DP-SLAM 2.0

Austin I. Eliazar and Ronald Parr
Department of Computer Science
Duke University
Durham, North Carolina 27708
Email: {eliazar,parr}@cs.duke.edu

***Abstract*— Probabilistic approaches have proved very successful at addressing the basic problems of robot localization and mapping and they have shown great promise on the combined problem of simultaneous localization and mapping (SLAM). One approach to SLAM assumes relatively sparse, relatively unambiguous landmarks and builds a Kalman filter over landmark positions. Other approaches assume dense sensor data which individually are not very distinctive, such as those available from a laser range finder. In earlier work, we presented an algorithm called DP-SLAM, which provided a very accurate solution to the latter case by efficiently maintaining a joint distribution over robot maps and poses. The approach assumed an extremely accurate laser range finder and a deterministic environment. In this work we demonstrate an improved map representation and laser penetration model, an improvement in the asymptotic efficiency of the algorithm, and empirical results of loop closing on a high resolution map of a very challenging domain.**

I. Introduction

Probabilistic approaches have proved very successful at addressing the basic problem of localization using particle filters [1]. Expectation Maximization (EM) has been used successfully to address the problem of mapping [2] and Kalman filters [3] have shown great promise on the combined problem of simultaneous localization and mapping (SLAM).

The challenge of SLAM is that of producing accurate maps in real time, based on a single pass over the sensor data, without an off line correction phase. One approach to SLAM assumes relatively sparse, relatively unambiguous landmarks and builds a Kalman filter over landmark positions [4], [5], [3]. Other approaches assume dense sensor data which individually are not very distinctive, such as those available from a laser range finder [6]. An advantage of such approaches is that they are capable of producing detailed maps that can be used for path planning. One straightforward way to produce such maps would be to localize the robot based upon a partial map and then update the map based upon the most likely position of the robot. However, this tends to produce maps with errors that accumulate over time. When the robot completes a physical loop in the environment, serious misalignment errors can result.

In earlier work, we presented an algorithm called DP-SLAM [7], which provided a very accurate solution to the dense data case by efficiently maintaining a joint distribution over robot maps and poses using a particle filter. The ability to represent a joint distribution over poses and maps gives DP-SLAM an advantage over approaches that represent only a single map. This gives DP-SLAM the ability to resolve map ambiguities as a natural part of the particle filtering process and obviates the explicit loop closing phase needed for other approaches [6], [8].

Efficiently maintaining the joint distribution required some significant data structure engineering because maps are large objects and a naive particle filter implementation would entail huge amounts of block memory copying as map hypotheses progress through the particle filter. By maintaining a tree representation of where the paths of different particles diverged, we exploited redundancies between the maps. As originally implemented, we were able to achieve a runtime which was in the worst case log-quadratic in the number of particles used.

In environments that were consistent with our assumptions, DP-SLAM was able to manage hundreds of maps at real time speed, and close loops of over 50m, with no discernible errors. This was achieved without predetermined landmarks and with no explicit map correction operations. However, some of our assumptions were considerably stricter than we would like. Our maps were occupancy grids with cells that were presumed to be either fully transparent or fully opaque. Errors resulting from partially or transiently occupied squares were simply absorbed in an unrealistically large laser error model. As a result, the algorithm did not perform well in environments with finely grained clutter.

The contributions of this paper are as follows: (1) By using a more efficient method of accessing the grid squares, and performing a closer analysis of rest of the algorithm, we have improved the asymptotic analysis of the algorithm from log-quadratic in the number of particles to quadratic. (Real world performance is substantially better than this.); (2) We introduce uncertainty into the map representation of DP-SLAM through the use of a novel laser penetration model; (3) We present empirical results demonstrating a significant improvement over our previous algorithm along many dimensions, including the ability to close a large loop at 3 cm resolution despite some extremely challenging and ambiguous sensor readings.

II. Particle Filter and SLAM overview

A particle filter is a simulation-based method of tracking a system with partially observable state. We briefly review particle filters here, but refer the reader to excellent overviews of this topic [9] and its application to robotics [10] for a more complete discussion.

