Topological Mapping with Multiple Visual Manifolds

Greg Grudic and Jane Mulligan
Department of Computer Science
University of Colorado at Boulder
Boulder, CO, 80309-0430
Email: $<$ grudic | janem $>$@cs.colorado.edu

Abstract— We address the problem of building topological maps in visual space for robot navigation. The nodes of our topological maps consist of clusters along manifolds, and we propose an unsupervised learning algorithm that automatically constructs these manifolds - the user need only specify the desired number of clusters and the minimum number of images per cluster. This spectral clustering like framework allows each cluster to optimize a separate set of clustering parameters, and we demonstrate empirically that this flexibility can significantly improve clustering results. We further propose a framework for servoing the robot in our manifold space, which would allow the robot to navigate from any point on one manifold (topological node) to any specified point on a second manifold. Finally, we present experimental results on indoor and outdoor image sequences demonstrating the efficacy of the proposed algorithm.

I. Introduction

Autonomous robots require maps of environments in order to effectively accomplish tasks. As a result, turning internal and external sensor readings into maps suitable for navigation, has become a topic of intense interest in robotics research [1]. Mapping techniques can be loosely divided into two types: topological and geometric approaches. Topological approaches can be thought of as robot-centric, or representations in sensor space. In contrast, geometric approaches seek to define sensor based maps in some world coordinate frame [2].

Numerous image-based approaches to topological mapping have been described in the past 10 years [3], [4], [5], [6]. Ideally an agent passes through the environment recording sequential sensor data, and uses this to automatically generate the places and transitions which provide a topological description of the space. In this paper we will describe our approach to clustering regions of the sensor sequence (in our case images) using proximity on low dimensional manifolds in sensor space. Our goal is to develop techniques to navigate on the manifold of clustered sensor sequences. We will not address mapping of images or landmarks to metric pose [3], [7].

Appearance-based mapping approaches typically use PCA representations, However, these inherently map an image to a point in the eigenspace. If the world exhibits perceptual aliasing, where several distinct positions in the sequence look similar, then there may not be enough information encoded in the eigenspace representation to disambiguate them. One obvious solution is to consider information about neighbouring views in space and time. This can be achieved by tracking (as is generally the case in landmark-based systems) or by exploiting spatial continuity in the learning data. In topological maps authors use the spatial solution of traversing nearby nodes to establish whether the current distinguished place has been seen before [8], [9].

Maeda et al. [10] propose precomputing active vision strategies for regions of the manifold which are close in eigenspace. By acquiring additional images they can disambiguate where the current pose falls. Similarly Kröse and Bunschoten [4] acquire additional images by rotating in place, in order distinguish positions with similar appearance. Kelly [11] composes extended image sequences into globally consistent mosaics and then tracks motion over the mosaic to estimate the current pose. Matsumoto et al. [12] navigate in a sequence of images, assuming that the robot makes transitions in order from one memorized image to the next. The method depends on a difference metric which increases monotonically allowing the robot to determine which neighbour in the sequence is closer.

Recent developments in learning in manifold space have lead to algorithms that can effectively find low dimensional structures in image sequences that lie on a single manifold [13]. However, such algorithms require and assume that a single manifold is sufficient to describe the visual data and, in general, do not function well when this is not true. In this paper we address the problem of identifying image clusters from a wide variety of image sequences. We propose a clustering algorithm that takes as input a user specified number of clusters, and a minimum number of images (or data points) per cluster. Given these inputs, the algorithm uses a spectral clustering [14], [15], [16] like framework to directly specify separate clustering parameters for each cluster. This property makes the algorithm significantly different from other spectral clustering algorithms that use the same clustering parameters for all clusters [14], [15], [16], and that generally require the user to specify these parameters. This property allows us to distinguish such common navigational features as branching paths, and correctly identify which manifold the robot is on. The method also provides a probability measure as to whether the current image view lies on a particular manifold.

We propose the following scenario. The robot moves (either via teleoperation or using some autonomous search strategy)

throughout the environment it is intended to work in. The sensor data that it observes is recorded and projected into manifold space. The data is then clustered in manifold space using unsupervised learning. Each cluster manifold in sensor space corresponds to node in a topological map. Building upon the results in [17], each cluster optimizes its own cluster parameters, allowing a greater clustering accuracy. We further propose a framework by which a robot can servo in this manifold space starting from any position in one one manifold, to any position on another manifold.

The proposed algorithm is described in Section II. Section II-A describes the semi-supervised algorithm upon which the proposed algorithm is based. Section II-B describes the extension to clustering. Section II-C describes the algorithm for estimating a separate set of clustering parameters for each cluster. Section II-D outlines the outlier detection framework. Section II-G describes our formulation for servoing on visual manifolds. Experimental results for synthetic data and real indoor and outdoor image sequences are presented in Section III. Conclusion and future work is described in Section IV.

II. ALGORITHM DEVELOPMENT

A. Semi-Supervised Learning

Semi-supervised learning is concerned with building predictive models from both labelled and unlabelled examples.

The algorithm proposed here for learning and servoing on visual manifolds is based on the Local and Global consistency method for semi-supervised learning introduced by Zhou *et.al.* [18]. We present a brief summary of this algorithm below.

