Topological Mapping Using Spectral Clustering and Classification

Emma Brunskill*, Thomas Kollar* and Nicholas Roy

***Abstract*— In this work we present an online method for generating topological maps from raw sensor information. We first describe an algorithm to automatically decompose a map into submap segments using a graph partitioning technique known as spectral clustering. We then describe how to train a classifier to recognize graph submaps from laser signatures using the AdaBoost machine learning algorithm. We demonstrate that the we can perform topological mapping by incrementally segmenting the world as the robot moves through its environment, and we can close the loop when the learned classifier recognizes that the robot has returned to a previously visited location.**

I. Introduction

Robotic mapping remains one of the key areas of robotics, in part because in order for a robot to perform a large number of useful tasks, a robot must be able to plan collision-free trajectories and track its own position within an environment as it moves. There is a huge body of work in automatically learning a map of the environment from sensor data such as laser range scans without an external source of global position information, known as the Simultaneous Localization and Mapping (SLAM) problem[5], [10], [1]. Full metric maps can consist of thousands of states in large environments; building such maps and using them in planning may be computationally expensive. A hybrid topological-metric map that consists of small submaps linked in a graph can greatly reduce the computational demands of mapping and planning in large environments. We propose an on-line algorithm that uses spectral clustering and machine learning techniques to construct a hybrid topological-metric map of a new environment.

Our approach has several benefits. One challenge in topological mapping is selecting how to segment an environment into smaller submaps while ensuring that planning is not hindered by a suboptimal topological division [10]. Spectral clustering segments the world into subregions that tend to be strongly connected in a graph-theoretic sense: in particular it tends to segment locations which are weak connectors between tightly connected regions. Weak connectors tend to be associated with corners or doorways in building environments, and therefore spectral clustering provides a natural division of the space that is intuitive for navigation.

Identifying when the robot returns to a previously visited location, known as loop closing, is an essential capability of any robust mapping algorithm. In our approach we use classifiers to identify when the robot revisits a location. These classifiers work by identifying a set of features that serves to uniquely distinguish a particular submap from all other submaps encountered so far. This approach has the benefit that it depends not on the loop length, but on the presence of a unique set of feature values to identify a location, and therefore is likely to scale well to large environments. In contrast to related work [8], we will use machine learning techniques (specifically, boosting classification) to automatically learn which features (and their associated values) from a large feature set best uniquely identify each submap.

Our mapping procedure automatically learns a topological representation of the world. As the robot explores, new range data is added to the current submap using standard gridmap techniques. When the clustering algorithm detects that the robot has entered a new submap, the previous submap is segmented out. As each submap is segmented out, a classifier is trained to recognize the submap from its perceptual signature. An individual submap is therefore associated both with a metric submap and a classifier that predicts whether an individual laser scan comes from that subregion. As a submap is entered, the existing classifiers are used to determine if a previous submap has been re-entered or if a new submap must be created. The complete algorithm for map construction is given in Algorithm II-A.

The rest of the paper will be organized as follows. We will first discuss in Section II the application of spectral clustering algorithm to mapping. In Section III, we describe submap recognition and loop closure. Then in Section IV we provide results. Finally we conclude with a discussion of related and future work in Sections V and VI.

II. Spectral Mapping

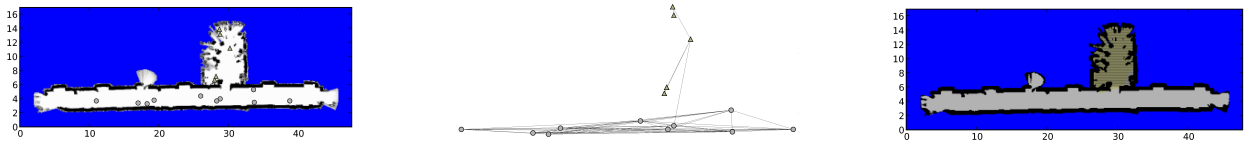
We first require a method that can subdivide the environment into a set of submaps. We would like a method that is unsupervised[1] and can be used in an online fashion. Using the intuition that the best submaps consist of points that are spatially correlated,, we employ spatial clustering to extract submaps in an unsupervised fashion. The goal of clustering is to divide a set of n-dimensional points into k maximally similar subsets. The statistics literature contains many variants of such algorithms and spectral clustering has been shown to be one of the most efficient and robust.