A particle filter maintains a weighted (and normalized) set of sampled states, $S = \{s_1 \dots s_m\}$, called *particles*. At

each step, upon observing an observation o (or vector of observations), the particle filter:

1) Samples m new states $S' = \{s'_1 \dots s'_m\}$ from S with replacement.
2) Propagates each new state through a Markovian transition model: $P(s''|s')$.
3) Weights each new state according to a Markovian observation model: $P(o|s'')$
4) Normalizes the weights for the new set of states

Particle filters are easy to implement and have been used to track multimodal distributions for many practical problems [9].

A particle filter is a natural approach to the localization problem, where the robot pose is the hidden state to be tracked. The state transition is the robot's movement and the observations are the robot's sensor readings, all of which are noisy and/or ambiguous. (Note that in many environments, even a highly accurate laser is not sufficient to resolve ambiguities.)

The change of the state over time is handled by a motion model. Usually, the motion indicated by the odometry is taken as the basis for the motion model, as it is a reliable measure of the amount that the wheels have turned. However, odometry is a notoriously inaccurate measure of actual robot motion, even in the best of environments. Motion models differ across robots and types of terrain, but generally consist of a shift, to account for systematic errors and Gaussian noise.

After simulation, we need to weight the particles based on the robot's current observations of the environment. The position described by each particle corresponds to a distinct point and orientation in a known map. If we assume that map grid squares are either fully opaque or fully transparent to the laser, and that laser measurements have normally distributed error, it is relatively straightforward to compute the probability of a laser scan. Given the model and pose, each sensor reading is correctly treated as an independent observation [11]. The total posterior for particle i is then $P_i = \prod_k P_{\mathcal{L}}(\delta(i,k)|s_i, m)$, where $P_{\mathcal{L}}$ is a normal distribution and $\delta(i,k)$ is the difference between the expected and perceived distances for sensor (laser cast) k and particle i given map m.

Some approaches for SLAM using particle filters with dense observations attempt to maintain a single map with multiple robot poses [8], an approach that we avoided with DP-SLAM because it leads to errors that accumulate over time necessitating potentially costly off-line corrections to the map.

III. Algorithm and Analysis

DP-SLAM works by maintaining a joint distribution over robot poses and maps via a particle filter. Each particle corresponds to a distinct hypothesis about the map and the robot's position and orientation within the map. Our maps are grids with M squares. If the particle filter maintains P particles, a naive implementation of a particle filter that maintains this distribution would require $O(MP)$ work per iteration. By contrast, DP-SLAM exploits redundancy between the different maps to avoid copying maps in the resampling phase of the particle filters. DP-SLAM performs a semantically equivalent computation to map copying with less effort. In previous work (DPSLAM 1.0) [7] we provided a bound of $O(ADP \log P)$ work per iteration, where A is the number of grid squares swept out by the laser, and D is a term related to the uncertainty in the environment. In the worst case, D approaches $O(P)$, but in practice it is often considerably less. In fact, this bound can be reduced even further, as will be shown later.

DP-SLAM maintains an *ancestry tree* of all of the particles. This tree contains all of the current particles as leaves. The parent of a given node represents the particle of the previous iteration from which that particle was resampled. For DP-SLAM 1.0, we described how this tree could be maintained by adding and deleting nodes in $O(AP \log P)$ time, ensuring that ancestor particles (those from a previous iteration) which have no bearing on the current generation of particles are removed. Another important aspect of maintaining the tree is to collapse non-branching nodes of the tree, a process which we previously analyzed as having an (amortized) upper bound of $O(ADP \log P)$. In the next section we present an improved analysis of the same process, which tightens the bound to $O(AP \log P)$.

Instead of explicitly maintaining multiple maps, DP-SLAM maintains only a single global grid. This grid can be viewed as a means of indexing the map updates made by the particles. Instead of storing a single observation at each grid square, we store the set of all observations made to that grid square by all particles in our ancestry tree. This is done by assigning each particle a unique ID and maintaining a balanced tree, which is indexed on particle IDs, at each grid square. We call this an *observation tree*. The total number of observations in the observation tree for any node is bounded by the number of nodes in the ancestry tree. Note that the global grid stores all of the information necessary to reconstruct the map that is consistent with any active particle. To retrieve a single particle's observations for a particular square, we use the ancestry tree to determine the particle's exact lineage. By comparing this list of the particle's ancestors to the set of observations, we can extract the most recent observation relevant to this particle. Figure 1 provides a pseudocode overview of the different steps of the DP-SLAM 2.0 algorithm.