Assume a set of N points $(\mathbf{x}_1, ..., \mathbf{x}_N)$ in d dimensional space $\mathbf{x}_i \in \Re^d$. Assume that these points belong to one of Q classes specified by $q \in 1, ..., Q$, and some of these points are labelled as belonging to a class, while others are unlabelled. We define a labelling matrix $Y \in \mathcal{N}^{N \times Q}$ with $Y_{iq} = 1$ if and only if point $\mathbf{x}_i$ has label q and 0 otherwise. This framework assumes that there is at least one example from each class in the set of points $(\mathbf{x}_1, ..., \mathbf{x}_N)$.

Next Zhou *et.al.* [18] define an affinity matrix $W_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$ if $i \neq j$ and 0 otherwise, where $\sigma > 0$ is user specified. The normalized form of this matrix is then computed as $S = D^{-1/2}WD^{-1/2}$ with $D_{ii} = \sum_{j=1}^{n} W_{ij}$ and $D_{ij} = 0, i \neq j$. Finally, an N by C matrix F is calculated as follows:

$$F = (I - \alpha S)^{-1} Y \qquad (1)$$

where I is the identity matrix and $0 < \alpha < 1$ is user specified. The unlabelled points in $(\mathbf{x}_1, ..., \mathbf{x}_N)$ are then labelled in Zhou *et.al.* [18] using:

$$y_i = \underset{j \in \{1, ..., C\}}{\arg \max} F_{ij} \qquad (2)$$

where F_{ij} is the element in matrix F in row i and column j.

It is easy to see that each column i of the matrix $(I - \alpha S)^{-1}$ constitutes an ordering of distances, along a manifold, from training example x_i to all other training examples x_j, for $i, j = 1, 2, ..., n$. It is this property that we will exploit

for clustering along manifolds [19]. More specifically, we will look for columns of $(I - \alpha S)^{-1}$ that define clusters by finding points that are at the center of large manifolds.

A further observation of this algorithm that is pertinent to our clustering algorithm, is that both σ and α are user specified. However, in the clustering formulation proposed in this paper, σ and α are byproducts of the optimization procedure, and are automatically chosen for each cluster (i.e. each cluster many have different values of σ and α). This makes the proposed algorithm more flexible and more automated then other spectral clustering implementations [16].

B. Extension to Clustering

From equation (1), it is evident that the solution to the semi-supervised learning problem only depends on the labels after the the matrix $(I - \alpha S)$ has been inverted. This matrix only contains the training data inputs, $\{x_1, ..., x_n\}$, and it is this property that we exploit to derive our clustering algorithm. We define a matrix U as:

$$U = \beta (I - \alpha S)^{-1} = [u_1^T, ..., u_n^T] \qquad (3)$$

and note that U defines a graph or diffusion kernel as described in [20], [21]. In addition, the columns of U, denoted by u_i^T, define distances between training points on these graphs, which can be interpreted as distances along a manifold [19]. The ordering of these distances along each manifold is maintained independent of scaling. From U, we create a *normalized* matrix V, by scaling the columns of U to have unit length (which we exploit below to efficiently calculate clusters). We define this V matrix as:

$$V = \left[\frac{u_1^T}{\|u_1^T\|}, ..., \frac{u_n^T}{\|u_1^T\|} \right] = [v_1^T, ..., v_n^T] \qquad (4)$$

Note that, by definition, $\|v_i\| = 1$. Finally, we define a distance (along a manifold specified by U) between points x_i and x_j to be:

$$d_M(x_i, x_j) = 1 - v_i v_j^T \qquad (5)$$

Note that because $\|v_k\| = 1$ and each element in v_k is greater than or equal to zero (for all $k = 1, ..., N$), it must be the case that $0 \leq v_i v_j^T \leq 1$. If two points identically rank the relative distances to all other points on a manifold, our framework considers them identical. If this is the case for points x_i and x_j, then $d_M(x_i, x_j) = 0$. Conversely, if the point x_i has completely different distances along U to other points in the training data than point x_j, then $d_M(x_i, x_j)$ will approach 1. This leads to our definition of a distance matrix:

$$D_M = 1 - \begin{pmatrix} v_1 v_1^T & \cdots & v_1 v_n^T \\ \vdots & \ddots & \vdots \\ v_n v_1^T & \cdots & v_n v_n^T \end{pmatrix}$$
$$= \begin{pmatrix} d_M(x_1, x_1) & \cdots & d_M(x_1, x_n) \\ \vdots & \ddots & \vdots \\ d_M(x_n, x_1) & \cdots & d_M(x_n, x_n) \end{pmatrix} \qquad (6)$$

Therefore the (i, j) element of D_M defines a distance between two points x_i and x_j, which ranges between 0 and 1; 0 meaning that the points are identical, and 1 that they are furthest apart from one another.

