SLAM using Incremental Probabilistic PCA and Dimensionality Reduction

Emma Brunskill and Nicholas Roy
CSAIL, Massachusetts Institute of Technology
The Stata Centre, 32 Vassar St.
Cambridge, MA 02139
{*emmab*|*nickroy*}*@mit.edu*

***Abstract*— The recent progress in robot mapping (or SLAM) algorithms has focused on estimating either point features (such as landmarks) or grid-based representations. Both of these representations generally scale with the size of the environment, not the complexity of the environment. Many thousand parameters may be required even when the structure of the environment can be represented using a few geometric primitives with many fewer parameters. We describe a novel SLAM model called IPSLAM; our algorithm clusters sensor data into line segments using the Probabilistic PCA algorithm, which provides a data likelihood model that can be used within a SLAM algorithm for the simultaneous estimation of map and robot pose parameters. Unlike previous work in extracting line-based representations from point-based maps, IPSLAM builds non-point-based maps directly from the sensor data. We demonstrate our algorithm on mapping part of the MIT Stata Centre.**

***Index Terms*— Mapping, Mobile Robotics, PCA, Clustering**

I. Introduction

The recent progress in robot mapping (Simultaneous Localization and Mapping, or SLAM) algorithms has largely focused on building maps that either consist of point features (such as landmarks) or grid-based representations. Both of these representations are extremely dense; the number of points or grid cells required to represent some environments can be in the tens of thousands. The complexity of representation is unfortunately independent of the structure of the environment. Even when the structure of the environment can be represented using a few geometric primitives with a handful of parameters, the corresponding occupancy grid or feature set can still be large and complex.

One of the difficulties with building maps of geometric primitives, however, is that the statistical inference problem is difficult. In particular, the techniques for extracting higher-order geometric representations from data do not give an explicit probability model for the observed data. Geometric representations can be used to reduce the dimensionality of the point-based map representations after the map is completed, by for example using principal components analysis (PCA) to extract lines from data. However these algorithms work best at extracting map features from data that contains mostly random noise such as sensor error. When the data has not been corrected to compensate for systematic bias, such as odometry drift, dimensionality reduction algorithms are unable to extract the latent structure themselves. If we wish to build low-dimensional, geometric representations of the environment using SLAM, we need to be able to solve the chicken-and-egg problem of estimating the geometric representation of the map *and* the robot position. Reducing the map to a set of geometric primitives that do not provide a likelihood model of the data will not give us robust SLAM algorithms.

An additional motivation to building a geometric map in an online manner (as opposed to post-processing a point-feature map) is to facilitate motion planning. Many planning algorithms scale with the number of features in the environment (eg. [3]); by generating low-dimensional maps online, fast motion planning can result without completing the map. Additionally, by estimating the uncertainty of the geometric features online, exploration algorithms designed to maximize information gain [10] can be used to improve the map.

Our goal in this paper is to describe a SLAM algorithm called IPSLAM that creates a map composed of low-dimensional geometric primitives online, directly from the sensor data. We combine a dimensionality reduction technique that provides a well-formed probability model, called Probabilistic Principal Component Analysis [14], in conjunction with an online mapping algorithm called FastSLAM [8] to generate maps of the environment based on line segments. Our contribution is not a novel SLAM technique, but to show that existing SLAM techniques can be applied to a novel representation. The number of line segments is dynamically updated according to the sensor data. We demonstrate this algorithm on a navigating mobile robot in MIT's Stata Centre.