A. New Analysis

Since accessing the map for a given particle uses its lineage, the efficiency of the localization step is directly affected by the size of the ancestry tree. Accordingly, we would like to keep the ancestry tree minimal. The first and most obvious step to minimizing ancestry tree size is to prune away recursively those nodes which have no children, i.e., those which have no descendents in the current generation of particles. After the pruning phase, we collapse those parent nodes which only have one child by merging such parent and child nodes. When this process is complete, every branch in the tree will correspond to a different set of updates to the map. It is easy to verify that an ancestry tree with P particles at the leaves can have depth no more than P and have no more than P interior nodes.

DPSLAM 2.0 main loop

Resample particles with replacement: $O(P)$

Prune ancestry tree: $O(AP \log P)$
- Recursively remove childless nodes: $O(P)$
- Merge only children w/ parents: $O(AP \log P)$ (amortized)
  - For each only child
    - Remove child from observation tree
    - Insert child's observations w/parent's ID

Sample new poses based upon motion model: $O(P)$

Insert new particles into ancestry tree: $O(P)$

Compute new weights (Localization): $O(AP^2)$
- For each particle
  - For each laser cast
    - For each square hit
      - Lookup occupancy: $O(P)$

Normalize particle weights: $O(P)$

Update each particle's map: $O(APlogP)$
- For each particle
  - For each laser cast
    - For each square hit
      - Update d_τ and h: $O(logP)$

Fig. 1. The DPSLAM 2.0 algorithm.

Each ancestor node needs to maintain a list of those grid squares to which it has made updates. While these lists are not used for queries, they are important for removing the node's entries to the grid, should it ever be pruned. The amortized analysis of this process was previously stated as $O(ADP \log P)$. We show now that this cost is in fact $O(AP \log P)$.

To merge two nodes of the ancestry tree, we must go through the list of observations attributed to the child, and delete each observation from the corresponding grid square. Identical observations are then reinserted into the grid square, with a new key of the parent node's ID number. Since the observation tree is balanced, we can perform each deletion and insertion in $O(\log P)$ time. Thus, each merge has cost logarithmic in P and linear in the total number of observations involved.

We now provide an amortized bound on the total number of observation merges that iteration i of the particle filter can introduce. Consider the set of all observations inserted to the grid at this iteration. Each current particle makes $O(A)$ updates, each of which is recorded in the corresponding node in the ancestry tree. As all of these particles are from the current iteration, these updates are recorded on the leaves of the tree only.

Consider the A updates associated with a specific particle at iteration i. For these updates to move up the ancestry tree in a merge operation, all siblings of this particle must have first been removed from the tree. This implies that any node in the tree can absorb at most A observations from iteration i. As noted earlier, the ancestry tree is minimal; it has exactly P leaves and at most P interior nodes. Since no interior node can absorb observations from two different particles from iteration i, the total number of observation merges resulting from the observations made at iteration i is bounded by AP. Each observation merge costs $O(\log P)$, for a total amortized cost of $O(AP \log P)$.

B. Improved Search Algorithm

In isolation, the improved tree maintenance analysis of the previous section would not help the overall asymptotic complexity of DP-SLAM, since the $O(ADP \log P)$ localization cost of DP-SLAM 1.0 would dominate. To reduce the complexity of the algorithm as a whole, we introduce a new method for querying the map for each particle.

For localization, it is necessary for each particle to consider its entire set of observations, and compare them to the map it has built up to this point. Therefore, we must make $O(AP)$ accesses to the grid. The remaining $O(D \log P)$ term in the complexity of the DP-SLAM 1.0 algorithm came from the time needed to access the grid. Suppose the ancestry tree has depth D. (In the worst case, D can be as large as P, but it is typically much less in practice.) Previously, for each grid square, the particle performed a query for each of its D ancestors to determine if any of these ancestors had made an observation for this square. Each query has cost $O(\log P)$, for a total cost of $O(ADP \log P)$.