This allows us to set up the following optimization problem for clustering. In clustering, we want to pick clusters of points that are most similar to one another, while at the same time most different to points in other clusters. We start by assuming that there are c clusters and that each cluster is characterized by a single point. Thus, for c clusters, we have $x_{l_1}, ..., x_{l_c}$ points, where $x_{l_i} \in \{x_1, ..., x_n\}$ is a point in the training data, and $x_{l_i} \neq x_{l_j}$ for $i \neq j$. These points $x_{l_1}, ..., x_{l_c}$ are used to define clusters as follows. We define a n by c matrix F_V^* by taking the $l_1, ..., l_c$ columns of V (see equation (4)):

$$F_V^* = \left[v_{l_1}^T, ..., v_{l_c}^T \right] \qquad (7)$$

Then, as with semi-supervised learning, we assign a point x_i to a cluster:

$$y_i = \arg\max_{j \leq c} F_{Vij}^* \qquad (8)$$

where F_{Vij}^* is the entry of F_V^* given by row i and column j.

In in Section II-E we define an optimization framework for picking points $x_{l_1}, ..., x_{l_c}$ given that c is specified. In the next section we show how the clustering parameters, σ and α, for each cluster, can be directly estimated from the data.

C. Model Selection For Clustering

Let $\mathbf{p}_j$ be the set of points that belong to cluster j. Using matrix D_M we can define the mean distance between points in cluster j as:

$$\overline{D}_M^{jj} = E\left[D_M \left(\mathbf{p}_j, \mathbf{p}_j \right) \right]$$

where $D_M(\mathbf{p}_j, \mathbf{p}_j)$ denotes all entries of D_M (see equation (6)) corresponding to columns and rows of points $\mathbf{p}_j$ and $E[\cdot]$ is the average value of these entries. Similarly, the mean distance between points in cluster j and points in cluster k is given by:

$$\overline{D}_M^{jk} = E\left[D_M \left(\mathbf{p}_j, \mathbf{p}_k \right) \right]$$

Given that our goal is to find clusters that maximize the distances between points in different clusters, while minimizing the distances between points in the same cluster, we can state the following optimization problem: find σ, α, c, and $x_{l_1}, ..., x_{l_c}$ to maximize the following

$$\Omega\left(c\right) = \max_{\alpha, \sigma, c} \left[E\left[\overline{D}_M^{jj} \right]_{\{j=1,...,c\}} - E\left[\overline{D}_M^{jk} \right]_{\left\{ \substack{k=1,...,c \\ j=1,...,c \\ i \neq j} \right\}} \right]$$

However, there are two problems with this approach. First, the optimizing over all σ, α, c can be computationally expensive [17]. Second, and more importantly for the current paper, every cluster will contain the same σ. As is demonstrated by both synthetic and real data in Section III, this can lead to significant degradation in clustering performance when points in different clusters are not equally spaced. To avoid these difficulties, we take a recursive clustering approach, where the

dataset is split into two partitions at each step of the algorithm. The criteria used to determine this split is

$$\Omega\left(2\right) = \max_{\alpha, \sigma} \left[\frac{\min(|\mathbf{p}_1|, |\mathbf{p}_2|)}{|\mathbf{p}_1| + |\mathbf{p}_2|} \left(E\left[\overline{D}_M^{jj} \right]_{\{j=1\}} - E\left[\overline{D}_M^{jk} \right]_{\left\{ \substack{k=1 \\ j=2} \right\}} \right) \right] \qquad (9)$$

where $|\mathbf{p}_j|$ is the number of points in cluster j. This optimization encourages points to be split into two partitions of approximately equal size. In addition, σ and α are chosen such that the first cluster of points consists of points that are tightly clustered on one manifold, while the second cluster includes points which are far away from the first, but need not be on a single manifold. Thus σ and α are always optimized with respect to the first manifold.

This type of recursive partitioning continues until clusters have a minimum number of points (which is user specified by Q). This produces a set of C clusters. The user must also specify the desired number of clusters K. The constraint on K and Q is that $K < C$. The C clusters are finally combined to give the best K clusters that each have maximal values of $\Omega(2)$ in equation (9).

In the next two sections, the description of the proposed algorithm is completed by first defining the concept of cluster outliers in manifold space, and then showing how this is used to define points that seed clusters.

D. Outlier Detection

We define a cluster independent outlier point to be one that is, on average, furthest away from all other points. This can be directly calculated from equation (6) by taking the average of the columns of D_M, and defining a outlier cluster independent vector O_d as follows:

$$O_d = \frac{1}{n} \left[\sum D_{M1}^T, ..., \sum D_{Mn}^T \right] = [O_{d1}, ..., O_{dn}] \qquad (10)$$

where the element O_{di} is the average distance (along a manifold) between point x_i and all the other points and $D_M = \left[D_{M1}^T, ..., D_{Mn}^T \right]$. By ordering the values of O_{di} in decreasing order, we order the points from furthest to closest, and the points appearing first in the list constitute the outliers.

