. The middle column shows the evolution of the central submap, as the particle reenters the submap multiple times. The right-hand column shows the first step of the breadth-first map sample generation starting from each illustrated submap. The top right-hand corner shows the full breadth-first expanded map sample at time step 5, just before SegSLAM closes the loop.

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

4.5. Matching

Matching can be thought of as loop closure, overlap detection, or map reentry. Fundamentally, it is the realization that the current environment matches a place that has been seen before. In Fairfield and Wettergreen (2009), we discussed methods for matching octree evidence maps together. In the preceding section, we described how to sample local metric maps from the SegMap. 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 are weighted according to the quality of the fit, and these weights are used to stochastically select from among the set of matches (which always includes the current segment, a null match).

4.5.1. Winnowing Match Candidates

We use a cascade of criteria to try to discard as many candidates as possible as quickly as possible. The first criterion, 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 criterion, spatial proximity, queries several voxels near the current robot position in the candidate map (using the candidate transform) to see whether 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 in a particular local metric map sample.

4.5.2. 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 Fairfield and Wettergreen (2009). 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 maps are matched using iterative closest point (ICP) on the octree-binned point clouds, which is a very fast method that reduces the influence of point density. 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:

$$w(T_m) \approx p(\text{err}_{\text{ICP}}|T_m)\ p\left(\tau_m^2\right). \tag{18}$$

To verify our map-based matching methods, we tested finding matches with particle filter localization. Because each transformation in the local map sampling process adds some uncertainty, we can estimate the uncertainty in

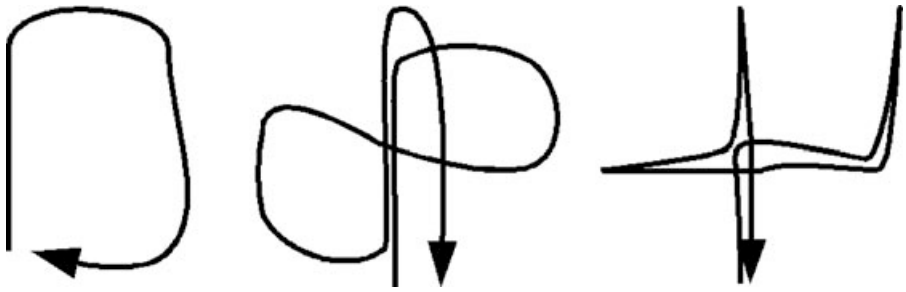

Figure 8. We investigate the application of different metrics to characterize SLAM performance in three broad topological classes: a single loop, multiple intersecting loops, and a star with multiple legs.

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. Recall that SegSLAM runs a few seconds in the past in order to effectively look ahead to find good segmentation points: this same time window of robot perceptions (motion and range measurements) is 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 measurement errors, yielding a strong indicator when the particle filter converges to a good solution:

$$w_{\text{loc}}(T_m) \approx p(\text{err}_X)\ p(|\Sigma_X|)\ p\left(\tau_m^2\right), \tag{19}$$

where $\text{err}_X = (z - \hat{z})^2$ is the particle measurement 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.

5. 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 and illustrate each method with an experiment with the Cave Crawler robot from different sites. The first method is to simply see whether the algorithm properly detects a loop closure, which is illustrated by a multilevel loop from a parking garage. The second method is to subjectively examine a large map with many loops, to see whether there are inconsistencies or obvious flaws; this is illustrated with a data set from a coal mine. The third method is to search for the minimum entropy map, and we illustrate this method with a data set from a building collected during its construction. These three data sets also correspond to different topological classes: a single loop, multiple intersecting loops, and a star of out and back legs (Figure 8).

5.1. Cave Crawler

Cave Crawler is an autonomous mobile robot that was designed to explore and map abandoned mines (Morris et al., 2006) (Figure 9). Cave Crawler uses a Crossbow 400 inertial measurement unit (IMU) and wheel odometry as position measurements, and the mapping sensors are forward- and backward-looking 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. An important distinction is that unlike many SLAM data sets, in which the vehicle comes to a complete stop to collect its 3D data, all the data sets presented here involve (almost) continuous movement. This significantly complicates the sensor calibration problem and makes it more difficult to estimate the yaw bias, the most significant source of error in dead reckoning.