This time complexity is based on each ancestor of the particle performing an independent search. Consider what would happen if we instead performed a batch search for all of the ancestors together. First, we would need to sort the ancestry for each particle by their ID numbers. If we generate particle IDs sequentially, this is trivial since children always have larger IDs than their parents. We can then perform a binary search in the ancestry for the ID at the root of the observation tree. If the search fails, this still partitions the ancestry and we can then recursively search the children of the observation tree root using using half of the ancestry tree for each child.

In the worst case for analysis, we will split the ancestry list almost evenly at each level of the observation tree and incur cost logarithmic in the range of ancestors that is searched until we reach reach the level $\log D$ of the observation tree. At level i of the observation tree, we will have performed 2^i binary searches on $O(\frac{D}{2^i})$ elements of the ancestry. The total work for the first $\log D$ levels of the tree is

$$\sum_{i=0}^{\log D} 2^i \log(\frac{D}{2^i}) = \sum_{i=0}^{\log D} 2^i(\log(D) - \log(2^i)) = O(D).$$

After the first $\log D$ levels of the tree, the number of elements from our ancestry list at each node in the tree is necessarily down to one. For the remainder of the observation tree, we perform $O(D)$ separate searches, for a cost of $O(D \log(\frac{P}{D}))$. Therefore, the total complexity for localization can be reduced

to $O(AP(D + D\log(\frac{P}{D})))$. In the worst case, where D approaches P, the complexity is $O(AP^2)$.

IV. Probabilistic Maps

The DP-SLAM 1.0 implementation assumed that grid squares are either totally occupied or totally empty. Conditioned on the robot position, there is only one possible square in which a laser cast of a given length can terminate. We added new observations to the map whenever the following two conditions were satisfied: 1) The cast did not pass through an existing obstruction in the map and 2) The cast did not terminate within 3 grid squares of an existing obstruction along the ray cast by the laser. Once an obstruction is entered in the map, it could not be removed or updated. We now describe a probabilistic model for the laser's penetration through space and how this model is updated.

There are three main sources of uncertainty that we would like to accommodate in our model. The first has to do with small objects, and irregular surfaces. Regardless of the resolution of our grid, there will always be objects which do not fit well within a grid square. Rough surfaces, such as rocks and plants, as well as small items like fences and bushes are very difficult to represent explicitly in a deterministic fashion. Another issue arises from the discretization of the world. Most surfaces will not align well with the given grid, by only partially occupying certain squares, and will give differing readings depending on which portion of the grid square is scanned. This can be particularly problematic for surfaces which are nearly parallel to the current scan. Lastly, there are objects in the world that behave in manners too complex to easily model, such as moving people, or surfaces which can occasionally reflect the sensor. We would like a method which can recover gracefully from these aberrant events, instead of placing permanent errors in the map.

The idea of using probabilistic map representations is possibly as old as the topic robotic mapping itself [12]. Many of the earliest SLAM methods employed probabilistic occupancy grids, which were especially useful for sonar sensors prone to noisy and/or spurious measurements. However, by concentrating on a more precise sensor, such as a laser range finder, and the behavior of our own algorithm, we present a more appropriate method for representing uncertainty in the map, which takes into account the distance the laser travels through each grid square.

A. Map Representation and Observation Model

One important goal of our model is the idea that the probability of laser penetration should depend on the distance traveled through a grid square. Earlier approaches to estimating the the total probability of the scan would trace the scan through the map, and weigh the measurement error associated with each potential obstacle by the probability that the scan has actually reached the obstacle [1]. In an occupancy grid with partial occupancy, each cell is a potential obstacle.