Similarly, we can find outliers within a cluster j by looking at the $D_M^{jj} = D_M(\mathbf{p}_j, \mathbf{p}_j)$ matrix defined above. Specifically, we obtain an outlier O_d^j vector for cluster j as follows:

$$O_d^j = \frac{1}{n} \left[\sum D_{M1}^{jjT}, ..., \sum D_{Mn}^{jjT} \right] = \left[O_{d1}^j, ..., O_{dn}^j \right] \qquad (11)$$

where O_{di}^j is the mean distance of x_j to all other points in its cluster. Thus the point which has maximum O_{di}^j is the one which is *most outside* the cluster (i.e. the greatest outlier), while the point that has minimum O_{di}^j is *most inside* of the cluster.

E. Seeding Clusters

As outlined above, the points x_{l_1}, x_{l_2} are used to specify clusters $\Omega(2)$ in equation (9). These points can be identified by looking at the cluster independent outlier vector O_d defined above. In this paper we use the following greedy heuristic to identify $x_{l_1}, ..., x_{l_c}$. First we assign x_{l_1} to the point that is closest to all other points, which is defined by the point that has the smallest value O_{di}. To find x_{l_2}, we multiply each element of $1 - O_d$ by the corresponding element in the column vector $D_{Ml_1}^T$, to obtain an new, re-weighted vector of O_d^2 as follows:

$$O_d^2 = \left[O_{d1}^1 D_{Ml_1}^T(1), ..., O_{dn}^1 D_{Ml_1}^T(n)\right] = \left[O_{d1}^2, ..., O_{dn}^2\right] \tag{12}$$

where $O_{di}^1 = O_{di}$. This re-weighting serves to separate out points in the first cluster seeded by x_{l_1}. The point x_{l_2} then corresponds to the point which has minimum O_{di}^2.

F. Assigning New Points to Clusters

Let $\mathbf{p}_j$ be the set of points in cluster j, let σ_j be the associated Gaussian kernel parameter, and let x_{l_j} be the point which defines the center of the cluster (as defined above). Symbolize the new point being classified into one of the clusters as $\mathbf{x}$, and $\mathbf{p}_j^* = \mathbf{p}_j \bigcup \mathbf{x}$ be the set of points that includes points in cluster j and the new point $\mathbf{x}$ being classified. These points are used to calculate an affinity matrix W^j (using σ_j) and corresponding S^j. Finally, we can calculate

$$F^j = \beta \left(I - \alpha S^j\right)^{-1} Y^j \tag{13}$$

Where $\alpha = 0.99$ (set arbitrarily), and Y^j is a vector of the same length as $\mathbf{p}_j^*$, containing zeros everywhere except at the position associated with the center of cluster j, namely x_{l_j}. If the index of the new point $\mathbf{x}$ is k, then the value of $F^j(k)$ defines how close this new point is to the center of cluster j - the larger $F^j(k)$ is, the closer this point is to the center of the cluster. Therefore, we assign point $\mathbf{x}$ to cluster y using:

$$y = \arg\max_j F^j(k)$$

G. A formulation for Servoing on Visual Manifolds

Our formulation of servoing on the manifold is based on the formulation given in the previous section. Assume that Y^j is a vector of the same length as $\mathbf{p}_j^*$, containing zeros everywhere except at the position associated with the robot's desired position on the manifold of cluster j. If the index of the robot's current position (i.e. what the robot currently sees) $\mathbf{x}$ is k, then the value of $F^j(k)$ defines how close this new point is to the desired manifold position. Therefor, we propose to servo the robot to maximize the signal $F^j(k)$ until it reaches the desired manifold location. In the initial implementation this minimization is done via minimization in sensor space.

This framework allows the robot to establish any position and any visual manifold that it has previously encountered. It also allows traversal from the end of one manifold to the start of another.

H. Implementation Details

The optimization problem in equation (9) is solved using the matlab function *fmincon*. This gives a two dimensional optimization formulation in σ and α space to maximize $\Omega(2)$ in equation (9).

III. EXPERIMENTAL RESULTS

A. Synthetic Data

Four synthetic data sets, shown in Figure 1, were used to illustrate the functioning of the proposed algorithm. The first three show points along paths in 2 dimensional space, and the last shows points sampled from paths in 3 dimensional space.

For all the synthetic data sets, the number of clusters was set to $K = 3$, and minimum number of points per cluster was set to $Q = 20$. In order to illustrate the properties of the proposed algorithm, we compared it to the Normalized Cuts algorithm [14] on four synthetic data sets.
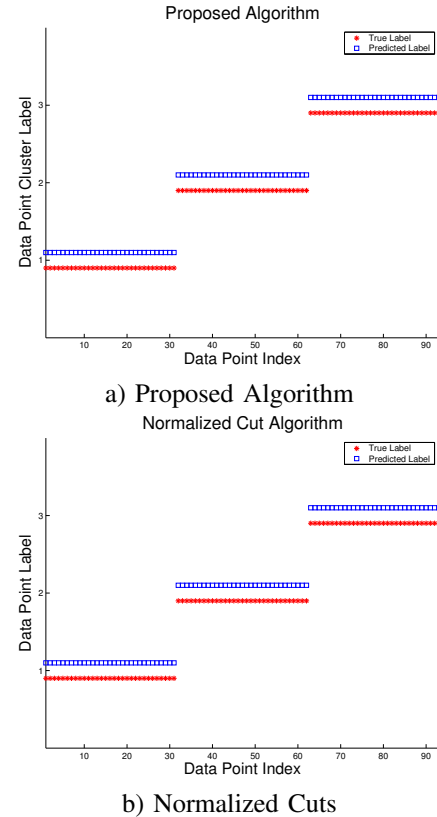

a) Proposed Algorithm

b) Normalized Cuts

Fig. 2. Line Data Results.