A. Spectral Clustering

Spectral clustering is a graph partitioning algorithm; it divides graph nodes into groups so that connectivity is maximized between nodes in the same cluster and the connectivity is minimized between nodes in different clusters. Certain forms of spectral clustering can approximate the normalized cut of a graph, which roughly correspond to partitioning a graph into k sections of equal size, minimizing the overall interconnection between groups [9].

*The first two authors contributed equally to this paper. E. Brunskill, T.Kollar and N.Roy are members of CSAIL, MIT. emma@csail.mit.edu,(kollar,nickroy)@mit.edu

[1]Unsupervised learning implies that no external labeling is required for the learner to recognize that some training data is different from other training data. In a supervised setting, some external source would be required to label which places in the world are in the same submaps: this is generally not possible for autonomous map building.



(a) Sampling (b) Affinity graph (c) Partitioning

Fig. 1. The steps of spectral clustering.

Algorithm 1 Map Construction Algorithm Outline

While not finished creating map

- Move forward and gather a new set of laser scan observations
- Scan match new observations and add to submap
- Every 50 steps, use spectral clustering to detect if discovered a new submap (Sections II-A and II-B).
- If found new submap
  - Use learned classifiers to determine if the new submap is existing submap (Section III-B).
  - If recognized as a previous submap, merge new submap with previous submap and close loop in topological map.
  - If not recognized as a previous submap, add submap to graph and train new classifier for new submap (Section III-A).

Algorithm 2 Spectral Clustering

- Form the similarity matrix S.
- Define D a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} S_{ij}$ and $L = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$
- Find the number of clusters k from eigenanalysis of L
- Find $x_1 \dots x_k$ the largest eigenvectors of L and form $X = \begin{pmatrix} | & | & | \\ x_1 & x_2 & x_k \\ | & | & | \end{pmatrix} \in \Re^{n \times k}$
- Set Y to be X with rows re-normalized to have unit length s.t. $Y_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{\frac{1}{2}}}$
- Use k-means clustering on the rows of Y
- Assign the original point $s_i$to cluster k iff row i of Y was assigned to cluster k

Given a set of n points $V = \{v_1, v_2 \dots, v_n\}$ in R^l cluster them into k clusters as follows:

At the core of the standard spectral clustering algorithm [12] is the similarity matrix S which defines the distance of one point to another. Each entry S_{ij} in the matrix is a scale-invariant distance given by

$$S_{ij} = \begin{cases} e^{(-\frac{1}{\sigma_i \sigma_j} \|v_i - v_j\|^2)} \text{for } i \neq j \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

Note here that v_i and v_j are the points to be clustered and, following [12], each σ_i is defined as the median of the l nearest neighbors. The weighting terms σ_i, σ_j provide scale invariance so that neighbors only appear distant in relation to other neighbors. To ensure that our clustering is based on connectivity we normalize the similarity matrix by computing a diagonal normalization term D such that $D_{ii} = \sum_{j=1}^{n} S_{ij}$, and then compute the normalized similarity as $L = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$. The eigenvalues of the L normalized similarity matrix are used to determine the number of clusters (subgraphs) for a given set of nodes, and the projection of each node on to the eigenvectors of the matrix determines its cluster.

B. Applying Spectral Clustering

Each grid map can be viewed as a graph of nodes with each node at map position (x, y) and edges placed between all pairs of mutually visible locations, where visibility is given by whether a straight line between the places lies in free space. Spectral clustering will then tend to segment a map into clusters of nodes with the fewest connections between nodes. That is, spectral clustering will tend to cluster contiguous "chunks" of space together. We can obtain a substantial increase in computational efficiency by down-sampling the grid map into an actual graph. Map segmentation using spectral clustering as described in Algorithm 2 is performed according to the following steps depicted in Fig use 1. First, a set of points is sampled in the free space of the metric map. These constitute the nodes in our graph, shown by the dots in Figure 1a. Edges are then generated between mutually-visible points based on straight line visibility determined by the metric map, shown by the graph in Figure 1b. Each edge is weighted by the distance between the points (distances between points with no edge are set to 0). σ_i is computed and the similarity matrix is created according to equation 1. Spectral clustering is used to compute the eigenvectors of the resulting graph (Algorithm 2) and to partition the points into spatially significant regions. Using nearest neighbor, each grid cell is then given a label, as shown in Figure 1c. Figure 2 shows spectral clustering applied to a more complicated map.

