Incremental Spectral Clustering and Its Application To Topological Mapping

Christoffer Valgren*, Tom Duckett† and Achim Lilienthal*

* Applied Autonomous Sensor Systems
Örebro University
SE-701 82 Örebro, Sweden
christoffer.wahlgren@tech.oru.se,
achim@lilienthals.de

† Department of Computing and Informatics
University of Lincoln
Brayford Pool, Lincoln, LN6 7TS, United Kingdom
tduckett@lincoln.ac.uk

***Abstract*— This paper presents a novel use of spectral clustering algorithms to support cases where the entries in the affinity matrix are costly to compute. The method is incremental – the spectral clustering algorithm is applied to the affinity matrix after each row/column is added – which makes it possible to inspect the clusters as new data points are added. The method is well suited to the problem of appearance-based, on-line topological mapping for mobile robots. In this problem domain, we show that we can reduce environment-dependent parameters of the clustering algorithm to just a single, intuitive parameter. Experimental results in large outdoor and indoor environments show that we can close loops correctly by computing only a fraction of the entries in the affinity matrix. The accompanying video clip shows how an example map is produced by the algorithm.**

I. Introduction

Spectral clustering methods have become increasingly popular. This family of algorithms have been proven successful in a number of problem domains, such as computer vision [1], [2], speech recognition [3], and classification of biological data [4], [5]. Their primary strength is that they successfully can cluster data where other well-known methods (such as k-means) cannot be applied or fail. Spectral clustering does not require that the data can be represented as coordinates in Euclidean n-space – it is sufficient that a similarity measure between the points can be computed. Common to all spectral clustering algorithms is that they take as input an affinity matrix, which describes the similarity between the data points. Similarity is usually expressed by Euclidean distance, but it can equally well be described by some other measure.

In appearance-based topological mapping problems, the environment at different positions is captured by sensors into "snapshots". Such a snapshot is usually a very high-dimensional descriptor (image [6], "fingerprint" [7], etc.) of the environment, and it is therefore usually futile to directly apply a clustering algorithm to the collection of snapshots. However, if it is possible to compute a measure of similarity between snapshots, we can compute an affinity matrix and then apply a spectral clustering method to extract the nodes of the topological map.

There are at least two serious drawbacks with this approach. First, in most spectral clustering algorithms, the number of nodes has to be set by hand. It is possible to handle this by simply iterating with an increasing number of nodes, and halting the iteration when the resulting clustering is "good enough". In contrast, the number of clusters in our algorithm follows from a parameter that might be selected in a natural way directly from the data. Secondly, computing the affinity matrix can be costly, even with efficient algorithms to compute each entry. This is noticeable in particular when the data set becomes large. It would be preferable if we did not have to compute all entries of the affinity matrix, and yet could apply a spectral clustering algorithm.

In this paper we present a general-purpose, incremental spectral clustering algorithm that addresses the issues above. Because the affinity matrix is not completely evaluated, the method will be approximate. Nevertheless, it produces very good results in our special area of interest: on-line topological mapping by a mobile robot. We first describe the algorithm and show the generality of the method by applying it to toy examples. Finally, we show how the method can produce a large, appearance-based topological map.

II. Related Work

Spectral clustering comes in a variety of flavors. In this paper, we will exclusively use the method proposed by Ng, Jordan and Weiss [8]. A great overview of the different methods is available in [9].

There are many approaches to appearance-based topological mapping. Gaspar et al. [6] applied principal component analysis to condense a large data set of panoramic images into a smaller set of eigenimages that was used for localization. Tapus and Siegwart [7] extract "fingerprints" from panoramic images and laser scan data, and add nodes to the topological map whenever an important change in the environment (or, rather, to the fingerprint) occurs.

Mulligan and Grudic [10] used spectral clustering on image data to produce topological maps for a mobile robot. Zivkovic et al. [11] produced an appearance-based hierarchical map using spectral clustering in order to obtain an approximate solution to the normalized cut problem.