The first synthetic data set consists points sampled from three parallel lines 2D (see Figure 1a). The points are equally spaced for each line. Figure 2b shows the results obtained with algorithm proposed in this paper. The Y axis shows the clusters and the X axis shows the point indices. The red stars indicate the true labels and blue squares indicate the predicted labels. Similarly, Figure 2c shows the results obtained with Normalized Cuts algorithm with $\sigma = 0.085$. Note that both algorithms successfully clustered this data set.
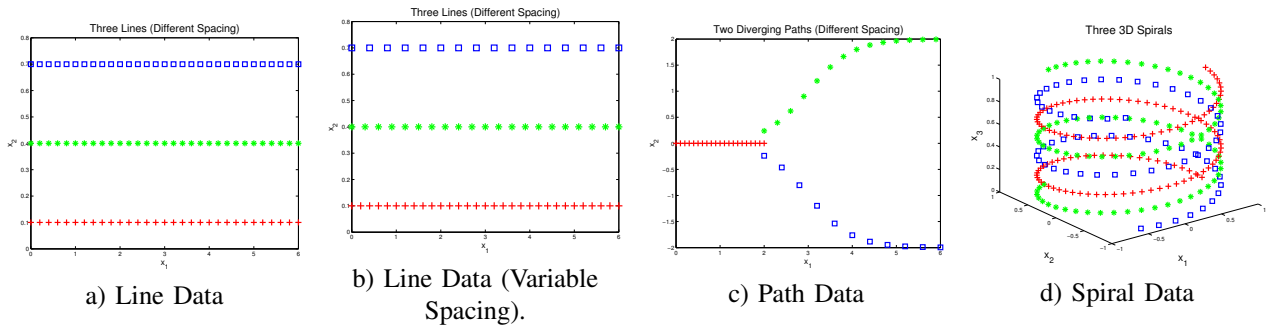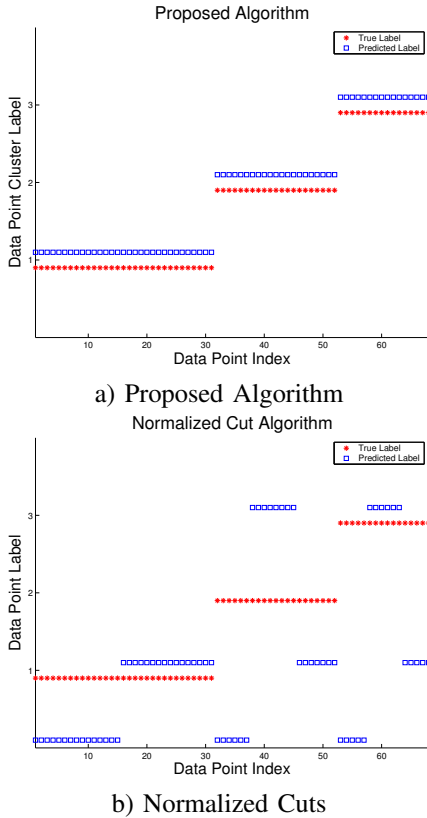
a) Line Data

b) Line Data (Variable Spacing).

c) Path Data

d) Spiral Data

Fig. 1.   Synthetic datasets.

to a single sigma for all three clusters.

a) Proposed Algorithm

b) Normalized Cuts

Fig. 3.   Synthetic Line Data (Variable Spacing).

a) Proposed Algorithm

b) Normalized Cuts

Fig. 4.   Synthetic Path Data.

In contrast the second data set contains points sampled at different intervals from each of three lines (see Figure 1b). Figure 3a shows the results obtained with algorithm proposed in this paper. The Y axis shows the clusters and the X axis shows the point indices. The red stars indicate the true labels and blue squares indicate the predicted labels. Similarly, Figure 3b shows the results obtained with Normalized Cuts algorithm with $\sigma = 0.075$. This value for sigma constitutes the lowest clustering error (see [16] for a definition of clustering error) for the Normalized Cuts algorithm. Note that the proposed algorithm was able to successfully cluster this data, while the Normalized Cuts algorithm was not. This is because the algorithm proposed in this paper selected a different sigma for each cluster, while the Normalized Cuts algorithm was limited