II. The SLAM Problem

The SLAM problem is typically phrased as estimating the parameters of the robot's position while simultaneously estimating the parameters of the map. Let us denote the robot's pose at time t as $\mathbf{x}_t$, and the parameters that describe the world (location of features, free space, etc.) as $\Theta = (\theta_1, \theta_2, \ldots, \theta_n)$ where each θ_i corresponds to the parameters of some feature, such as the position of a landmark. As the robot receives a control u_t at time t, its position changes probabilistically according to some transition distribution:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, u_t). \tag{1}$$

After each motion the robot receives sensor measurements $\mathbf{z}_t$. These observations are generated according to a distribution that depends on the robot position and the world state:

$$p(\mathbf{z}_t|\mathbf{x}_t, \Theta). \tag{2}$$

Since we do not know the robot position or the map, the SLAM problem is that of estimating the joint distribution over the robot and map state, $p(\mathbf{x}_t, \Theta)$, as controls are given

and sensor measurements are received. The Bayes' filter is most commonly used to perform this estimation:

$$p(\mathbf{x}_{t+1}|u_t) = \int_{\mathbf{X}_t} p(\mathbf{x}_{t+1}|\mathbf{x}_t, u_t)p(\mathbf{x}_t)d\mathbf{x}_t \quad (3)$$

$$p(\mathbf{x}_{t+1}, \Theta|\mathbf{z}_t) = \frac{p(\mathbf{z}_t|\mathbf{x}_t, \Theta)p(\mathbf{x}_t, \Theta)}{p(\mathbf{z}_t)}. \quad (4)$$

Equations 3 and 4 are updated after each control and each sensor measurement, to maintain a posterior estimate $p(\mathbf{x}_t, \Theta|\mathbf{z}_t, u_t)$.

The choice of map representation, or the form for $p(\mathbf{x}_t, \Theta)$, is a key factor in any SLAM algorithm. In mapping based on grids, the distribution is typically factored into a distribution $p(\mathbf{x}_t)$ and a multinomial $p(\Theta)$, both of which are jointly optimized offline. In Kalman filter-based SLAM, the joint distribution is represented as a multivariate Gaussian with some mean and covariance. The Kalman filter approach is popular because it allows for very efficient updates in worlds with a small number of parameters and can be extended to efficient updates in large worlds with appropriate decomposition [2]. However, because a motion-sensor update is $O(n^2)$ (where n is the total number of parameters in $\mathbf{x}$ and Θ), the estimation problem becomes progressively more computationally challenging as the size and complexity of the world increases.

This is one of the primary difficulties in SLAM—the problem of representational complexity. As the size of the world increases, so can the complexity of the estimation problem, even if the structure of the world itself does not increase significantly in complexity. Estimating a map that consists of a long corridor should not increase quadratically in the length of the corridor, but building a point-based map requires tracking point features down the entire length, which increases the number of features as the corridor gets longer. The recent arrival of constant-time SLAM algorithms such as Atlas [2] or the Sparse Extended Information Filter [12] mean that the statistical inference procedure itself need not scale at all with the number of features, but the complexity of the representation still does. This motivates a potential for low-dimensional geometric representations to be useful.

III. Dimensionality Reduction

Given a large number of point features that all correspond to the same environmental structure, a reasonable approach might be to use a dimensionality-reduction technique to extract higher-order, lower-dimensional representations that still capture the data. One such representation is Principal Components Analysis, that computes a set of basis functions that can be linearly combined to represent a collection of data; PCA has been used extensively in the vision community to find low-dimensional representations of very high-dimensional images, such as faces [15]. If we apply PCA to appropriately clustered sets of point features in a map, we can extract line segments to represent each cluster of point features, generating very efficient representations, e.g., a wall containing 1,000 point features in a 2-d world could be represented using just the 4 parameters of the end-points of 2-d line segment.

Although this approach has been applied successfully in a number of applications to convert point-feature-based models into line-segment based models (e.g. [6], [7], [13]), the problem is that current SLAM approaches (building the initial map) still require coping with the representational complexity of the point-features during the map-building process. What we would like is to build a model of the environment directly from line segments.

Unfortunately, most existing dimensionality reduction techniques such as PCA or line fitting do not contain the probabilistic models of the data necessary for robust SLAM. PCA can be viewed in probabilistic terms as finding a low-dimensional manifold that maximizes the likelihood of the data, but it does not provide a well-formed probability distribution over the manifold itself. Without a likelihood model of the data, we cannot compute $p(\mathbf{z}_t|\mathbf{x}_t, \Theta)$ as in equation 4.