5.2. Loop Closing Error: 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 Carnegie Mellon University campus is a convenient, multilevel structure with three exits on different levels, which allows Cave Crawler to traverse 3D loops. In the data set 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 10 and 11).

The metric used in this experiment for evaluating SLAM and SegSLAM performance is the loop closure error. In our regular RBPF, we can generate a position estimate from the particle cloud by taking a weighted average of the

Figure 9. The autonomous mine mapping robot Cave Crawler.

Figure 10. Photo of the two parking garage entrances used: The vehicle exited the upper entrance and entered via the lower entrance.

particle positions. This approach usually will not work with SegSLAM because the particles in different segments 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 performances of RBPF SLAM and SegSLAM, we ran each approach 20 times on the data set, 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 II, the dead-reckoning error of the original data is significant due to yaw bias (Figure 12), and the RBPF manages to close the loop with about 40 particles. But SegSLAM is able to reliably match very accurately even with just one particle. Whereas the matching does come at some computational cost for low numbers of particles, as shown by the run times in Table III, surprisingly for higher numbers of particles, SegSLAM is actually faster! Because SegSLAM adds the segmentation and matching steps to the RBPF, this may be explained by the fact that manipulating the segmented octree maps is faster than manipulating the full maps (although there is no difference in the octree depths, voxel dimensions, etc., between the maps).

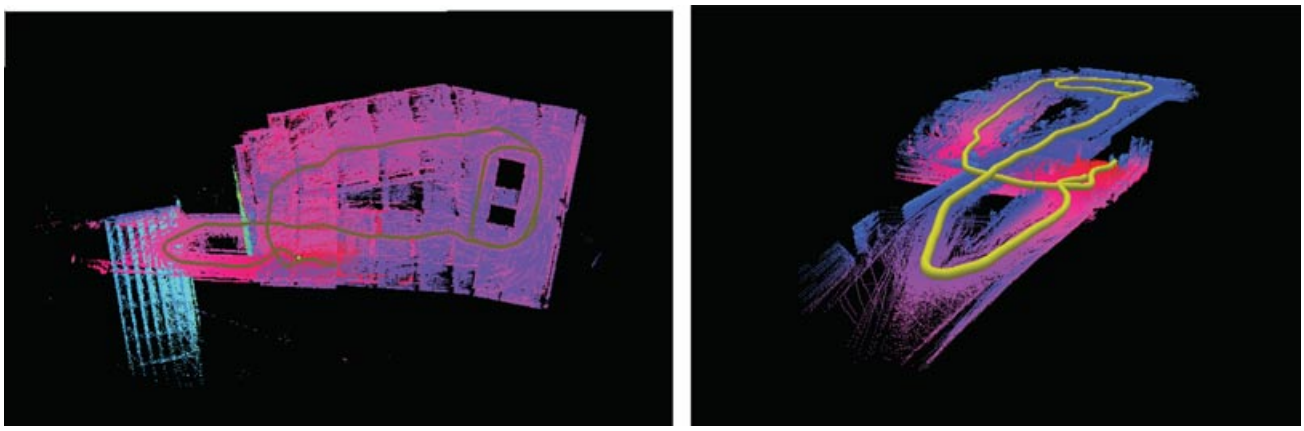

Figure 11. Left: A top isometric view of the parking garage data set, showing the vehicle path, which includes two loops on two different levels. The prominent structure on the left is a bridge outside the parking garage. Right: A perspective view of the parking garage entrances on two levels, showing the vehicle path.

5.3. Map Goodness: Coal Mine

Bruceton Mine is a research mine near Pittsburgh, Pennsylvania, and a common location for Cave Crawler tests. The data set used here was collected by the subterranean robotics team on May 14, 2007, and comprises a 1,300-m traverse through the mine, including several loops (Figure 13). This site has also been described by Thrun et al. (2003), but it is important to note that although most prior data sets consisted of stationary laser scans, the data used here were collected while the robotic vehicle was continuously moving.