The third data set contains points sampled at different intervals from each of three paths that that converge at a single point (see Figure 1c). Figure 4a shows the results obtained with algorithm proposed in this paper. The Y axis shows the clusters and the X axis shows the point indices. The red stars indicate the true labels and blue squares indicate the predicted labels. Similarly, Figure 4b shows the results obtained with Normalized Cuts algorithm with $\sigma = 0.079$. This value for sigma constitutes the lowest clustering error (see [16] for a definition of clustering error) for the Normalized Cuts algorithm. As with the second data set, the proposed algorithm was able to successfully cluster this data, while the Normalized Cuts algorithm was not. Once more, this is the

algorithm proposed in this paper selected a different sigma for each cluster, while the Normalized Cuts algorithm was limited to a single sigma for all three clusters.
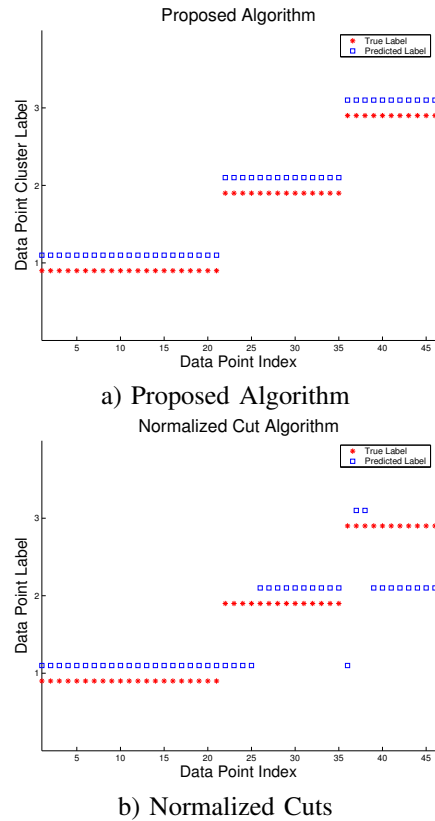

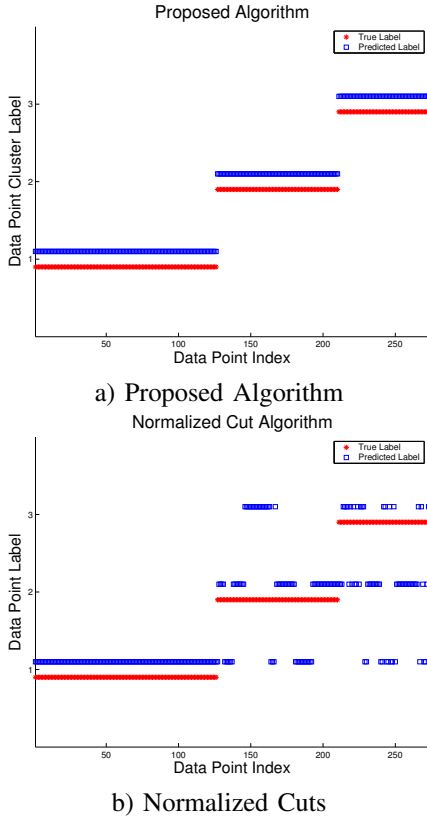a) Proposed Algorithm

b) Normalized Cuts

Fig. 5.   Synthetic Spiral Data.

The final data set contains points sampled at different intervals from each of three spirals in 3D space (see Figure 1d). Figure 5b shows the results obtained with algorithm proposed in this paper. The Y axis shows the clusters and the X axis shows the point indices. The red stars indicate the true labels and blue squares indicate the predicted labels. Similarly, Figure 5c shows the results obtained with Normalized Cuts algorithm with $\sigma = 0.051$. This value for sigma constitutes the lowest clustering error (see [16] for a definition of clustering error) for the Normalized Cuts algorithm. As with the previous two data sets, the proposed algorithm was able to successfully cluster this data, while the Normalized Cuts algorithm was not. Once more, this is the algorithm proposed in this paper selected a different sigma for each cluster, while the Normalized Cuts algorithm was limited to a single sigma for all three clusters.

These synthetic sets show that the Normalized Cuts algorithm is very sensitive to sampling intervals within cluster points.

The running times of the proposed algorithm (matlab implementation) on these datasets ranged from a few seconds to a few minutes (on a 1.8 GHz Pentium 4 running Windows XP).

B. Indoor and Outdoor Image Data

The real world data sets consists of 6 sets of 320 by 240 RGB images (therefore each image is defined by 3x320x240 pixels). Each sequence contains 150 images, for a total of 900 images. These images are summarized in Figure 7. Each row in Figure 7 shows one of 8 clusters. The first cluster (in the first row) shows images down an indoor hallway. The second cluster (in the second row) shows images obtained after exiting the hallway and turning left. The third cluster (in the third row) shows images obtained after exiting the hallway and continuing straight. The fourth cluster (in the fourth row) shows images obtained after exiting the hallway and turning right.

The last 4 rows show outdoor scenes. The fifth cluster (in the fifth row) shows images obtained while walking on a lawn towards a driveway. The sixth cluster (in the sixth row) shows images obtained after leaving the lawn and turning left. The seventh set of images (in the seventh row) shows images obtained after leaving the lawn and continuing straight. Finally, the eighth cluster (in the eighth row) shows images obtained after leaving the lawn and turning right.

75 images from each sequence were used to construct the manifolds and the remaining 75 images (every second one) were used to test the localization on each manifold. The results on the test data for the algorithm proposed in Section II-F are given in Figure 6a (we set $K = 8$ clusters and $Q = 35$ minimum points per cluster). The 8 clusters are defined on the Y axis. The red stars indicate the true labels and blue squares indicate the predicted labels. The clustering error of the proposed algorithm on the test data is 2.89%.