Consider Figure 2, where we can see two possible scans of equal length, with different orientations. Note that the axis

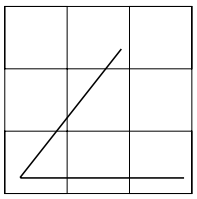

Fig. 2. Effect of angle on number of grid cells penetrated.

parallel scan passes through three grid squares, while the diagonal scan passes through 4. (In general, the number of squares visited can differ by a factor of $\sqrt{2}$.) Without a method of weighting the visited grid squares based on the distance the laser has traveled through them, two scans of equal length traveling through similar grid squares can have vastly different probabilities (since they represent a different number of possible laser obstacles) based solely on the orientation of the map. Notice also that some of the squares that the angled scan passes through are only clipped at the corner. Even if these squares are known to interrupt laser scans with high probability, the effects of these squares on the the total probability of the scan should be discounted. This is largely due to the fact that the sensor only detects the boundaries of objects, and thus map squares which are likely to be occupied are almost always only partially occupied. These effects can cause a localization algorithm to prefer to align some scans along axes for spurious reasons.

Our second goal in developing a laser penetration model was that the the model should be *consistent*. We derive our notion of consistency from the following thought experiment: Consider a laser cast that passes through two adjacent squares of identical composition and travels the same distance through each square. The probability that the laser beam will be interrupted should be the same as if the laser had traveled twice the distance through a single, larger square of identical composition to the two smaller ones. More generally, consistency suggests that our level of discretization should not directly affect the probability that a laser cast of given a length will be interrupted.

If we define, $P_c(x, \rho)$ to be the cumulative probability that the laser will have been interrupted after traveling distance x through a medium of type ρ, our consistency condition can be stated more generally in terms of k divisions as,

$$P_c(x, \rho) = P_c(x/k, \rho) \sum_{i=1}^{k} (1 - P_c(x/k, \rho))^{(i-1)}.$$

The exponential distribution, $P_c(x, \rho) = 1 - e^{-x/\rho}$, for a scalar ρ, satisfies this condition. We will refer to ρ as the *opacity* of grid square.

For any laser cast, we can express the interaction between the model and the trace of the cast up to some point n as a vector of distances traveled through grid squares $\mathbf{x} = (x_1 \dots x_n)$ and the corresponding opacities $\boldsymbol{\rho} = (\rho_1 \dots \rho_n)$ of those squares. As noted earlier, the distances will not be uniform. The cumulative probability that the laser cast is

interrupted by squares up to and including n is therefore,

$$P_c(\mathbf{x}, \boldsymbol{\rho}\ ) = \sum_{i=1}^{n} P_c(x_i, \rho_i) \prod_{j=1}^{i-1} (1 - P_c(x_j, \rho_j)).$$

We express the probability that the laser cast will be interrupted at grid square j as $P(\text{stop} = j)$, which is computed as the probability that the laser has reached square $j-1$ and then stopped at j,

$$P(\text{stop} = j) = P_c(x_j, \rho_j)(1 - P_c(\mathbf{x}_{1:j-1}, \boldsymbol{\rho}_{\ 1:j-1})),$$

where $\mathbf{x}_{1:j-1}$ and $\boldsymbol{\rho}_{\ 1:j-1}$ have the natural interpretation as fragments of the $\mathbf{x}$ and $\boldsymbol{\rho}$ vectors.

Suppose the vector $\boldsymbol{\delta} = (\delta_1 \ldots \delta_n)$ is a vector of differences such that δ_i is the distance between the laser distance measurement and grid square i along the trace of the laser cast. We express the conditional probability of the measurement given that the laser beam that was interrupted in square i as $P_{\mathcal{L}}(\delta_i | \text{stop} = i)$, for which we make the typical assumption of normally distributed measurement noise. The probability of the measurement is then the sum, over all grid squares in the range of the laser, of the product of the conditional probability of the measurement given that the beam has stopped, and the probability that the beam stopped in each square,

$$\sum_{i=1}^{n} P_{\mathcal{L}}(\delta_i | \text{stop} = i) P(\text{stop} = i).$$

B. Map Updates

The mean of the exponential distribution with opacity ρ is simply ρ, which makes updating our map particularly simple. For each square, we maintain what is essentially a laser odometer that sums the total distance d_τ traveled by laser scans through the square. We also keep track of the number of times that the laser is presumed to have stopped in the square, h. Our estimate of ρ is therefore $\hat{\rho} = d_\tau / h$. In our initial implementation of DP-SLAM 2.0 we treat the laser as a reliable measurement when updating the map. Thus, stop counts and grid odometers are updated under the assumption that the reported laser depth is correct. We realize that this remains somewhat contradictory to our observation model and we plan to implement soft updates, which will bring our approach closer to an incremental version of EM, in future work.