A simple method for evaluating SLAM and SegSLAM performance is to look at the maps and judge their subjective "goodness." For a traditional RBPF, this is fairly simple, because even though there are many maps (one per particle), it is rare that the maps differ significantly except in the last few hundred meters, and so if one succeeds in building a "good" map, they all will succeed. For SegSLAM, the case is different because we can draw samples only from the SegMap. If only one sample 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 is only one 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 match between segments and not diverge.

RBPF SLAM never successfully closed all the loops; even with 1,000 particles (taking 2.8 h of computation) it simply could not deal with the high yaw error and the many interlocking loops, and even the best outcomes mistakenly merged two parallel tunnels (Figure 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 13) and was effectively doing feature detection in a sparse environment.

Table II. 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.

	μ/σ	
Particles	RBPF SLAM	SegSLAM
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 III. Run time in seconds for the parking garage loop data set.

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

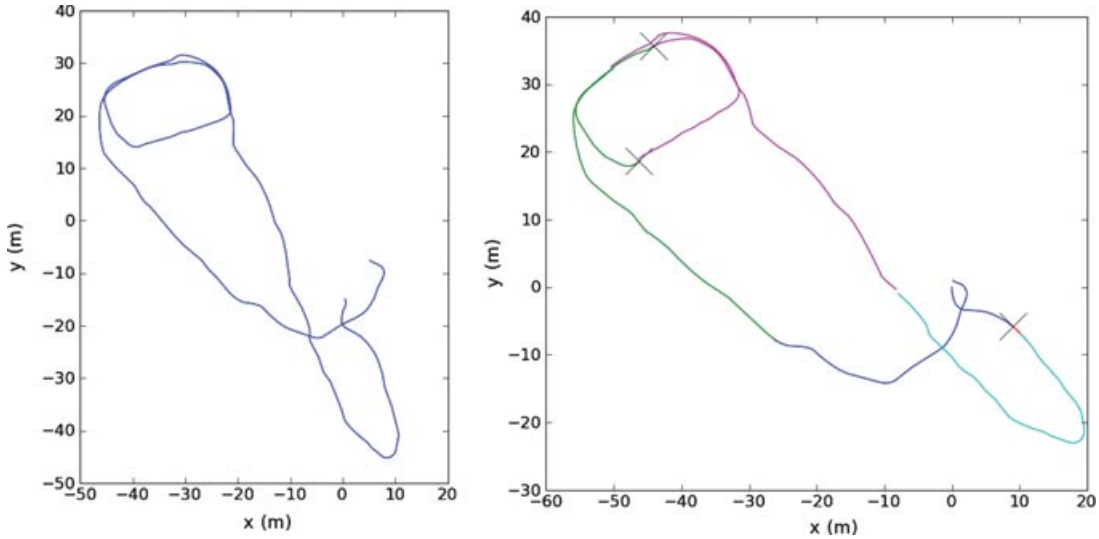

Figure 12. Left: Dead reckoning for the parking garage data set, showing the significant position error. Note that due to automobile traffic, the vehicle did not return exactly to its start position. Right: A reconstructed SegSLAM trajectory, showing how SegSLAM segmented the data set and correctly detected matches (indicated by Xs) and reentered previously mapped segments (as indicated by the segment coloring).

5.4. Minimum Entropy Map: Construction Site

Cave Crawler mapped out a portion of the Gates Building, traveling 892 m over three different levels. This data set was collected during the construction of the building, and although 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 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.

Each voxel $\theta_i[x, y, z]$ in the evidence grid map θ_i is a Bernoulli random variable that estimates whether the voxel