III. Incremental Spectral Clustering

A. Spectral clustering

To produce the examples in this paper, we use the clustering algorithm by Ng, Jordan and Weiss [8]. We also apply the modification suggested by Verma and Meila [9] in order to achieve greater numerical stability (Algorithm 1).

Algorithm 1 Modified NJW algorithm

```
function CLUSTER(affinity matrix $\mathbf{A} \in \mathbf{R}^{n \times n}$, number of clusters $k$)
    $\mathbf{A}_{ii} \leftarrow 0$
    $\mathbf{D}_{ii} \leftarrow$sum of row $i$ of $\mathbf{A}$; $\mathbf{D}$ is a diagonal matrix
    $\mathbf{X} \leftarrow k$ largest generalized eigenvectors of $\mathbf{Av} = \lambda \mathbf{Dv}$
    $\mathbf{Y} \leftarrow$normalized rows of $\mathbf{X}$
    $C \leftarrow k$ clusters of rows in $\mathbf{Y}$, using k-means or similar
    return $C$
end function
```

As input to the spectral clustering algorithm, we assume an affinity matrix $\mathbf{A}$. This matrix is computed using the distances between the points in the data set S influenced by the scaling parameter σ:

$$\mathbf{A}_{ij} = \exp\left(-\left(\frac{d(s_i, s_j)^2}{2\sigma^2}\right)\right) \tag{1}$$

where $d(s_i, s_j)$ denotes the distance between points s_i and s_j.

The value of the scaling parameter σ is very important, and alone determines how similarity depends on distance. If σ is set too high, compared to the true scale of the problem, most points will appear similar. If σ is set too low, the similarity between even close points will be low. Both of these scenarios imply suboptimal clustering. Computing the value of σ is, however, out of the scope of this paper (but see for example [12]). For our purposes, we find that σ naturally follows from another parameter (section V-B).

The NJW algorithm further requires as input the number of clusters k, since the algorithm utilizes the k-means clustering algorithm to produce the clusters.

B. Incremental spectral clustering

The incremental spectral clustering method starts with an empty data set A and thus an empty affinity matrix $\mathbf{A}$. For each data point $s_i \in S$ that is added to the data set A, the algorithm iteratively estimates a *cluster representative* for each cluster. The cluster representative is the data point that is most similar to all other points.[1]

We require that the cluster representative is not too far away from any point in the cluster. If it is, the number of clusters must be increased and a new clustering is performed. We call the smallest allowed distance the *similarity threshold*.

Whenever the number of clusters is increased (and when each cluster has a suitable cluster representative), the entries in the affinity matrix that have been assigned to a cluster are replaced by a single cluster representative. The original contents of the cluster are stored for future use in computation of a new cluster representative, if it becomes necessary. The affinity matrix is thus shrunk to a smaller size. The process then continues with the next data point.

Incremental spectral clustering is summarized in Algorithm 2. It requires two external functions. The function *sim* computes the affinity between data points, i.e. it would typically compute the affinity between one or more data points according to equation 1. The function *CLUSTER* computes k clusters from the current affinity matrix $\mathbf{A}$.

The method presented in this paper is not restricted to the modified NJW algorithm. Any spectral clustering algorithm that takes an affinity matrix and a number of clusters as input (i.e. it implements the *CLUSTER* function) could be used without major modifications to the method.