A. Probabilistic PCA

The Probabilistic PCA algorithm developed by Tipping and Bishop [14] does provide a way to compute a low-dimensional representation with a well-formed probability distribution of higher dimensional data. Let us consider the problem of fitting some data $\mathbf{Z}$ of dimension d to some lower-dimensional parameterization ζ of dimension q. If this lower-dimensional parameterization of $\mathbf{Z}$ is linear, then we want some projection matrix $\mathbf{W}$ and offset μ so that $\mathbf{Z} = \mathbf{W}\zeta + \mu$, where μ is the mean offset of the data. However, for data that only approximates a line (because of measurement noise, for instance), no such $\mathbf{W}$ exists. The best we can do is minimize the error in representation, typically choosing to minimize squared-error such as in conventional PCA so that

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} ||\mathbf{Z} - \mathbf{W}\zeta + \mu||^2. \quad (5)$$

Instead let us explicitly represent the noise ϵ present in the observations:

$$\mathbf{Z} = \mathbf{W}\zeta + \mu + \epsilon \quad (6)$$

If we assume isotropic noise $\epsilon \sim N(0, \sigma^2\mathbf{I})$ and a normal distribution of the low dimensional parameterization ζ, we can compute the marginal distribution of $\mathbf{Z}$:

$$p(\zeta) = \mathcal{N}(0, 1) = (2\pi)^{-q/2} \exp\left\{-\frac{1}{2}\zeta_i^T\zeta_i\right\} \quad (7)$$

$$p(\mathbf{Z}) = \int p(\mathbf{Z}|\zeta)p(\zeta)d\zeta \quad (8)$$

$$\Rightarrow p(\mathbf{Z}) \sim N(\mu, C) \quad (9)$$

where

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \quad (10)$$

Equation 8 follows from the total law of probability, and equation 9 follows from combining equations 8, 6 and 7. Equation 9 defines a probability model of the high dimensional observations and which means the likelihood of any observation $\mathbf{z}$ can be directly computed given the model and $\mathbf{C}$. Tipping and Bishop [14] show that the maximum likelihood estimates of the parameters $\mathbf{W}_{ML}$, σ^2 and μ occur when μ is equal to the mean of the observations, and

$$\mathbf{W}_{ML} = \mathbf{U}(\Lambda - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \quad (11)$$

where $\mathbf{R}$ is an arbitrary rotation matrix and $\mathbf{U}$ and Λ are the principal eigenvectors and eigenvalues respectively of the observation covariance matrix,

$$S = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \mu)(\mathbf{z}_i - \mu)^T, \tag{12}$$

for N observations in $\mathbf{Z}$. At the maximum likelihood point (where $\mathbf{W} = \mathbf{W_{ML}}$) the maximum likelihood estimate for σ^2 is

$$\sigma^2_{ML} = \frac{1}{d-q}\sum_{j=q+1}^{d}\lambda_j \tag{13}$$

where λ_j is the jth eigenvalue of the observation covariance matrix (eigenvalues are ordered by descending magnitude). In the case of computing the 1-D parameterization of 2-D data, σ^2_{ML} is just the eigenvalue of the non-principal component, or in other words, the variance perpendicular to the principal component of the observations.

Notice that we can recover conventional PCA by maximizing

$$\lim_{\sigma^2 \to 0} \mathcal{L}(p(\mathbf{Z})), \tag{14}$$

that is, the likelihood of the data as σ^2 goes to 0. This would allow us to recover $\mathbf{W}$ from the data (and a low-dimensional representation), but would not provide a probability model of the data.

There are two key properties to this model that are beneficial to the problem at hand. First we now have an explicit probability model of the data, $p(\mathbf{Z})$, allowing us to compute the likelihood of any observation. Second, this probability model also can be used to compute the parameters for a mixture of such models using *Expectation-Maximization* as described by Tipping and Bishop [14]. Therefore, given a known set of features, this method can be used to compute the set of 1-D principle components which maximizes the likelihood of the data. For example, once the number of walls n is estimated, probabilistic principal component analysis (PPCA) can be used to compute the set of n 1-D vectors that maximize the likelihood of the underlying 2-D laser range data points.