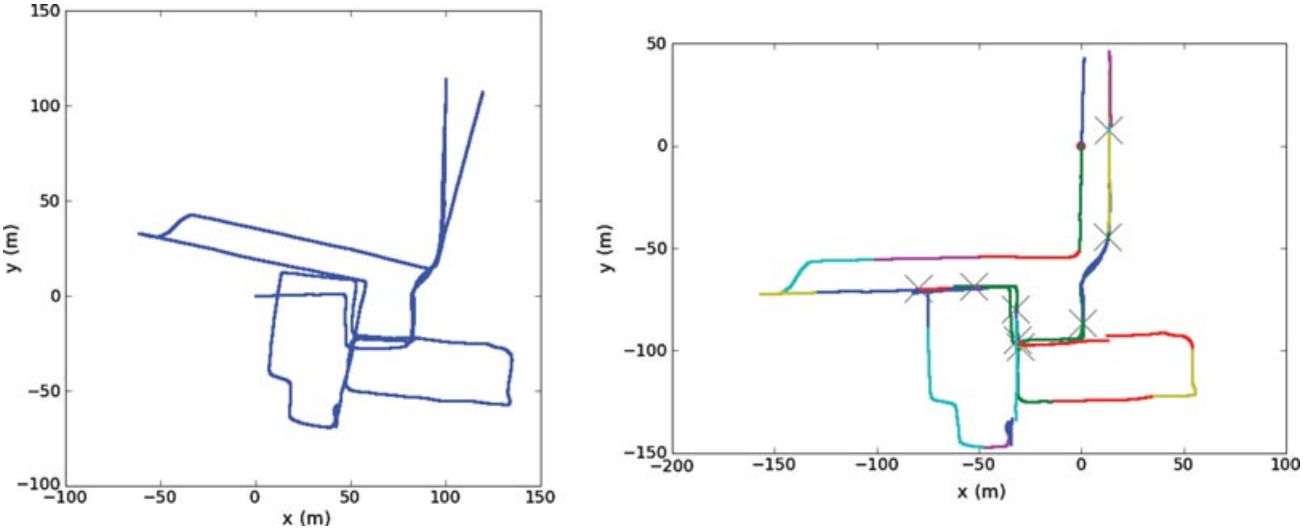

Figure 13. Comparison of the dead-reckoned path (left) with a sample SegSLAM path (right). Segments are color coded such that nearby lines with the same color indicate that a particle reentered a prior segment (a limited palette means that distant lines may share the same color as well). Xs mark matches/reentries.

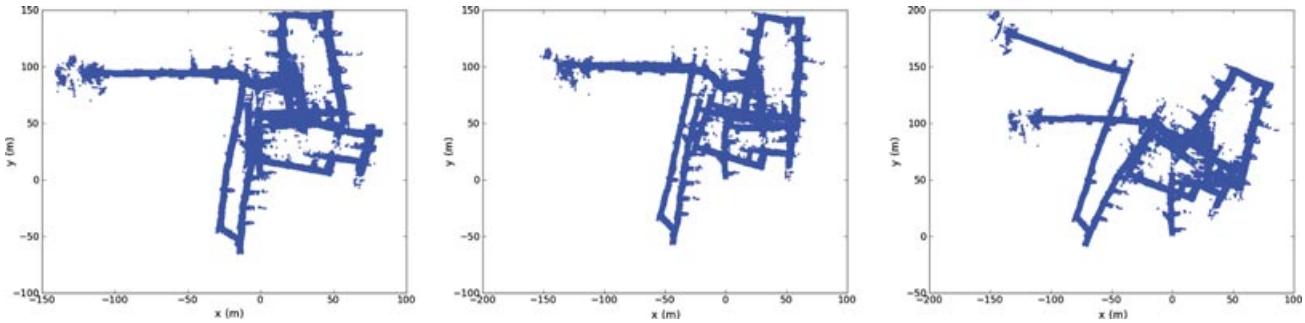

Figure 14. Example maps showing RBPF SLAM's failure on the Bruceton data set, with 200, 500, and 1,000 particles.

is empty or occupied. Thus the entropy of a voxel is

$$H(\theta[x, y, z]) = -\rho \log(\rho) - (1 - \rho) \log(1 - \rho), \tag{20}$$

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:

$$H(\theta) = \sum_{\forall x,y,z} H(\theta[x, y, z]). \tag{21}$$

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 will 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:

$$H(\theta) = \sum_{x,y,z \in \text{Obs}} \gamma H(\theta[x, y, z]). \tag{22}$$

In searching for the minimum entropy map, SegSLAM is at a distinct advantage over RBPF SLAM, because its segments encode an exponential number of possible maps, whereas RBPF SLAM can offer only one map per particle. In a sense, we are searching to see whether "good" maps have any support in the distribution over maps represented by the SegMap. But because generating and evaluating hundreds or thousands of map samples is computationally expensive, minimum entropy search is necessarily an offline operation.