C. Map Creation

The algorithm described above assumes that we have an existing metric map that we are segmenting, but we can also apply this algorithm to an incomplete map that is being built incrementally. From an initial location, the robot is driven through the environment, collecting laser scan readings and registering them using scan matching and its local odometry. Every few complete laser scan readings, spectral clustering is run on the current submap of the robot. The eigenvalues of the resulting map will determine whether a new region should be introduced. This essentially reduces to the problem of identifying the number of regions (or clusters) that spectral clustering should return. The magnitude of the eigenvalues of the similarity matrix generally correlate with the number of clusters in the graph, and a large discontinuity in the matrix spectrum can often be used to determine the number of clusters. In particular, when the robot's trajectory starts in one cluster and transitions to a new cluster, the magnitude of the second eigenvalue generally changssubstantially. Figure 3

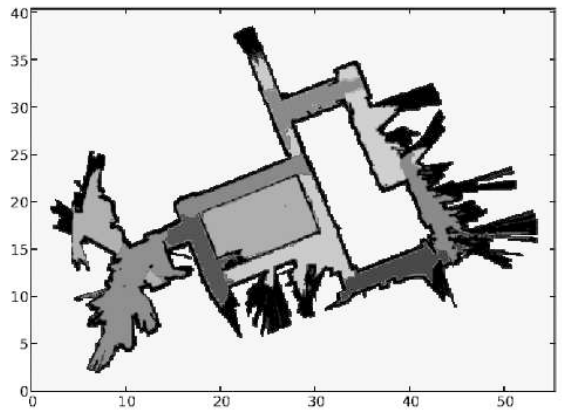


Fig. 2. The Stata Center, segmented into different submaps (different shades) by the spectral clustering algorithm.

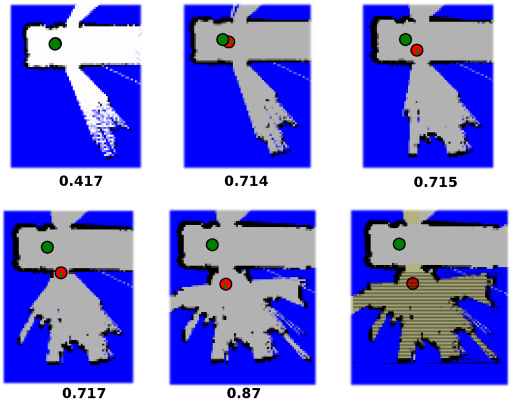


Fig. 3. Eigenvalue sequence as the robot (moving dot) moves through the environment. TThe numbers below each map represent the second largest eigenvalue. Notice the large transition in eigenvalue magnitude in the second and fifth map.

shows a large transition in the second eigenvalue magnitude in the second map, as the robot discovers the second room, and again in the fifth map as the robot fully enters the second room. At this point, the spectral clustering determines that the submap should be two submaps, and a new submap is created consisting of the map section containing the old cluster. The threshold on submap generation for the second eigenvalue was set experimentally to 0.8.

This process is repeated as the robot is driven around the new environment. These submaps, along with the connectivity between submaps, form a map of the new environment which is both metric (inside each submap) and topological (the connections between the submaps). In contrast to fully metric map building approaches, there is no full alignment done between the submaps.

III. Submap Recognition

Any map representation must allow an agent to quickly localize itself within the map. This challenge is particularly prevalent in online mapping algorithms that lack global position information since the robot must be able to recognize it has returned to a previously visited location in order to close loops in the environment. In our hybrid-topological map representation, localization corresponds to submap identification. In the online map creation algorithm we have described so far we have not outlined how the robot can localize itself to a given submap. In addition, anytime the robot re-visits one of these submaps, it will create a new submap of this area. In order to avoid redundant submaps, there must be a mechanism to identify that a new submap is the same as a pre-existing submap, and close the resulting loop. Submap identification is done by learning to label the robot's submap from local laser scans using the AdaBoost algorithm [2]. Loop closure is performed by identifying when the classifiers associated with two submaps overlap sufficiently that the two submaps are merged.