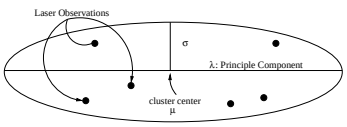


Fig. 1. Probabilistic Principal Components Analysis.

Two immediate issues arise when using this approach in an incremental fashion to do SLAM. The first is how to determine the number of features (or number of mixtures) over which to perform PPCA, since this is typically not known in advance The second is how to do PPCA incrementally so that as new data arrives, the parameters of the old features are updated accordingly.

B. Determing the Number of Low Level Features

The problem of estimating the number of line segments present in the environment, or the number of mixtures to be used in PPCA, is very similar to the problem of determining K in the K-means algorithm. Without prior knowledge of the domain, it can be very difficult to pre-select the correct number of clusters, yet important, since both K-means and mixtures of $PPCA$ perform poorly with bad choices of the K or the number of mixtures. To remedy this problem, Pelleg and Moore [9] created X-means, an algorithm which determines the best K (out of a range of Ks) number of clusters for a dataset. Their algorithm uses the Baysian Information Criteria to determine when to split a cluster into two. Hammerly and Elkan [4] improved upon their result in an algorithm G-means which uses the Anderson-Darling (AD) statistic to split clusters until the data allocated to each cluster passes the hypothesis of being normally distributed ($A^2(Q) > 1.035$, which corresponds to the probability the distribution is Gaussian is $> 99\%$). This result avoids the X-means assumption of the data being spherically distributed about a center.

In our approach we integrate G-means with PPCA to produce an algorithm which determines online the best number of mixtures and the best parameters of those mixtures to fit the data at hand. The parameters of the model are further improved by excluding outlier points (those points whose probability under the cluster that maximizes their likelihood is less than a given threshold, typically 1%) and recomputing the clusters' parameters.

C. Incremental Probabilistic PCA

In online SLAM, a robot wanders through a new environment. At each time step, the robot receives some new observations and integrates these observations into a map and estimate of its trajectory. Probabilistic PCA, in combination with G-means, produces a mixture of K low dimensional models when given a set of observations. To compute maps incrementally, a simple solution would be to store all observations, and at each new time step add in the new observations and re-run PPCA+G-means on the entire dataset. However, this approach is undesirable for two reasons. First, it becomes intractable to store all raw observations if the robot is on a prolonged period of exploration. More importantly, it is computationally expensive to run PPCA+G-means on the entire observation set at each time step.

Instead, we propose to maintain only a compact representation of the incremental map using the principal components and variances obtained from PPCA. At each new time step, a new set of observations are received. These new observations are processed by PPCA+G-means to produce a set of K' new clusters or principal components that succinctly represent the new observations. These K' clusters are then merged with the K clusters of the prior map representing the state of the world up through the previous time step. There are two main components to the merge procedure: the cluster correspondance problem (determining the mapping between old clusters and new clusters) and the cluster parameters update problem (how to merge two associated clusters).

D. Cluster Correspondence

Cluster association is determined by calculating a metric between every pair $(k_i \in K, k'_j \in K')$ pair of clusters. For each pair of clusters, first the dot product is computed between the normalized principal component vectors of the two clusters. If the dot product is below certain threshold

(in our experiments, a difference in angle of 60° was used, chosen experimentally) then the distance measure is set to $+\infty$, to indicate that these two clusters should not be merged. Otherwise, if the cluster centers are within the sum of the principal variances of the two clusters, another metric is computed. Alternatively, the distance between the cluster centers is used as a metric.

Cluster matching is done greedily by sequentially matching the next pair of clusters that have the smallest distance metric up to some distance threshold.