We ran RBPF SLAM on the Gates data set with 20–800 particles and found that 200 particles, which ran in just under 3 h or about twice real time, was sufficient to generate consistent maps in which each leg of the star topology was aligned with itself and the SLAM position estimate accurately returned to the start position. However, because there were no real loops in the data set, each of the legs of the star had some error relative to the other legs, yielding a gradual misalignment between the levels of the building (Figures 15 and 16).

SegSLAM with 40 particles ran in 693 s and after an offline search for the minimum entropy map yielded the map shown in Figure 16. The SegSLAM map entropy was 10,755, compared with the minimum RBPF SLAM map entropy of 13,561. Without ground truth these entropy values can be considered only relatively, but they demonstrate that SegSLAM can produce significantly better global maps than RBPF SLAM while running 15 times faster, although the search for the best global map encoded in the SegMap must be performed offline.

6. CONCLUSION

We have presented a robust, real-time, submap-based 3D SLAM approach called SegSLAM that uses an extension to the particle filter prediction step to determine the particle submap: weighting, resampling, and updating are still applied as in standard RBPF SLAM.

We demonstrated SegSLAM with the Cave Crawler robot in several environments, including a mine, a parking garage, and a multilevel partially constructed building. We showed that it is faster and 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.

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 reenter those segments perform localization. This avoids the evidence erosion problem that degrades the long-term operation of RBPF SLAM with evidence grids. Second, as with most segmented SLAM approaches, submaps can be quickly merged by adding the evidence grids 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.