Algorithm 2 Incremental spectral clustering

```
$\mathbf{A}$ is empty                        ▷ $\mathbf{A}$ is the current affinity matrix
$A \leftarrow \emptyset$                             ▷ $A$ is the current set
$k \leftarrow 1$                             ▷ $k$ is the number of nodes
for all $s_i \in S$ do
    $A \leftarrow A \cup s_i$
    $a \leftarrow \text{sim}(s_i, A)$                     ▷ $a$ is a row vector
    $\mathbf{A} \leftarrow \begin{bmatrix} \mathbf{A} & a^T \\ a & 0 \end{bmatrix}$
    $new\_node \leftarrow false$

    ▷ $\mathbf{C}_n$ is an $i \times j$ matrix
    ▷ $R_n$ is the point in $C_n$ most similar to all
    ▷ other points in $C_n$
    ▷ $N_n$ is the similarity value for cluster $R_n$

    $N_n \leftarrow 0$
    while $min(N_n) <$ similarity threshold do
        $C \leftarrow \text{CLUSTER}(\mathbf{A}, k)$
        for all $C_n \in C$ do
            $\mathbf{C}_n \leftarrow \text{sim}(C_n, C_n)$
            $R_n = \text{argmax}_{i \in C_n}(\min_{j \in C_n}(\mathbf{C}_n))$
            $N_n = \max_{i \in C_n}(\min_{j \in C_n}(\mathbf{C}_n))$
        end for
        if $min(N_n) <$ similarity threshold then
            $new\_node \leftarrow true$
            $k \leftarrow k + 1$
        end if
    end while
    if $new\_node$ then
        $A \leftarrow R$
        $\mathbf{A} \leftarrow \text{sim}(A, A)$
    end if
end for
```

C. Tweaking the algorithm

There are some issues not addressed with Algorithm 2:

- While the algorithm can successfully handle cases where two clusters are merged, there is no functionality

[1]If the data points *do* have a representation in Euclidean space, the cluster representative would be the point closest to the cluster centroid.

for splitting clusters. This can be easily introduced by observing a newly created cluster C_n that did not pass the similarity threshold, and performing a separate spectral clustering on the matrix $\mathbf{C}_n$. The resulting clusters are then written back to A.

- The number of clusters k *always* increases. Because the method is iterative, the resulting number of clusters is usually higher than necessary - this is especially true when the data points arrive randomly (instead of in cluster order). Improved clustering can be achieved by regularly or randomly decreasing the number of clusters k by 1.

The implementation of the algorithm in this paper has these additional features. The decreasing of the number of clusters was done with a probability of 0.1; this value was found to be a good trade-off between speed and having too many clusters.

D. When to use incremental spectral clustering

If the data points cannot readily be represented by a coordinate in n-dimensional space, but it is possible to compute a similarity measure, spectral clustering is a good option. If the entries in the matrix are costly to compute, and an approximate result can be accepted, the incremental spectral clustering method is significantly faster. Further, if the data should be clustered on-line (i.e. clusters should be formed as new data points are added to the data set), the incremental spectral clustering method is preferred for speed reasons.[2] Finally, if the number of clusters of the data is unknown, the incremental clustering algorithm is preferrable since it automatically determines the number of clusters by using the similarity threshold.

E. Results on synthetic data

Some examples on synthetic data illustrate the strengths and weaknesses of the algorithm. The examples in Fig. 1 and Fig. 2 illustrate the importance of the similarity threshold.[3] The data points represent 6 coordinates at $(-0.5, 0.0)$, $(0.5, 0.0)$, $(-0.5, 1.0)$, $(0.5, 1.0)$, $(-0.5, -1.0)$, $(0.5, -1.0)$ with Gaussian noise added. With knowledge of the noise distribution, it is possible to set the similarity threshold so that the points are almost perfectly classified. Fig. 1 also illustrates that it is possible to achieve satisfactory clustering without computing the entire affinity matrix; only 69% of the total number of entries in the affinity matrix were evaluated.

Fig. 3 shows an example which the NJW algorithm, given a correct value of sigma and the number of clusters, can cluster into two intuitive clusters - a center cluster and a ring surrounding it. The incremental spectral clustering successfully finds the center cluster, but must split the ring up into several smaller parts. This is because we do not allow the points within a cluster to be too dissimilar. In this case, if we were to set the similarity threshold to a too high value, the algorithm would not be able to differentiate between the center cluster and the surrounding ring. The number of clusters in this case is thus directly determined by the similarity threshold.