E. Cluster Merging

Once cluster association has been performed, the map parameters must be updated. If an old cluster has no match, its parameters are left unchanged. If a new cluster has no match, it is added to the new map only if it has the support of a sufficient number of observations, and its non-principal component variance is low. The motivation here is to add in new features to the map only if they are well defined (represent 1-D features well, as represented by low non-principal component variance (see figure 2 and note that if σ^2 is small the model becomes line-like)) and help explain a good number of the new observations.

For cluster pairs (k', k), their parameters must be merged to compute a new cluster. This is done using a variant of Weng et. al. [16]'s Incremental Principal Component Analysis (IPCA) algorithm. IPCA provides a way to iteratively re-estimate the principal components of a data set without requiring the storage of all data. Let z_m be the mth d-dimensional observation and $A = \sum_{i=1}^{m}(z_i - \mu)(z_m - \mu)^T$ be the dxd sample covariance matrix. Then letting x_i be the ith normalized estimate of an eigenvector of A (and λ the corresponding eigenvalue) yields

$$Ax_i = \sum_{i=1}^{m}(z_i - \mu)(z_i - \mu)^T x_i = \lambda x_i \equiv v_m \qquad (15)$$

Note that by maintaining v_m the current estimate of the eigenvalue and eigenvector can be computed by $\lambda = ||v||$ and $x_i = v/||v||$. By initially setting v_0 to the principal component of the first observation set, recursive estimation of v_m is possible.

$$v_m = \frac{m-1}{m}v_{m-1} + \frac{1}{m}(z_m - \mu)(z_m - \mu)^T \frac{v_{m-1}}{||v_{m-1}||} \qquad (16)$$

This has the intuitive effect of pulling the old estimate of the principal component towards the new observations.

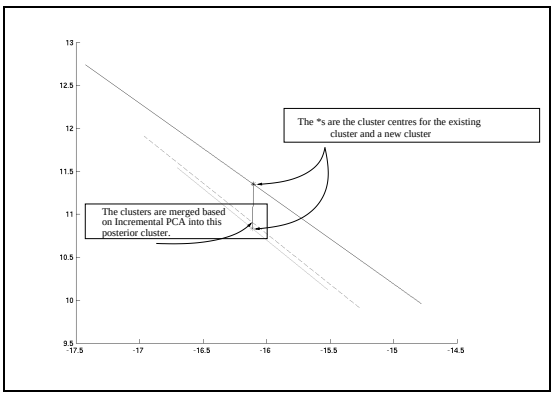


Fig. 2. The process of incorporating a new cluster into the existing map.

This procedure allows the computation of merged principal components from an old and new cluster. The merged cluster center is computed by taking a weighted average of the old and new cluster centers. See figure 2 for an illustration of this procedure.

IV. Fast SLAM

So far we have described how to incrementally compute a set of clusters or line segments that represent a map of the environment. We now explain how the line segment estimation is used to produce a complete SLAM model.

A. Robot Pose Estimation

Robot pose estimation is computed using particle filtering in a manner similar to Thrun's FastSLAM [12]. A collection of particles $\mathbf{S}$ representing possible robot trajectories and current poses is maintained. At each time step t, each particle i generates a new estimate of the robot's pose using the new translation and rotation odometry reading u_t

$$s_t^{[i]} \sim p(s_t^{[i]}|u_t, s_{t-1}^{[i]}) \qquad (17)$$

where Gaussian noise associated with translation and rotation is added as part of the motion model.

B. Map Estimation

At each time step t, a new set of environmental observations $\mathbf{Z}$ is received. Then each particle i computes a map of the environment represented by a set of line segments or clusters C_i using the incremental probabilistic principal component analysis procedure described above.