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

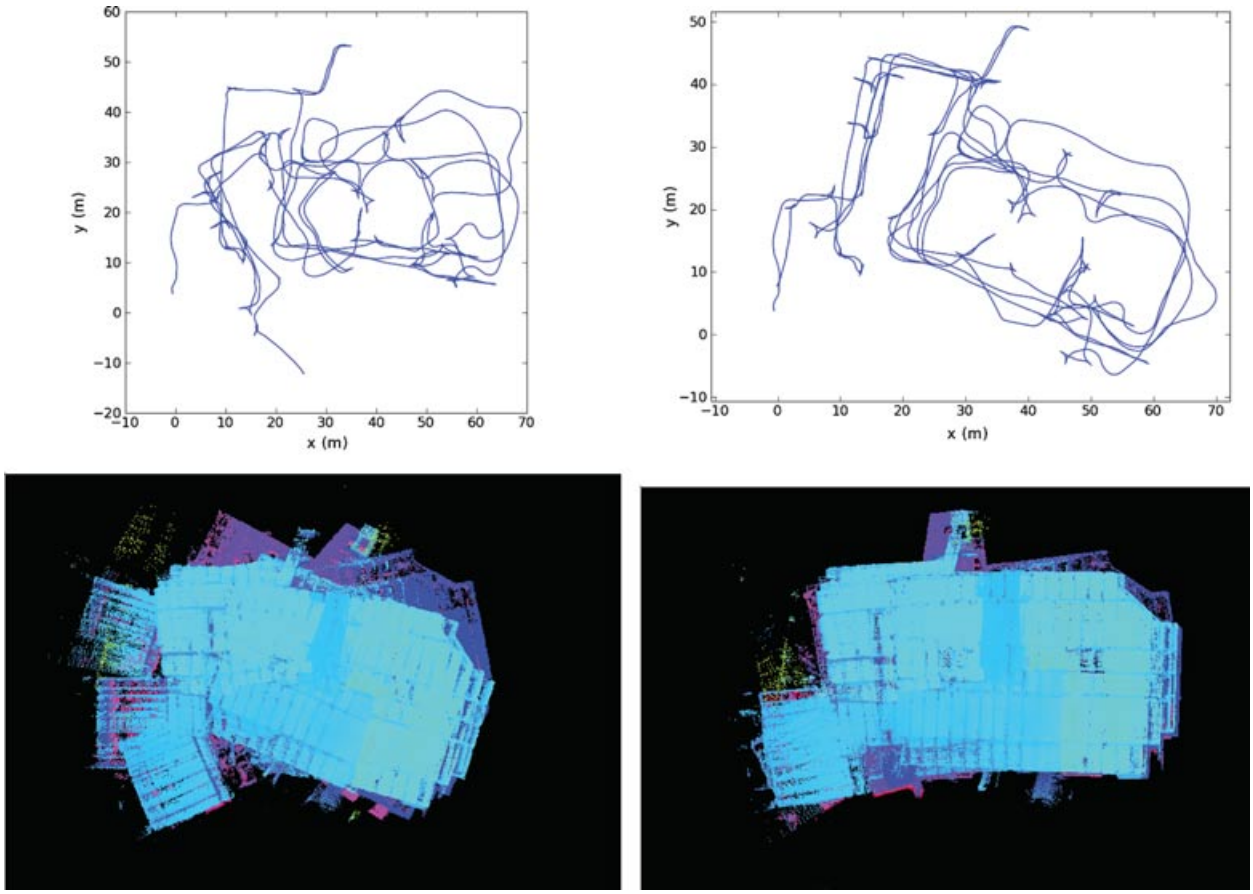

Figure 15. Raw dead reckoning (left) and calibrated dead reckoning using offline estimate of the heading bias (right) trajectories and point cloud for the Gates data set. Even with the optimal constant-heading bias, there is still obvious misalignment between the three levels of the building.

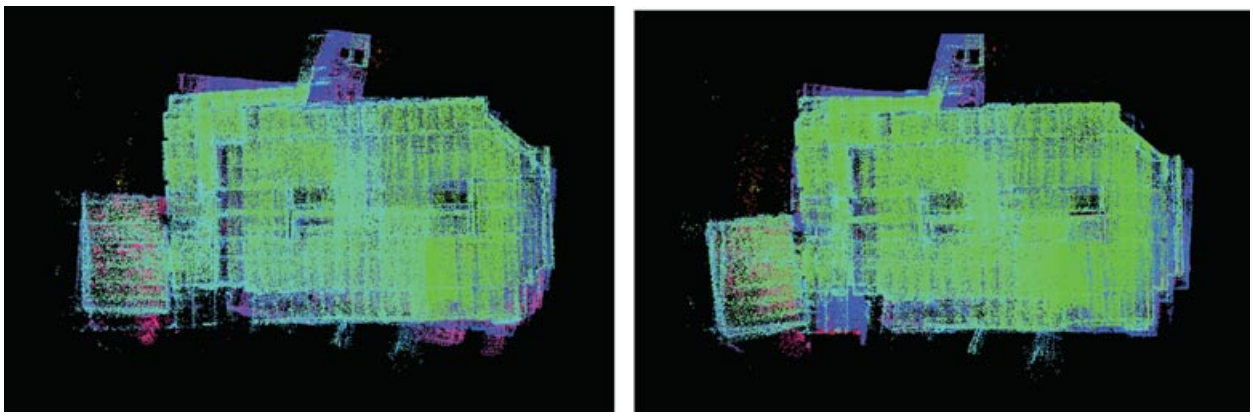

Figure 16. Left: The map from 200p RBPF SLAM, run time 3 h. Right: The best map from an offline entropy minimizing search of the SegMap after running SegSLAM with 40 particles, run time less than 12 min. Both maps show some misalignment between the levels of the building, but the SegSLAM map is distinctly better aligned, which is reflected in their respective map entropies.

was a deliberate trade-off that is suitable for applications that do not require a globally accurate metric map, because SegSLAM provides a locally consistent metric map that is sufficient for short-term planning, together with a global topological map that is useful for long-term planning. We are currently investigating offline methods for using the SegMap as a starting point for constructing a globally consistent metric map.

In future work, we would like to more clearly investigate the consistency and long-term performance of SegSLAM, particularly with regard to low-probability regions of the SegMap. There are several possible methods for pruning submaps or entire segments out of the SegMap, including finding and discarding nonviable segments and merging well-registered segments.

We would like to evaluate our general 3D methods on 2D data. We believe that SegSLAM's advantages, including the efficient octree-based map representation, real-time performance, support for multiple metric and topological hypotheses, and ability to close large loops, will apply to 2D data sets as well.