The cost of updating odometer values and stop counts has no effect on the asymptotic complexity of the algorithm. New particles are already required to check previous opacity values for localization purposes. Each particle then inserts a new set of observations, containing new odometer values and hit counts, into the observation trees in the global grid. In practice, however, DP-SLAM 2.0 makes more map updates than DP-SLAM 1.0. The original algorithm would not insert new observations if they simply repeated observations of parent particles. In DP-SLAM 2.0, repeated observations of the same grid square imply changes in the opacity for the square and require updates.

Grid squares which have not been observed by any ancestor of a particle under consideration are treated in a special manner. The difficulty with such squares is that it is difficult to estimate the probability of the laser stopping without previous experience. Therefore, we attempt first to explain a given observation as best we can using the existing map entries and then attribute any remaining probability mass to the previously unobserved grid squares in the most forgiving manner possible.

Algorithmically this means that we initially treat the unknown grid squares as having $P_c(x, \rho) = 0$. The line trace is continued past the observed endpoint to a distance of six standard deviations of the laser noise model. This allows us to be certain that any reasonable point in the laser's trajectory is allowed as a possible source of the observation. In cases where the laser has traveled through previously unobserved grid squares, the total computed probability that the laser would be stopped, given the trajectory and the map, may be less than one. In this case, the unknown area of the map is given the benefit of the doubt; any remaining probability is attributed to the previously unobserved grid square in the laser's trajectory which is closest to the observed endpoint. Thus any probability mass which is unable to be explained by our previous observations is assumed to come from an object in an unexplored area of the map of unknown occupancy.

This proposed observation model assumes that all of the error in the observations are normally distributed. Since each observation is independent, a single poor observation can have a drastic impact on the total probability of a given pose. However, in reality, there exist many other causes for disagreements between the observations and the map. Reflections from specular surfaces, non-static objects, and discretization errors are but a few sources which can give rise to highly erratic readings. Fortunately, these are all fairly low probability events, and do not need to be handled specifically by the observation model. However, their impact on the total probability should be limited. Therefore, a certain amount of "background noise" is allowed in the observations. Any single observation has a lower limit imposed on its probability, which can be interpreted as the probability of an unmodeled event causing an erratic reading. In practice, this background noise level was set to be relatively low compared to the normal observation model, less than 0.5%.

V. Empirical Results

We tested our algorithm on sensor logs generated by our iRobot ATRV Jr. in a cyclic hallway environment, with observations made approximately every 15cm. The robot is equipped with a SICK laser range finder, which scans at a height of 7cm from the floor. The size of this environment is roughly 24m x 60m, with the robot completing a loop approximately 12m x 40m. Figure 3 shows a map produced with a resolution of 3cm per grid square, using 10,000 particles.

The robot begins the run at the very bottom left corner of the map, and travels down the long hallway on the bottom, before turning up at the right end of the map and returning along the top hallway. The loop is finally closed near the top left of the