C. Particle Resampling

Initially all particles simply update the robot's pose based on the new odometry reading. This creates a particle set whose distribution is independent of the environmental observations. To correct for this, importance sampling [11] weights each particle based on the likelihood of the observations given the particle pose, and performs weighted resampling to generate a new particle set. In the present scenario, PPCA provides a probability distribution over the data for a given cluster. A particle i's weight w_i at iteration t is thereby computed by:

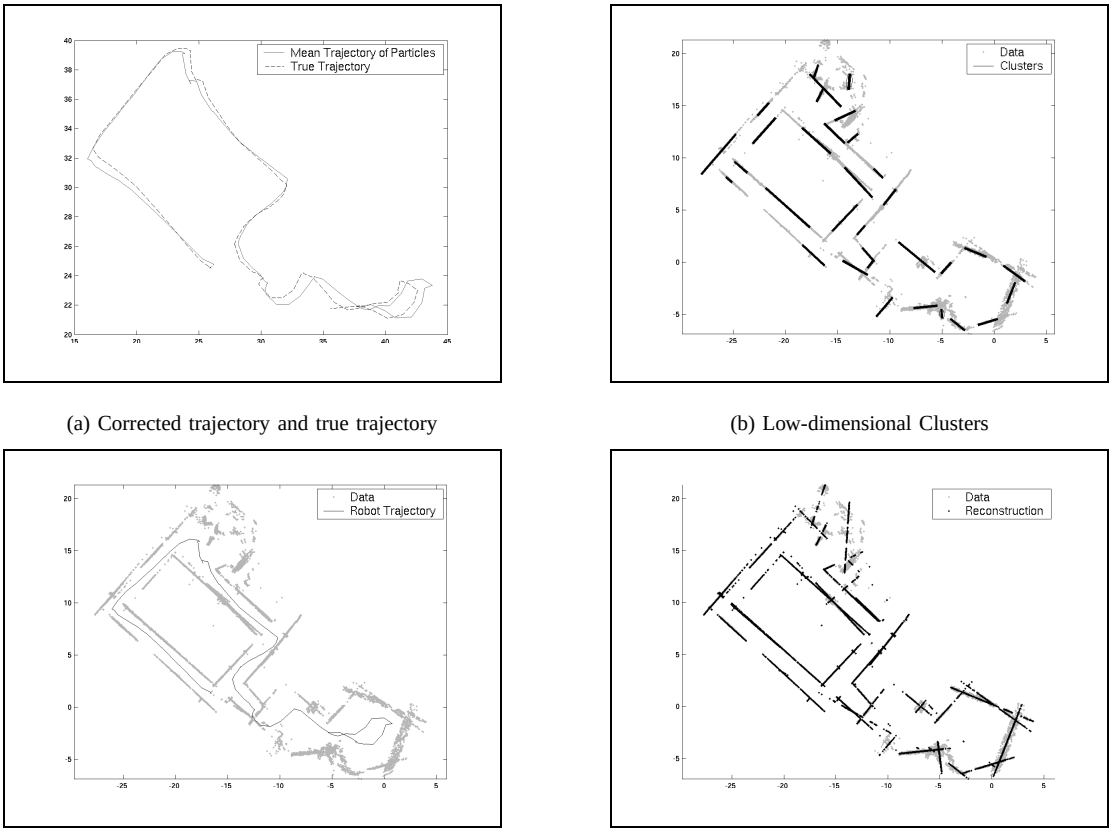
$$w_i = \prod_{j=1}^{|\mathbf{Z}^{[t]}|} \max p(z_j^{[t]}|\mathbf{C_i}) \qquad (18)$$

In other words, each measurement is assigned to the cluster that maximizes its probability, and so the weight of a particle represents the likelihood it assigns to the current set of observations $\mathbf{Z}^{[\mathbf{t}]}$.

Particles are then resampled with replacement before the next time step. The complete algorithm can be seen in table I.

V. Experimental Results

Our algorithm was tested on a dataset collected from the MIT Stata center. 201 particles were used. Figure 4a shows the true robot trajectory compared to the mean particle trajectory over time. The alignment is generally good; unfortunately this data set does not close the large loop entirely, but we see that the small loop in the trajectory is closed appropriately, and the small shift at the end of the



(a) Corrected trajectory and true trajectory

(b) Low-dimensional Clusters

(c) Registered laser data

(d) Reconstructed laser data

Fig. 4. The map built from SLAM using the low-dimensional representation. (a) The corrected trajectory is the solid black line, and the true trajectory is the solid grey line. The true trajectory was found by tracking the robot's position in a pre-built grid map. (b) The constructed map, represented by the clusters. (c) The map as represented by registered laser data. (d) The map as represented by the low-dimensional projection of laser data.