A. AdaBoost

AdaBoost is a popular machine learning technique [2] that learns a binary classifier $h(x)$ given a set of labeled training example pairs (y_i, x_i) where y_i is the label (± 1) of the example x_i (which consists of k features $f_{i1} \ldots f_{ik}$). The label of new examples x_i' is then predicted by the sign of the classifier $h(x_i')$. The key idea of AdaBoost is to produce a series of weak Boolean classifiers $h_j(x)$, each that predict the labels of the training set x_i with better than random (50 percent) accuracy. Each weak classifier h_j is assigned a certain number of votes α_j and the strong classifier is composed of the sum of the weighted weak classifiers $\sum_j \alpha_j h_j(x)$. We follow Viola and Jones' AdaBoost variant [11], in which weak classifiers $h_j(x)$ consist of thresholding on a single feature f_k to produce an estimate of the class label:

$$h_j(x_i) = \begin{cases} 1 & \text{if } p_j f_{ki} < t_j \\ -1 & \text{otherwise} \end{cases}$$

where p_j is 1 or -1 depending on the inequality direction, and t_j is the threshold value. For example, if a feature f_k is the length of the longest wall in the current laser scan, then a weak classifier h_j associated with some class (submap) j might vote 1 if the wall is longer than some threshold t_j, otherwise the h_j will vote -1 against the laser scan being in class j. For each feature, the optimal values of p_j and t_j are determined by exhaustively enumerating all possible split points in the training data set along with the sign of p_j to minimize the misclassification rate of the weak classifier on the training data set.

For the first weak classifier h_j, the feature f_j is chosen (with the corresponding best t_j and p_j for that feature) that minimizes the number of errors on the raw training data set, with all training examples weighted equally ($w_i = 1/N \ \forall i$):

$$f_j = \arg \min_{f_j, t_j, p_j} \sum_{i=1}^{N} \frac{|h_j(x_i) - y_i|}{N}. \quad (2)$$

This weak classifier h_j is given a number of votes α_j that reflect how well it classified the training data. Then the training samples are re=weighted to give samples that were misclassified by the first weak classifier a higher weight w_i':

$$w_i' = \frac{w_i e^{-y_i \alpha_j h_j(x_i)}}{\sum_m w_m e^{-y_m \alpha_j h_j(x_m)}} \quad (3)$$

Next a new weak classifier is found to minimize the error on the re-weighted training sets. This process is repeated for a fixed total number of weak classifiers. By re-weighting the examples to give more weight to training samples that were

TABLE I

FEATURES USED FOR SUBMAP CLASSIFIERS.

1.Mean and standard deviation of the difference between consecutive laser range readings
2.Number of gaps (difference between consecutive range readings > threshold)
3.Mean, standard deviation, skew and kurtosis of laser range readings
4.The max range - min range, and the angle between the max range and min range
5.Area, perimeter, area convexity, perimeter convexity
6.Circularity, compactness
7.Seven moment invariant features
8.Number of readings at max range (all other measurements excludes max range readings)
9.λ_1/λ_2 where λ_1, λ_2 are the first two eigenvalues associated with the first two principle component analysis vectors (rotation invariant estimate of aspect ratio)
10.Mean distance to centroid
11.Number of second derivative sign flips (measure of bumpiness)
12.Probability distribution over the distance between two randomly selected points on boundary of submap formed by computed (x,y) pairs from (r,θ) readings

incorrectly classified, the weak classifiers correctly classify different samples, and the overall strong classifier improves.

We use an extension of this approach called AdaBoost.M2 [2] which handles multi-class classification by decomposing a D-class classification task into D separate 1-vs-all binary classification tasks. We selected AdaBoost.M2 due to its simplicity and good performance on our task.

B. Applying AdaBoost: Feature Selection

In our task we wish to classify laser scans into regions produced by spectral clustering using AdaBoost.M2. To build a classifier using AdaBoost we must select a set of features based on the laser scan readings to use to perform classification. In this application we extract a set of rotation invariant geometric features from each laser scan reading: rotation invariant features are critical to ensure that the robot can recognize previously visited locations from new orientations. See Table I for a list of all the features used.

These features are then used to create D 1-vs-all classifiers. A new laser scan is labeled with the cluster (submap) that corresponds to the classifier giving the highest votes to the scan (i.e., the classifier is most sure it has correctly classified this point). Therefore, classifiers localize the robot to a certain submap. Standard metric map localization techniques can be used to localize the robot within the given submap.