REFERENCES

Arulampalam, S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian Bayesian tracking. IEEE Transactions on Signal Processing, 50(2), 174–188.

Blanco, J.-L., Fernández-Madrigal, J.-A., & González, J. (2008). Towards a unified Bayesian approach to hybrid metric-topological SLAM. IEEE Transactions on Robotics, 24(2), 259–270.

Blanco, J.-L., González, J., & Fernández-Madrigal, J.-A. (2006, May). Consistent observation grouping for generating metric-topological maps that improves robot localization. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006), Orlando, FL.

Blanco, J.L., González, J., & Fernández-Madrigal, J.A. (2009). Subjective local maps for hybrid metric-topological SLAM. Robotics and Autonomous Systems, 57(1), 64–74.

Bosse, M., Newman, P., Leonard, J., & Teller, S. (2004). Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. International Journal of Robotics Research, 23(12), 1113–1139.

Bulata, H., & Devy, M. (1996, April). Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In International Conference Robotics and Automation, Minneapolis, MN (vol. 2, pp. 1054–1060).

Chong, K., & Kleeman, L. (1999). Feature-based mapping in real, large scale environments using an ultrasonic array. International Journal of Robotic Research, 18(1), 3–19.

Choset, H., & Nagatani, K. (2001). Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. IEEE Transactions on Robotics and Automation, 17(1), 125–137.

Doucet, A., de Freitas, N., Murphy, K., & Russell, S. (2000, July). Rao–Blackwellised particle filtering for dynamic Bayesian networks. In Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), Stanford, CA (pp. 176–183).

Eliazar, A., & Parr, R. (2006). Hierarchical linear/constant time SLAM using particle filters for dense maps. In Y. Weiss, B. Schölkopf, and J. Platt (Eds.), Advances in neural information processing systems 18 (pp. 339–346). Cambridge, MA: MIT Press.

Estrada, C., Neira, J., & Tardós, J. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. IEEE Transactions on Robotics, 21(4), 588–596.

Fairfield, N. (2009). Localization, mapping, and planning in 3D environments. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Fairfield, N., Kantor, G., Jonak, D., & Wettergreen, D. (2008). Depthx autonomy software: Design and field results (Tech. Rep. CMU-RI-TR-08-09). Pittsburgh, PA: Robotics Institute.

Fairfield, N., Kantor, G., & Wettergreen, D. (2007). Real-time SLAM with octree evidence grids for exploration in underwater tunnels. Journal of Field Robotics, 24, 3–21.

Fairfield, N., & Wettergreen, D. (2008). Active localization on the ocean floor with multibeam sonar. In Proceedings of MTS/IEEE OCEANS, Quebec City, Canada.

Fairfield, N., & Wettergreen, D. (2009, May). Evidence grid-based methods for 3D map matching. In International Conference Robotics and Automation, Kobe, Japan.

Frese, U. (2006). Treemap: An o(log n) algorithm for indoor simultaneous localization and mapping. Autonomous Robots, 21(2), 103–122.

Friedman, S., Pasula, H., & Fox, D. (2007, January). Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In IJCAI, Hyderabad, India (pp. 2109–2114).

Dudek, G., Freedman, P., & Hadjres, S. (1993, August). Using local information in a nonlocal way for mapping graph-like works. In International Joint Conference on Artificial Intelligence, Chambéry, France (pp. 1639–1645).

Grisetti, G., Stachniss, C., & Burgard, W. (2005, April). Improving grid-based SLAM with Rao–Blackwellized particle filters by adaptive proposals and selective resampling. In Proceedings of IEEE International Conference on Robotics and Automation, Barcelona, Spain (pp. 2443–2448).

Grisettio, G., Tipaldi, G.D., Stachniss, C., Burgard, W., & Nardi, D. (2007). Fast and accurate SLAM with Rao–Blackwellized particle filters. Robotics and Autonomous Systems, 55(1), 30–38.

Guivant, J., & Nebot, E. (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. IEEE Transactions on Robotics and Automation, 17(3), 242–257.

Havran, V. (1999). A summary of octree ray traversal algorithms. Ray Tracing News, 12(2).