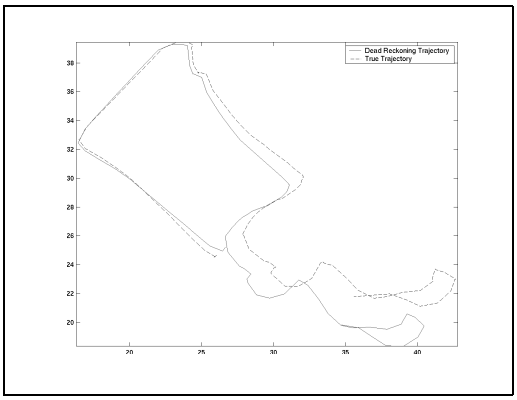


Fig. 3. The map built from pose estimates given by raw odometry. As expected, the error in the pose estimates accumulates and the resulting map is poor.

trajectory would no doubt be corrected with more data. (It is possible to introduce arbitrary error in the L_2 norm between a perfect map and ground truth by rotating the map about the start pose before comparison. The trajectory shown here follows a correction procedure to shift and rotate the map back to the best orientation possible for minimizing error.)

Figure 4b shows the map clusters associated by the mean particle on top of the laser data. Note that the clusters are probability distributions, not line segments, and are therefore plotted by showing one standard deviation of the data likelihood distribution on either side of the cluster center; the clusters may seem shortened but are capable of representing more structure than is plotted. Data points whose likelihood under any cluster was lower than a threshold (0.01) were not used in computing cluster parameters, and data with probability less than 10^{-8} are not displayed. The map has clearly captured the majority of the high level structure of the environment, using a fraction of the parameters needed for a point based representation. Indeed, the final mean particle map consisted of 33 clusters, approximately .5% of the parameters needed to represent the full point-based representation. Figure 4c shows the mean particle robot trajectory through the environment.

In addition, the reconstruction of the raw data points was computed by assigning data points to the cluster that maximized their likelihood, projecting down to the low dimensional representation, and then re-projecting up to the high dimensional space. Figure 4d shows the reconstructed data points. As would be desired, the reconstructed data points provide a nice, low noise representation of the environment.

VI. Related Work

There is a large body of work on SLAM and only the most relevant results will be discussed here. Leonard et. al. [6] used the Hough transform to group observations into point or plane features when performing concurrent mapping and

1) i=1
2) Initialize particle filter S with odometry reading i
3) Collect first set of observations $\mathbf{Z_0}$
4) for each particle $s_k \in S$
   a) Run PPCA+G-means on observations $\mathbf{Z_0}$ and output a set of clusters C
   b) $w_k = p(\mathbf{Z_0}|C)$
5) Resample the particles with replacement using weights w
6) While $i <$ num observation sets
   a) Update each particle's position using the new odometry reading
   b) Collect new observations $\mathbf{Z_i}$
   c) for each particle $s_k \in P$
      i) Run PPCA+G-means on observations $\mathbf{Z_i}$ and output a set of clusters C_{new}
      ii) Map new clusters C_{new} to old clusters C where possible
      iii) for j=1:number of matching cluster pairs
         A) Use IPCA to merge cluster pair j into cluster $C_{merge,j}$
      iv) Copy all unmatched old clusters to C_{merge}
      v) If a new cluster has a low non-principal-component variance and sufficient number of new observations Z belonging to it, add it to C_{merge}
      vi) $w_k = p(\mathbf{Z_i}|C)$
   d) Resample the particles with replacement using weights w

TABLE I
THE IPSLAM ALGORITHM

localization (CML). Similar to the approach in this paper, they create new features from a set of new measurements, and then fuse features together. However, this representation lacks a probability model.