C. Loop Closure

Given the trained submap classifiers, the robot must be able to use the classifiers to determine when it has re-entered map. Loop closure is not considered until there are at least two submaps (and associated classifiers c_1, c_2 for each submap). Recall that each submap i is associated with a set of training data points consisting of a robot pose and laser range measurements x_i. When a third submap is initialized, both of the existing submap classifiers are run on the new training dataset x_3. A number is then computed which reflects how many of the new training data points are considered by the existing classifiers to be part of the existing submaps:

$$fc(co, x_3) = \frac{N_{3o}}{|x_3|} \tag{4}$$

where c_o is the existing classifier of one of the submaps, and N_{3o} is the number of new training datapoints x_3 classified by c_o as being part of submap o. When fc is large, then existing classifier c_o is voting strongly that the new submap is in fact previous submap o.

In addition, a new classifier c_3 is trained for the new submap x_3 against each existing submap (x_1 and then separately x_2), and c_3 is tested on a submap that was *not* used in training (x_2 and then x_3 respectively). The fraction of points claimed is again computed using equation 4. This fraction is large when the new classifier c_i votes strongly that the previous submap o is the same as the new submap.

To compute the similarity of two submaps, we use

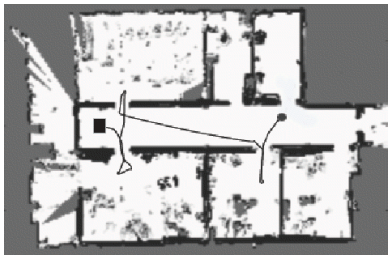
$$\xi(o, n) = fc(c_o, x_i) fc(c_i, x_o) \tag{5}$$

where n is a new submap, c_i is our new classifier, o is an existing submap and ξ is our measure of similarity between the two submaps. Essentially, we are computing the fraction of the other submap's points the first submap classifier "claims" as its own, and vice versa to determine how much the classifiers overlap. This number ξ will be somewhere between 0 and 1: if it is high, then it is likely that the "new" submap is in fact the same as an older submap, and instead of initializing a new submap, we should merge it into an existing submap, thereby closing the loop. This process is repeated for each new submap, so at each new region, the robot is effectively checking whether it is now in a pre-existing location, and if it is, closes the loop. Experimentally, we determined that an appropriate threshold for determining that two submaps were the same was $\xi = .175$. This may seem low, but in fact with several clusters the overlap claimed between them quickly drops to very low numbers, and this threshold was found to provide good results.

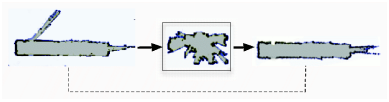
Note that in comparison to many other approaches, the ability to detect a loop does not automatically degrade if the robot has traveled a long distance since the last visit of a location: rather the ability to detect a location depends solely on being able to uniquely identify the regions (submaps) output through the spectral clustering technique. Therefore this technique is unlikely to work well in completely homogeneous environments, in which all rooms (regions) look the same. However, in practice there are many environments of interest that will contain uniquely distinguishable regions, including many office and residential buildings.

IV. RESULTS

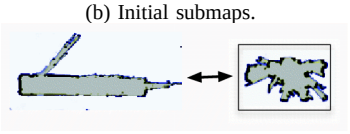
We have tested our complete map building algorithm on two separate environments where we simulated laser and pose readings of a robot. The first, the Freiburg dataset, was a good check to ensure that our technique could repeatedly close small loops, as the robot trajectory here repeatedly returned to the same corridor in between visiting different rooms. The second dataset, the Stata building, involved closing a larger loop as the robot took a tour of the third floor of that building. In both cases we successfully initialized submaps and closed loops. In order to evaluate the use of these representations, we evaluated how well the robot could identify its current submap based on a single laser scan reading (this is similar to the kidnapped robot problem, and in our case, is effectively evaluating the test set error of



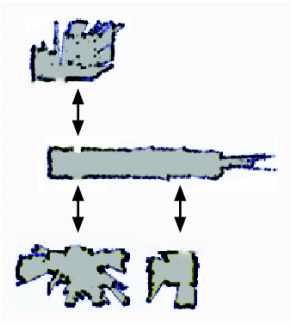
(a) Ground truth map.



(b) Initial submaps.

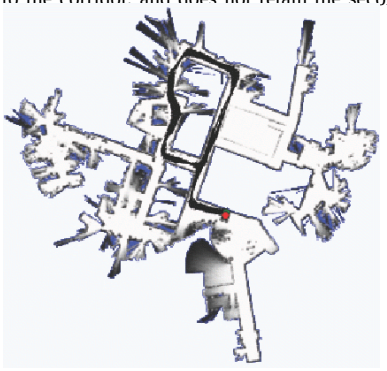


(c) Loop closing.