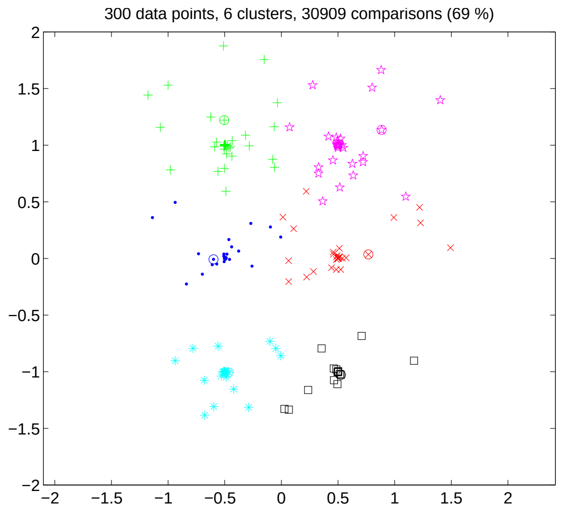

Fig. 1. Incremental spectral clustering. 6 clusters with Gaussian noise added. Similarity threshold is based on the true cluster distance 1.0.

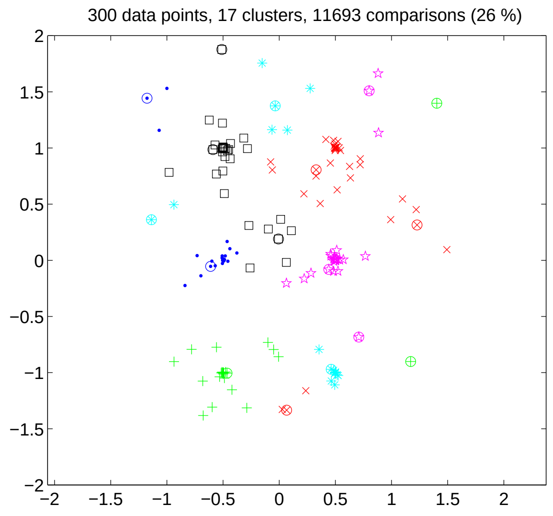

Fig. 2. Incremental spectral clustering. Same data points as in 1. Here, the similarity threshold has been halved, increasing the number of clusters but increasing performance.

F. Performance

The incremental spectral clustering algorithm applies the spectral clustering algorithm multiple times to the data, and also introduces some additional overhead in searching for the cluster representatives, splitting and merging clusters, etc. However, this does not mean that incremental spectral clustering is slower than normal spectral clustering.

[2]Consider a naive on-line approach using normal spectral clustering, where clustering is performed after each data point. The affinity matrix will grow as n^2, which means that the computation time will soon be larger than for the incremental spectral clustering algorithm (which tries to limit the size of the affinity matrix). This is true even if the cost of computing the entries in the affinity matrix is ignored.

[3]These examples are for illustration only. Simple clustering tasks such as these would usually be better tackled by a simpler clustering algorithm.

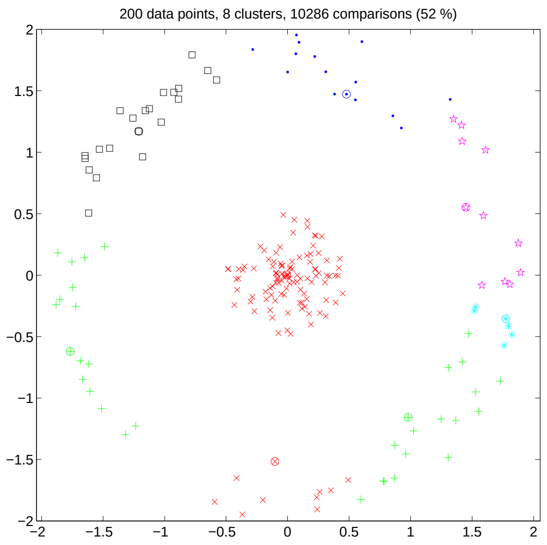

Fig. 3. Incremental spectral clustering. Concave areas can be handled, but the clustering differs from what a human would intuitively choose.