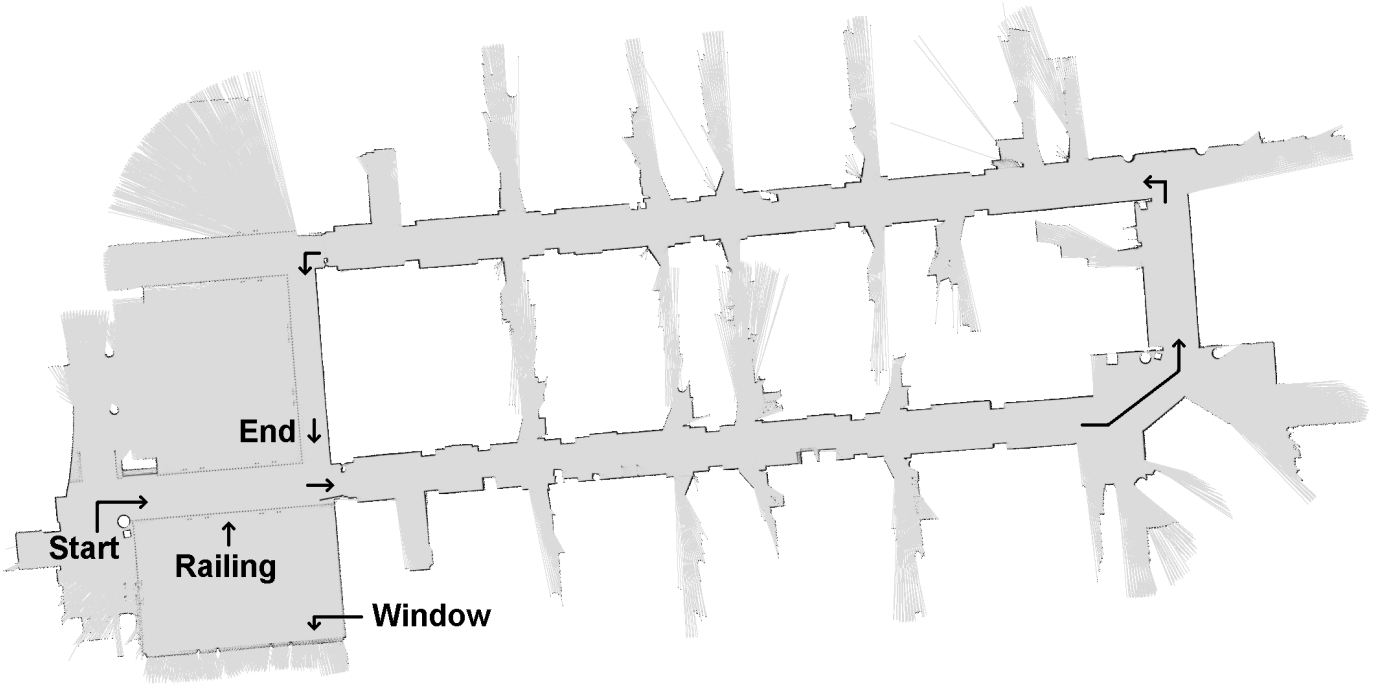

Fig. 3. A challenging domain at 3cm resolution. We strongly encourage readers to view a soft version of this document or (better) to download a full size map from our web page to appreciate the fine detail in this map.

Fig. 4. Robot perspective on the catwalk and railing, taken close to the *Railing* label in Figure 3. Slight changes in the robot position will affect which balusters are hit by the laser range finder, and which are missed.

map. We emphasize that our algorithm knows *nothing* about loops and makes no explicit effort to correct map errors. The extraordinary precision and seamless nature of our maps arises solely from the robustness of maintaining a joint distribution over robot positions and maps.

Details about the map, the environment, and the robot's trajectory give insight into the robustness and accuracy of the algorithm. The area in which the robot starts is on a raised catwalk, with a railing supported by many thin balusters (Figure 4), which at our scanning height appear as a series of small, evenly spaced obstacles. (The individual balusters are visible when the map is zoomed in.) These small balusters provide a very difficult challenge for both localization and mapping. Another challenge is a large set of windows along the bottom edge of the map, on the left side. The glass is a semi-opaque surface to the laser, only occasionally stopping the laser, due to dirt and angle of incidence. On close inspection, this area of the map may appear blurry, with some possible line doubling. This is actually because the window is double paned, and the laser has a chance of being stopped by either pane.

Other features of note are the intervening openings, which occur between the two long stretches of hall. When the robot moves from left to right along the bottom hallway, it sees only the right walls of these passages, and it sees the left walls on the return trip. Therefore, these passages provide no clues that would make it any easier for the robot to close the large loop in the map.