The results obtained for the randomized cuts algorithm with a single sigma set to 186.5 (experimentation indicated that this sigma resulted in the best clustering error) are given in Figure 6b. The test data was classified using the out-of-sample algorithm derived in [22]. The clustering error for this algorithm on the test data was 34.4%.

Thus the algorithm proposed in this paper gave a 92% improvement in clustering error over the Normalized Cuts algorithm by allowing each cluster to individually optimized for sigma.

The running time of the proposed algorithm on these real datasets (matlab implementation) was 16 minutes on a 1.8 GHz Pentium 4 running Windows XP.

IV. SUMMARY AND CONCLUSIONS

We proposed a formulation for autonomously building topological maps in a robot's visual space. Each node in the topological map consists of the cluster along a manifold. The algorithm presented allows each cluster to independently optimize it's own clustering parameters. This results in a spectral clustering like procedure that can significantly outperform the clustering performance of algorithms which constrain all clusters to have the same affinity matrix parameters. The proposed algorithm requires the user to specify the desired number of clusters and the minimum number of points per

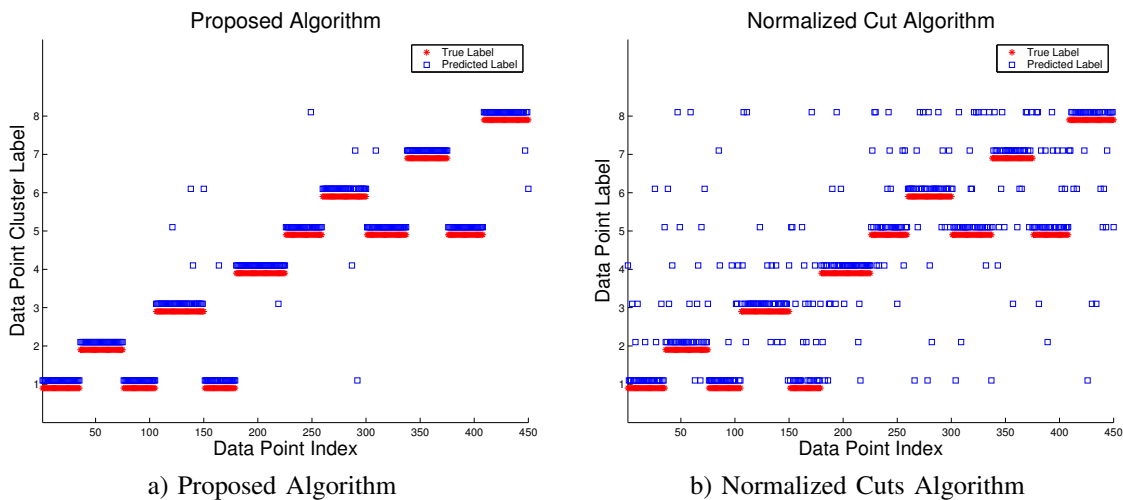a) Proposed Algorithm   b) Normalized Cuts Algorithm

Fig. 6. Results on Image Data.

cluster - the affinity matrix parameters are completely defined by these two parameters.

This paper suggests a number of open research questions. First, the optimization procedure posed was solved using an off the shelf optimization algorithm (matlab optimization in 2D). Reanalyzing the theory behind this optimization may allow a more direct solution to this optimization problem. Second, we outlined a procedure for servoing the robot along the manifolds it has previously encountered, but we have not yet evaluated this formulation on a real robot. This constitutes the most important experimental next step in our overall formulation. It will allow the robot to traverse from any point on one manifold, to any point on another manifold. Third, the user is required to specify the number of desired clusters. The framework developed here may allow the number of clusters to be directly optimized for. Finally, the initial results presented appear promising and further experimental investigation appears warranted.

ACKNOWLEDGMENT

REFERENCES

[1] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[2] Emilio Remolina and Benjamin Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152:47–104, 2004.

[3] Fabrice Pourraz and James L. Crowley. Continuity properties of the appearance manifold for mobile robot position estimation. In *Symposium for Intelligent Robotics Systems, SIRS'98, 1998. To Appear.*, 1998.

[4] B. J. A. Kröse and R. Bunschoten. Probabilistic localization by appearance models and active vision. In *Proc. of the 1999 IEEE International Conference on Robotics and Automation*, pages 2255–2260, Detroit, MI, May 1999.

[5] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc of the 2000 IEEE Int. Conf. on Robotics and Automation*, pages 1023–1029, San Francisco, CA, April 2000.

[6] Benjamin Kuipers and Patrick Beeson. Bootstrap learning for place recognition. In *Proc of the 18th Natnl Conf on AI (AAAI02)*, 2002.

[7] Robert Sim and Gregory Dudek. Comparing image-based localization methods. In *Proc Int. Joint Conf. on AI (IJCAI03)*, 2003.

[8] B. J. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[9] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robot exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, Dec 1991.

[10] S. Maeda, Y. Kuno, and Y. Shirai. Active navigation vision based on eigenspace analysis. In *Proc. 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1018–1023, 1997.