Consider the data set shown in Fig. 1. An unoptimized Matlab implementation of the incremental spectral clustering algorithm takes about 70 seconds for the entire data set. The time to perform spectral clustering on the entire affinity matrix is about 0.5 seconds on the same machine. However, the incremental spectral clustering algorithm needs only 69% of the affinity matrix to obtain the clusters. This means that we (in this particular case) reach the break-even point at about 5 milliseconds per entry in the affinity matrix. If the evaluation of an average entry in the affinity matrix takes more time than this, then incremental spectral clustering is faster than spectral clustering.

The performance of incremental spectral clustering is very dependent on the distribution of the points, and thus it is very difficult to estimate the performance of the algorithm. Larger matrices obviously take longer to handle in general, which would imply that data sets with large clusters containing many points should be faster to process. This is not necessarily true, however, because the search for the cluster representative means that more entries of the affinity matrix have to be evaluated.

IV. Topological Mapping Using Vision

As mentioned in the introduction, the reason for developing the incremental spectral clustering algorithm was originally to produce topological maps using vision. With the incremental spectral clustering algorithm in place, the key to produce correct topological maps is to compute correct affinities within a set of images. In our case, we have a sequence of panoramic images obtained from a mobile robot. We use local features extracted from the images to compute the affinity matrix.

A. Matching images using local features

The images are acquired by an omnidirectional camera, consisting of a curved mirror lens mounted above a digital camera. Local features are extracted from the images. We use SIFT, the Scale-Invariant Feature Transform, which was first presented by David Lowe in 1999 [13]. The main characteristic of SIFT is that it uses a feature description that is invariant to scaling and rotation. It is also partially invariant to changes in illumination and camera location.

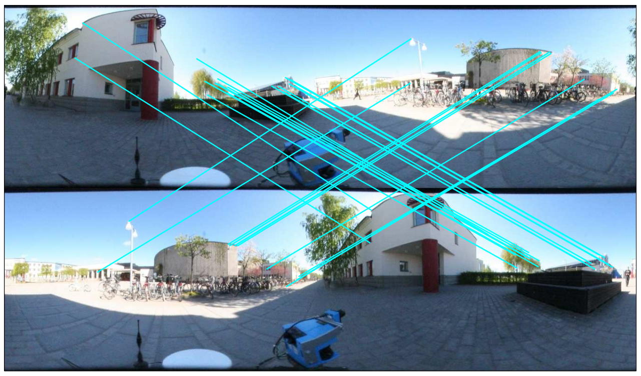

Fig. 4. Example of feature matching on typical images used in this paper.

The local features extracted from one image can be matched to features from another image. Using local features for image comparison in this way has several advantages over methods that use global features for the comparison: it is less sensitive to occlusion and changing environments [14], and it is possible to directly use the number of feature matches as a measure of image similarity. There is, however, always a risk that some features will be wrongly matched. We set a threshold N_{min} for the minimum number of local feature matches before two images are said to match each other.

The feature matching algorithm calculates the Euclidean distance between each feature in image i and all the features in image j. A potential match is found if the smallest distance is smaller than 60% of the second smallest distance. Note that a feature f_i in image i may match feature f_j in image j, without f_j matching f_i. To reduce the chance of false matches, we require reciprocal matching, which means that the features must mutually match each other.

An example of feature matching is shown in Fig. 4. Note that matches for features corresponding to the robot have been removed.

B. The distance measure

The number of matches $M(i,j)$ between two images i and j can be used to compute the corresponding entry in the affinity matrix. We use the following simple formula to compute the distance measure[4] $d(i,j)$ that is used in the computation of the affinity (1):

$$d(i,j) = \frac{1}{M(i,j)+1} \tag{2}$$

[4]We note that $d(i,j)$ is not a true distance measure in the geometric sense; it does not fulfil $d(i,i) = 0$, neither does it fulfil the triangle inequality. However, it is difficult to construct a true distance metric for image similarity, and we have chosen the easy route here. The resulting affinity matrix will still be useful.