Jefferies, M., Cosgrove, M., Baker, J., & Yeap, W. (2004, September). The correspondence problem in topological metric mapping—Using absolute metric maps to close cycles. In KES, Wellington, New Zealand (pp. 232–239).

Kortenkamp, D., & Weymouth, T. (1994). 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, pp. 979–984). Menlo Park, CA: American Association for Artificial Intelligence.

Kuipers, B., & Byun, Y. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Robotics and Autonomous Systems, 8, 47–63.

Leonard, J., & Feder, H. (2001). Decoupled stochastic mapping. IEEE Journal of Oceanic Engineering, 26(4), 561–571.

Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I., & Choset, H. (2003, October). Hierarchical simultaneous localization and mapping. In 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03) (vol. 1, pp. 448–453).

Liu, J. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statistics and Computing, 6(2), 113–119.

Modayil, J., Beeson, P., & Kuipers, B. (2004, September). Using the topological skeleton for scalable global metrical map-building. In IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan (pp. 1530–1536).

Montemerlo, M. (2003). FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002, July). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In Proceedings of the AAAI National Conference on Artificial Intelligence, Edmondton, Canada (pp. 593–598).

Morris, A., Ferguson, D., Omohundro, Z., Bradley, D., Silver, D., Baker, C., Thayer, S., Whittaker, W., & Whittaker, W. (2006). Recent developments in subterranean robotics. Journal of Field Robotics, 23(1), 35–57.

Murphy, K. (1999, November). Bayesian map learning in dynamic environments. In Neural Information Processing Systems, Denver, CO (pp. 1015–1021).

Newman, P., Leonard, J., & Rikoski, R. (2003, October). Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar. In Eleventh International Symposium of Robotics Research, Siena, Italy (pp. 409–420).

Paz, L., Jensfelt, P., Tardós, J., & Neira, J. (2007, April). EKF SLAM updates in O(n) with divide and conquer SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA.07), Rome, Italy.

Paz, L., & Neira, J. (2006, October). Optimal local map size for EKF-based SLAM. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China (pp. 5019–5025).

Ranganathan, A., & Dellaert, F. (2004, September). Inference in the space of topological maps: An MCMC-based approach. In Intelligent Robots and Systems, Sendai, Japan (vol. 2, pp. 1518–1523).

Remolina, E., & Kuipers, B. (2004). Towards a general theory of topological maps. Artificial Intelligence, 152(1), 47–104.

Roman, C. (2005). Self consistent bathymetric mapping from robotic vehicles in the deep ocean. Ph.D. thesis, Massachusetts Institute of Technology and Woods Hole Oceanographic Institution.

Savelli, F., & Kuipers, B. (2004, September). Loop-closing and planarity in topological map-building. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan.

Schultz, A., & Adams, W. (1998, May). Continuous localization using evidence grids. In Proceedings of International Conference on Robotics and Automation, Leuven, Belgium (vol. 4, pp. 2833–2839).

Simhon, S., & Dudek, G. (1998, October). A global topological map formed by local metric maps. In IROS, Victoria, Canada (vol. 3, pp. 1708–1714).

Stachniss, C. (2006). Exploration and mapping with mobile robots. Ph.D. thesis, University of Freiburg.

Tardós, J., Neira, J., Newman, P.M., & Leonard, J.J. (2002). Robust mapping and localization in indoor environments using sonar data. International Journal of Robotics Research, 21(4), 311–330.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence, 99(1), 21–71.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. Cambridge MA: The MIT Press.

Thrun, S., Haehnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., & Whittaker, W.L. (2003, September). A system for volumetric robotic mapping of abandoned mines. In Proceedings of IEEE International Conference on Robotics and Automation, Taipei, Taiwan.

Williams, S., Dissanayake, G., & Durrant-Whyte, H. (2002, May). An efficient approach to the simultaneous localisation and mapping problem. In IEEE International Conference on Robotics and Automation, Washington, DC (vol. 1, pp. 406–411).

Yamauchi, B., & Langley, P. (1996, May). Place learning in dynamic real-world environments. In Proceedings of RoboLearn 96, Key West, FL (pp. 123–129).