The loop is closed on the top left of the map on a catwalk parallel to the first. The accuracy in this map is high enough to maintain the correct number of balusters for the hand rail between the two catwalks. There is one section of map that may appear to be inaccurate, at the top right hand corner of the map. Here, there are two intersecting hallways, which meet at a slightly acute angle. This is enough of a disparity to make one end of the two "parallel" hallways in our map approximately 20cm closer together on the left end when compared to the right end. We were (pleasantly) surprised to discover, upon measuring the corresponding areas in the real world, that there was in fact a disparity of approximately 20cm in the

real building. Our algorithm had detected an anomaly in the construction.

We performed several other experiments on the same data log. Our first test involved the same algorithm with a grid resolution of 5cm. With the same parameters, and an equal number of particles, the loop still successfully closed. For comparison purposes, we also ran DP-SLAM 1.0 with the deterministic map approach. While the old method was able to do fairly well on the closed section of the hallways, the railing and windows caused difficulties from which it was unable to recover. Our last experiment involved a more simplistic approach to representing occupancy probabilities, which merely used a ratio of the number times the laser had stopped in the square compared to the number of times that it was observed. This method, while still able to handle some of the uncertainty in the environment, had significant skew and misalignment errors upon closing the loop. Additional experiments, annotated maps, pictures, and sensor logs are available at `http://www.cs.duke.edu/~parr/dpslam/`.

A reader knowledgeable about particle filters may be startled by the large number of particles we are using. The primary reason for this is that we are obliged to use a very wide proposal distribution because of extremely unreliable odometry readings. Thus, most of the particles we generate are bad ones that have no chance of being resampled at the next iteration of the particle filter. We exploit this fact by discarding bad particles before their weights are fully evaluated, using a method we described as *particle culling* in the DP-SLAM 1.0 paper. In practice, only a fraction of the stated numbers of particles is fully evaluated and the number kept is typically two orders of magnitude less. In practical terms, narrow corridors can be handled with 3000 particles at near real time speed on a fast PC. In more challenging environments like the one described here, particle evaluation is more time consuming because it requires longer line traces and more particles are required due to greater ambiguity. The map shown here required approximately 24 hours of CPU time. Raw speed was not our primary objective in mapping this challenging environment. If it were, we believe there are many areas, such as improving our proposal distribution, where dramatic speed increases could be obtained.

VI. Conclusion and Future Work

In this paper, we have extended the DP-SLAM algorithm to include probabilistic map occupancy, using a precise method which accounts for the distance that the laser has traveled through each grid square. These improvements were successfully tested on a much larger and noisier environment than we had attempted with our earlier algorithm.

Furthermore, we have revisited the algorithmic complexity of our method, and through a combination of improved analysis and a more efficient localization method, we have reduced the complexity from a worst case of log-quadratic in the number of particles, to merely quadratic.

There are several promising future directions for this research. We plan to test the DP-SLAM framework with noisier sensors and three dimensional maps. We also plan to explore the extent to which DP-SLAM-like methods can be combined with different map representations.

Acknowledgment

The authors gratefully acknowledge the support of the National Science Foundation and SAIC for this research. We also thank Dieter Fox for many helpful suggestions and comments.

References

[1] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[2] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, "Sonar-based mapping with mobile robots using EM," in *Proc. of the International Conference on Machine Learning*, 1999.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *IJCAI-03*. Morgan Kaufmann: 1151–1156, 2003.

[4] P. Cheeseman, P. Smith, and M. Self, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*. Springer-Verlag, 1990, pp. 167–193.

[5] J. H. Leonard, , and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," in *IEEE Transactions on Robotics and Automation*. IEEE, June 1991, pp. 376–382.

[6] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," 2000.

[7] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simulataneous localizatoin and mapping without predetermined landmarks," in *Proc. of the 18^{th} Int'l Joint Conf. on Artificial Intelligence (IJCAI-03)*. Acupulco: Morgan Kaufmann, 2003.

[8] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[9] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Berlin: Springer-Verlag, 2001.

[10] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[11] K. Murphy, "Bayesian map learning in dynamic environments," in *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[12] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *1985 IEEE International Conference on Robotics and Automation*. St. Louis, Missouri: IEEE Computer Society Press, Mar. 1985, pp. 116–121.