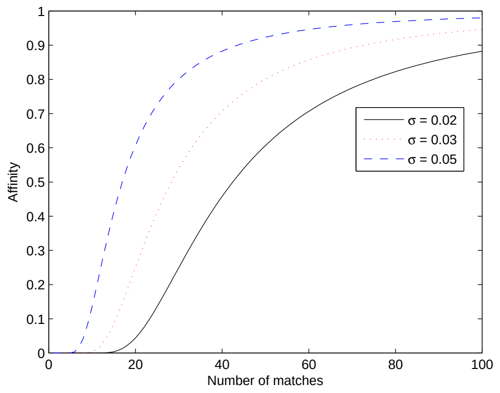

Fig. 5. The affinity computed as a function of σ and the number of matches.

V. Experiment

A. Experimental setup

The topological maps shown in Fig. 7 and 8 were produced from two sequences of images, acquired by an ActivMedia P3-AT robot fitted with a standard consumer-grade SLR digital camera (Canon EOS350D, 8 megapixels) with a curved mirror from 0-360.com. This camera-mirror combination produces omnidirectional images that can be unwrapped into high-resolution *spherical* images by a simple polar-to-Cartesian conversion. In what follows, we ignore the geometrical distortion in the spherical image.

The robot was teleoperated around a combined indoor and outdoor environment. The images were acquired by remote control, at semi-regular intervals. The positions of the images were determined by hand. However, the positions do not have any influence on the resulting topological map and are used only for visualization.

The images were unwrapped and scaled down to about 1300×400 pixels. Features were extracted from each image (on average, about 2000) and the number of matches between images was calculated.

B. Similarity threshold and the scaling parameter σ

The choice of the similarity threshold directly affects the clustering, because it determines the smallest similarity (or rather, largest dissimilarity) that we are prepared to accept in each node.

Usually, the similarity threshold comes from experience. Using SIFT features extracted from images, it is possible to inspect the images and manually select the correct number of SIFT matches for a “place”. One might expect the value to vary largely over a large data set. However, as long as the environment is not extreme in some way (i.e. it is completely featureless or contains repeating patterns), one value is usually sufficient for the entire data set [15].

We know that for our particular equipment, a good choice for the value of the minimum number of feature matches N_{min} is 15. The value varies with the resolution of the images and the environment to be mapped; here, however, the same value has been used for all data sets.

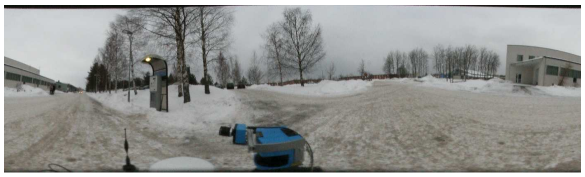

Fig. 6. Example of image from winter data set.

The similarity threshold is based on this value, as well as the scaling parameter σ. The affinity as a function of the number of matches $M(i, j)$ and σ is shown in Fig. 5. A good choice of σ is one where the affinity approaches zero when the number of matches approaches N_{min}. This ensures that spectral clustering does not unnecessarily create clusters that will have a too low maximum similarity value N_n (see Algorithm 2).

In our case, a choice of $N_{min} = 15$ implies $\sigma \approx 0.03$.

C. Results

Fig. 7 and 8 show two topological maps based on data sets acquired in winter and summer, respectively. The winter map exhibits some false links. This is probably due to poor feature matching in the snow covered environment. The summer map, which covers partly the same area as the winter set, is reproduced with correct topology. Fig. 9 illustrates the computation time requirement for each image acquired, excluding feature computation and matching. Matching two images from our data set took at least 1 second; the overhead introduced by incremental spectral clustering is thus negligible compared to the matching time. Because it was only necessary to evaluate roughly half of the entries in the full affinity matrix, the total computation time was nearly halved (compared to spectral clustering) for these maps.