[11] Alonzo Kelly. Mobile robot localization from large scale appearance mosaics. *International Journal of Robotics Research*, 19(11):1104–1125, 2000.

[12] Yoshio Matsumoto, Masayuki Inaba, and Hirochika Inoue. View-based approach to robot navigation. In *Proceedings of 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2000)*, pages 1702–1708, Takamatsu, Japan, Nov 2000.

[13] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Patter Recognition (CVPR'04)*, 2004.

[14] Jiambo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2002.

[15] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2002.

[16] Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms. Technical Report 03-05-01, Department of Computer Science and Engineering , University of Washington, 2003.

[17] Markus Breitenbach and Greg Grudic. Clustering through ranking on manifolds. In *The 22nd International Conference on Machine Learning (ICML 2005)*, 2005.

[18] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2004.

[19] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2004.

[20] J. R Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1983.

[21] J. Shrager, T. Hogg, and B. A. Huberman. Observation of phase transitions in spreading activation networks. *Science*, 236:1092–1094, 1987.

[22] Pascal Vincent Yoshua Bengio, Jean-Francois Paiement. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.
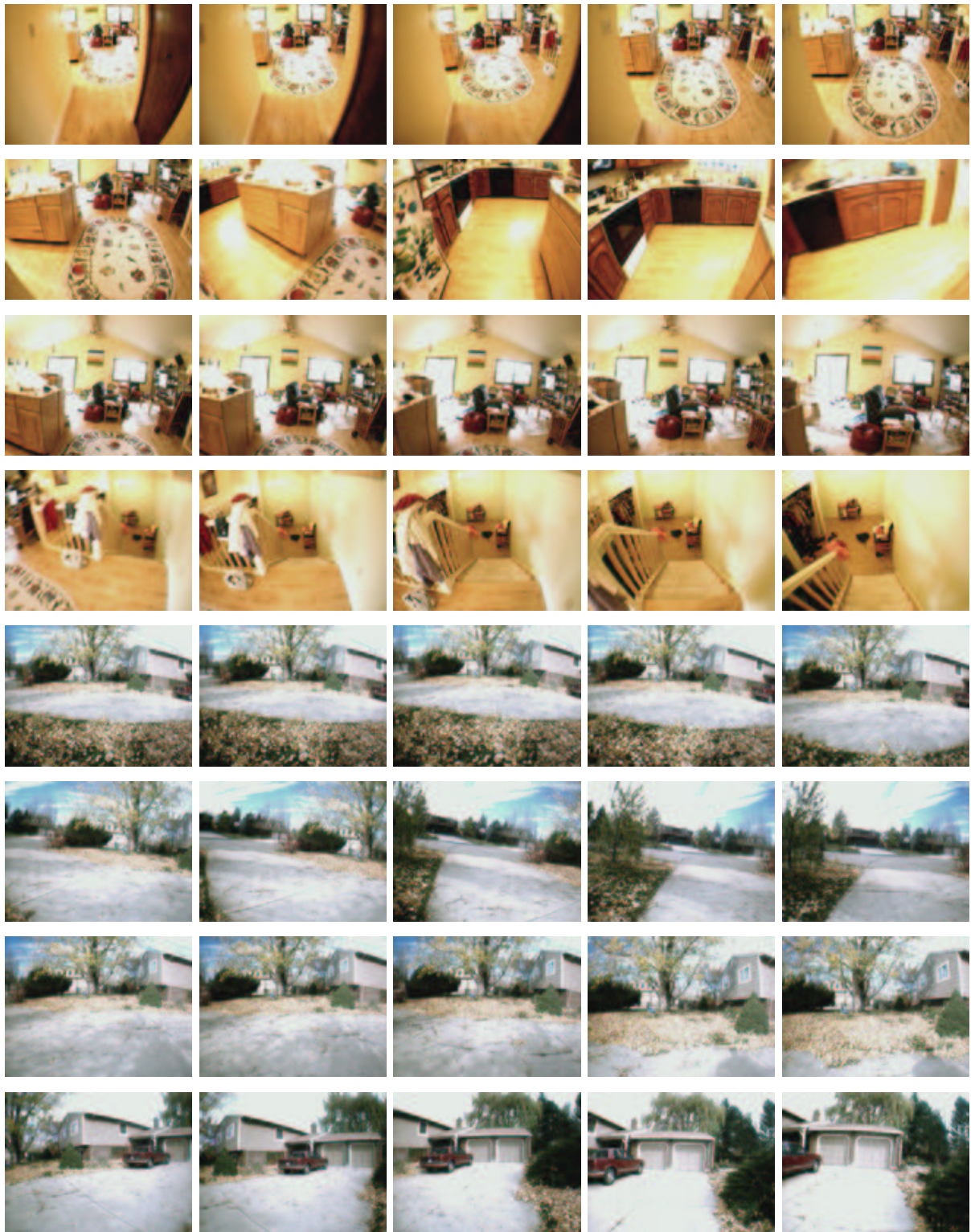
Fig. 7.   Images along 8 manifolds in indoor and outdoor environments.