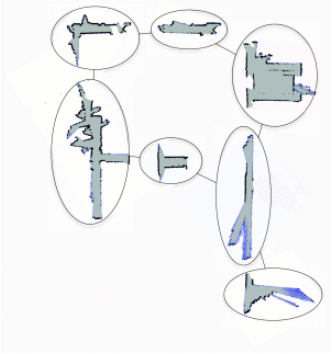


(d) Final topological map.

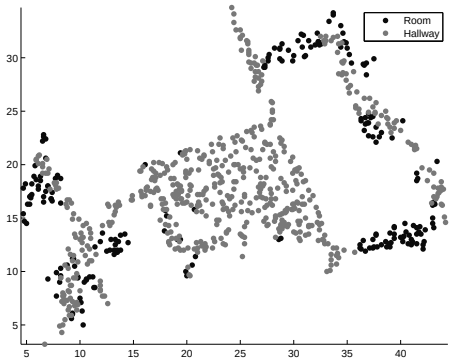
Fig. 4. Incremental Spectral Clustering and Loop Closing: Freiburg. (b) The submaps before loop closing (c) The classifier identifies that the robot has returned to the corridor, and does not retain the second corridor submap. (d) The final topological map.



(a) Map and robot trajectory



(b) Topological-metric map



(c) Room-hallway classifier segmentation.

Fig. 5. (a) The ground-truth metric map of the Stata building at MIT, and the robot trajectory during exploration. (b) The learned topological map. c) Non-optimized room-hallway classifier does not generalize well to segmenting Stata.

our submap classifiers) and also classified the submap of the robot as it again traversed the Stata dataset.

A. Online Partitioning and automatic loop closing

Figure 4a shows the trajectory of the robot as it goes through the Freiburg dataset. The black square indicates the start of the trajectory and the black circle indicates the end of the trajectory. Figure 4b shows the first three submaps created as the robot passes from the hallway into a room and then back into the hallway. This is successfully detected as a loop closure and the map is updated to only include two submaps (see Figure 4c). Figure 4d shows the final map representation. There are three two-way arrows in the topological map, successfully representing the robot's three returns to the central corridor.

Figures 5a&b show the simulated trajectory of the robot in the Stata environment and the associated topological-metric map constructed when the robot closes the loop. During a second trajectory around the loop, new submaps continued to be initialized and then merged with prior existing submaps. Occasionally an overlapping submap was created because it did not overlap sufficiently with the existing submaps. In the future we hope to explore whether using the topological information of the map can lead to the identification and thereby, elimination, of duplicate submaps.

TABLE II
LOCALIZATION ERROR RESULTS

	Training Set		Test Set	
	Recall	Precision	Recall	Precision
Freiburg dataset	0.9728	0.9818	0.9089	0.9212
Stata dataset	0.9689	0.9739	0.8300	0.8458

B. Localization

In order to evaluate how well our algorithm can be used to localize the robot to a particular submap after the map creation stage is finished, we divided up the Freiburg and Stata datasets each into a training set (80% of the original dataset) that was used to produce the original map (and associated classifiers) and a test set (20% of the original dataset) to evaluate on. The test set is like placing the robot at an unknown location, and asking it to predict its submap location from a single laser scan of its environment. We evaluated the predictions using precision and recall[2]. In order to localize well we desire high recall and precision rates. The results averaged over all submaps are presented in Table II. The high rates of recall and precision on the test sets indicate this method is likely to correctly localize the robot to a particular submap for new datapoints. To get a qualitative sense of the how well the robot is doing, we also ran the robot through a new trajectory on the Freiburg dataset and classified its location at each time step. Figure 6 shows the robot trajectory. Localization does a good job on the whole:

[2]For all test examples associated with a particular submap i, recall is the percent of the time the classifiers labeled those test examples correctly as submap i. Precision is of all test examples the classifier labeled as submap i, what percentage of the time was the test example actually from i. A classifier could achieve perfect recall for a particular submap i by labeling all test examples as i but would achieve low precision because by incorrectly label all other submap test examples as i.