Our approach to robot pose estimation and particle resampling draws heavily from Thrun's FastSLAM [12] approach. However, our line-based map representation is fundamentally different to the landmark representation used in FastSLAM and therefore our map estimation approach is novel. Our representation is closely related to Thrun et al.'s Expectation-Maximization approach to line-segment extraction [13], however, our algorithm is an on-line approach to SLAM, rather than a post-processing step on the map.

Artač et. al [1] used incremental PCA to store low-dimensional representations of images taken by a mobile robot as it moved around an environment. Comparing test images to these stored low-dimensional representations produced a good estimation of the location of the robot during acquisition of the test images. However, this work does not attempt to solve the problem of simultaneous localization and mapping.

VII. CONCLUSION

In this paper, we have described the IPSLAM algorithm, an algorithm for simultaneously localizing a robot and estimating a map of the environment based on low-dimensional geometric representation. The number of features in the environment is estimated dynamically using an approach drawing from the G-means algorithm [4]. Using this algorithm we are able to generate good low-dimensional representations of environmental structure using many fewer parameters than conventional point-based or grid-based representations. The algorithm is near-real-time, in that the major loss of real-time performance can be attributed to the language implementation (Matlab). We intend to re-implement the algorithm and demonstrate real-time performance shortly.

There do not exist good metrics for comparing map quality, however, visual comparison of our generated map suggests it corresponds well with a map built using a grid-based representation, including detecting smaller details like doors. Notice that the passageway appears to be partially blocked across between the two main segments of the map—this is in fact a doorway that is partially closed in the dataset.

The generated map does contain some errors, however. The incremental PCA algorithm is not based on a likelihood model and does make errors in merging clusters periodically. We intend to extend the incremental PCA algorithm to incorporate the full data likelihood model in future work. Additionally, a major source of error in the map is the fact that the MIT Stata Centre is a particularly difficult environment to model using straight-line segments, because many of the walls are in fact not flat or square. For example, the lower-right portion of the map, represented by a roughly pentagon-shaped outline of line segments. This is in fact a curved wall segment, which requires many small linear models to accurately represent. One possible extension to the IPSLAM model is to capture curved segments using non-linear PCA or Principal Curves models [5].

REFERENCES

[1] M. Artač, M. Jogan, and A. Leonardis. Mobile robot localization using an incremental eigenspace model. In *ICRA*, volume 2, 1025–1030, September 2002.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *ICRA*, volume 2, 1899–1906, September 2003.

[3] H. J. S. Feder, J. J. Leonard, and C. M. Smith. Adaptive mobile robot navigation and mapping. *Inter. Journal of Robotics Research, Special Issue on Field and Service Robotics*, 18(7): 650–668, July 1999.

[4] Greg Hammerly and Charles Elkan. Learning the k in k-means. *Neural Information Processing Systems*, 15, 2004.

[5] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406): 502–516, 1989.

[6] J. J. Leonard, R.Rikoski, P.Newman, and M.Bosse. Mapping Partially observable Features from Multiple Uncertain Vantage Points *International Journal of Robotics Research*, 21(10): 943–975, October 2002.

[7] Paul MacKenzie and Gregory Dudek. Precise positioning using model-based maps. In *ICRA*, 1615–1621, San Diego, CA, May 1994.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI*, 2002.

[9] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, 2000.

[10] Robert Sim and Gregory Dudek. Effective exploration strategies for the construction of visual maps. In *IROS*, 2003.

[11] A. Smith and A. Gelfand. Bayesian statistics without tears: a sampling-resampling perspective. *Amer. Statistician*, 46: 84–88, 1992.

[12] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *Inter. Journal of Robotics Research*, 2004. To Appear.

[13] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, W. Burgard. A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots. *IEEE Trans. Robotics and Automation*, 20(3): 433–443, June 2004.

[14] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21(3): 611–622, 1999.

[15] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1): 71–86, 1991.

[16] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principle component analysis. *PAMI*, 25(8): 1034–1040, 2003.