VI. Conclusion and future work

The algorithm outlined in this paper is incremental, which makes it useful for mobile robots that should update their map on-line. The algorithm supports cluster merging and splitting, which means that the algorithm successfully can close loops. Also, there is no need to compute the similarities between the current image and all previous images, which implies less computation time in those cases where the similarity measure is costly to compute.

We have shown how the incremental spectral clustering algorithm can use the power of spectral clustering to obtain clusters without evaluating the entire affinity matrix. The number of clusters does not need to be set; instead a similarity threshold, which in many cases may be a more intuitive parameter, is introduced.

Note that all topological maps shown in this paper have been computed using appearance only. It is highly likely that the maps can be greatly improved by introducing additional sensor information. Odometry or other methods of determining position would seem to be ideal candidates, as distance measurements are easily transformed into affinity.

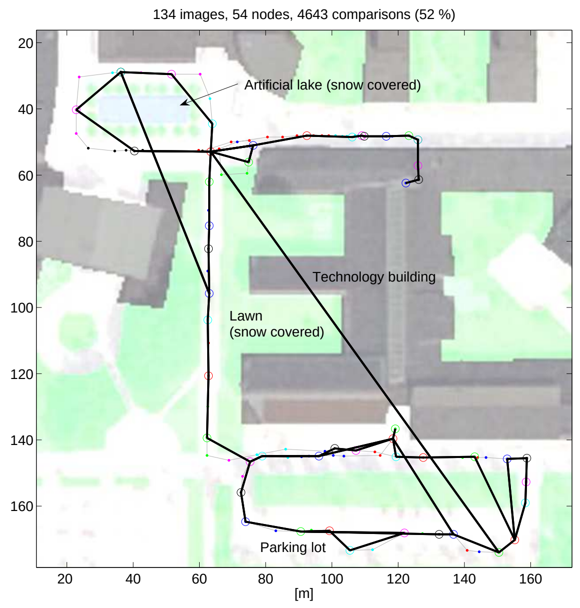
134 images, 54 nodes, 4643 comparisons (52 %)
Artificial lake (snow covered)
Technology building
Lawn
(snow covered)
Parking lot
[m]

Fig. 7. Resulting topological map from a data set acquired in winter (with snow). The false links in this data set are to be expected, as the images contained very few strong features (see Fig. 6 for a typical image). Total computation time was 160 seconds in Matlab on an AMD64-3500+, excluding feature computation and matching.

In the pipeline for future work is a more theoretical treatment, to more precisely highlight the cases where incremental spectral clustering out-performs other clustering algorithms. In addition, introduction of geometric constraints in the image matching (by using RANSAC, for example), will probably improve the resulting topological map.

VII. Acknowledgments

This work is partially supported by The Swedish Defence Material Administration.

References

[1] J. Park, H. Zha, and R. Kasturi, "Spectral clustering for robust motion segmentation," in *The 8th European Conference on Computer Vision*, 2004, pp. 390–401.

[2] H. Chang and D. Yeung, "Robust path-based spectral clustering with application to image segmentation," in *Proc. Intl. Conf. On Computer Vision*, 2005, pp. I: 278–285.

[3] F. R. Bach and M. I. Jordan, "Learning spectral clustering," UC Berkeley, Tech. Rep., 2003.

[4] W. Pentney and M. Meila, "Spectral clustering of biological sequence data," in *Proc. 25th Annual Conference of AAAI*, 2005.

[5] N. Speer, H. Fröhlich, C. Spieth, and A. Zell, "Functional grouping of genes using spectral clustering and gene ontology," in *Proc. IEEE Intl. Joint Conference on Neural Networks*, 2005, pp. 298–303.

[6] J. Gaspar, N. Winter, and J. Santos-Victor, "Vision-based navigation and environmental representations with an omni-directional camera." *IEEE Trans. Robotics and Automation*, vol. 16, pp. 890–898, 2000.