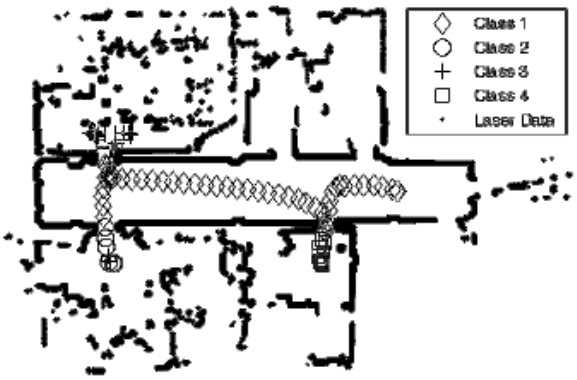


Fig. 6. Localization on a new trajectory in the Freiburg dataset.

the figure shows how the predicted submap changes as the robot goes in and out of the different regions that have been previously identified (see Figure 4).

The results we present here correspond to raw classification rates from a single scan associated with a single robot pose, with no constraints on consistency between consecutive robot poses. In the future, to improve localization we could include temporal information through methods such as using a Hidden Markov Model to help ensure smoothness. Another way to improve localization would be to utilize the connectivity information encoded in the topological map.

V. Related Work

In the interest of brevity we will only mention here a few of the most related pieces of research to the current work. The Atlas framework [1] is a mapping algorithm that produces hybrid metric-topological maps and takes a principled probabilistic approach to localization under uncertainty. Their method for submap matching relies on matching geometric features that do not tend to occur multiple times within a single map whereas our approach instead focuses on features that serve to uniquely distinguish a submap. Kuipers et al.[4], [7] incrementally build a topological map using Voronoi diagrams, looking for "constrictions" in the medial axis graph to define the local topologies. Thrun [10] constructs a Voronoi diagram based on a full metric map, inducing submap cuts at the points that minimize clearance.

Our loop closure approach relates to recent work by Newman et al. [8] that performs 3D SLAM in large outdoor environments . To close loops they search for maximum alignment between two sequences of camera images: images are described by the presence of a fixed set of image features. Using temporal information is likely to improve the accuracy of loop closure compared to non-temporal methods such as the one we present here and the authors employ a rigorous probabilistic method for deciding if an alignment is sufficient to be considered a loop closure. However, unlike our work, by having a fixed set of features, the authors have to handle explicitly the problem of "themes"– features that occur consistently over a short period of time, such as bricks, that do not help uniquely identify a place for loop closure.

While these approaches may not work well in an environment consisting of identical rooms and hallways, we argue that most real-world environments have local modifications (such as sofas or desks that move relatively little) that generate a unique signature that can be classified reliably.

There are clearly myriad possible approaches to extracting a partitioning of a map. We were inspired by existing supervised labeling techniques [3], [6] for semantic locations (such as room or hallway), but our experience was that existing learned semantic classifiers did not generalize well to new environments that are not naturally suited towards the typical notions of rooms and hallways. As an example, we trained an AdaBoost classifier to recognize rooms and hallways based on laser scan information on a hand-labeled version of the Freiburg dataset. We then ran this classifier on a set of laser scans extracted from a map of the third floor Stata building. The resulting room/hallway classification can be seen in Figure 5(c). Though care should be taken in interpreting these results as the room/hallway classifier was not optimized, comparing Figure 2 and Figure 5(c) suggests that room/hallway classifiers do not perform as well as spectral clustering for providing a segmentation of the environment into unique components.

VI. Conclusions

In this paper we have demonstrated the viability of using spectral clustering and classification techniques to automatically construct a hybrid topological-metric map representation of an environment.

References

[1] Mike Bosse, Paul Newman, John Leonard, and Seth Teller. Slam in large-scale cyclic environments using the atlas framework. *Internationl Journal of Robotics Research*, 23(12):1113–1139, 2004.

[2] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pages 23–37, 1995.

[3] Stephen Friedman, Hanna Pasula, and Dieter Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *IJCAI*, pages 2109–2114, 2007.

[4] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *ICRA*, 2004.

[5] J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE International Workshop on Intelligent Robots and Systems*, 1991.

[6] O. Martínez-Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *ICRA*, 2005.

[7] J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In *ICRA*, 2004.

[8] Paul Newman, David Cole, and Kin Ho. Outdoor slam using visual appearance and laser ranging. In *ICRA*, 2006.

[9] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[10] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, 1998.

[11] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[12] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, 2004.