[7] A. Tapus and R. Siegwart, "Incremental robot mapping with fingerprints of places," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, 2005.

[8] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, 2001.

[9] D. Verma and M. Meila, "A comparison of spectral clustering algorithms," University of Washington, Tech. Rep. UW-CSE-03-05-01, 2003.

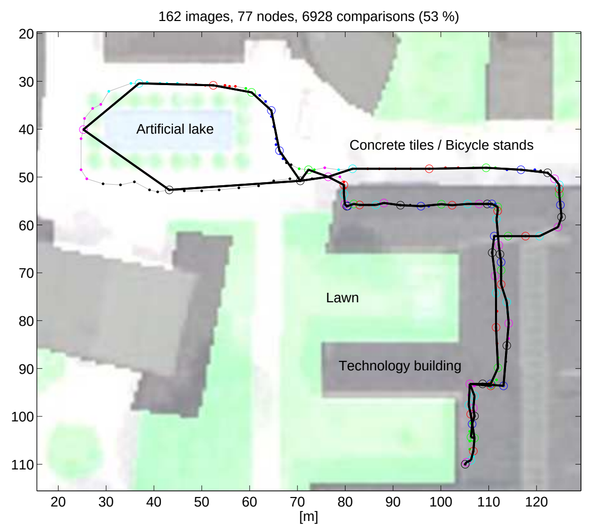
162 images, 77 nodes, 6928 comparisons (53 %)
Artificial lake
Concrete tiles / Bicycle stands
Lawn
Technology building
[m]

Fig. 8. Resulting topological map from a data set acquired in summer. Computation time was 140 seconds, excluding feature computation and matching (see Fig. 9).

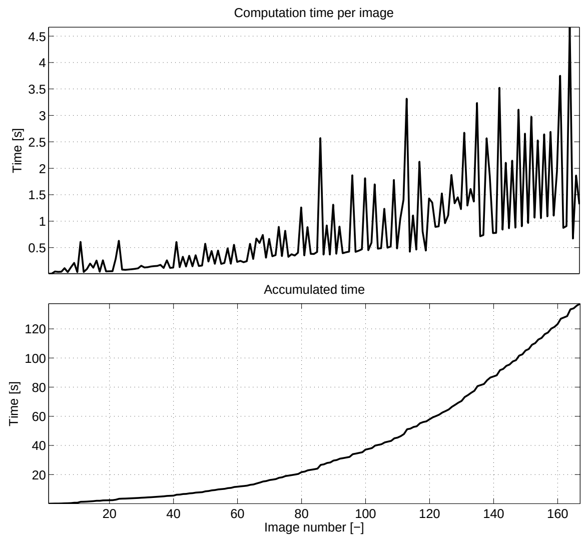

Fig. 9. Computation time per image and accumulated computation time for data set in Fig. 8, excluding feature computation and matching. The large deviation in computation time per image is due to that spectral clustering is not performed for every iteration, see Algorithm 2.

[10] J. Mulligan and G. Grudic, "Topological mapping with multiple visual manifolds," in *Proceedings of Robotics: Science and Systems*, 2005.

[11] Z. Zivkovic, B. Bakker, and B. Kröse, "Hierarchical map building and planning based on graph partitioning," in *Proc. IEEE Intl. Conf. Robotics and Automation*, 2006, pp. 803–809.

[12] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *NIPS*, 2004.

[13] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Computer Vision ICCV*, 1999.

[14] H. Andreasson, A. Treptow, and T. Duckett, "Localization for mobile robots using panoramic vision, local features and particle filter," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 2005)*, Barcelona, Spain, 2005, pp. 3348–3353.

[15] C. Valgren, A. Lilienthal, and T. Duckett, "Incremental topological mapping using omnidirectional vision," in *Proc. IEEE Int. Conf. On Intelligent Robots and Systems*, 2006, pp. 3441–3447.