Clustering Sensor Data for Terrain Identification using a Windowless Algorithm

Philippe Giguere
Centre for Intelligent Machines
McGill University
Montreal, Quebec, Canada H3A 2A7
Email: philg@cim.mcgill.ca

Gregory Dudek
Centre for Intelligent Machines
McGill University
Montreal, Quebec, Canada H3A 2A7
Email: dudek@cim.mcgill.ca

***Abstract*— In this paper we are interested in autonomous systems that can automatically develop terrain classifiers without human interaction or feedback. A key issue is clustering of sensor data from the same terrain. In this context, we present a novel off-line windowless clustering algorithm exploiting time-dependency between samples. In terrain coverage, sets of sensory measurements are returned that are spatially, and hence temporally correlated. Our algorithm works by finding a set of parameter values for a user-specified classifier that minimize a cost function. This cost function is related to change in classifier probability outputs over time. The main advantage over other existing methods is its ability to cluster data for fast-switching systems that either have high process or observation noise, or complex distributions that cannot be properly characterized within the average duration of a state. The algorithm was evaluated using three different classifiers (linear separator, mixture of Gaussians and *k*-NEAREST NEIGHBOR), over both synthetic data sets and mobile robot contact feedback sensor data, with success.**

I. INTRODUCTION

Identifying the local terrain properties has recently become a problem of increasing interest and relevance. This has been proposed with both non-contact sensors, as well as using tactile feedback. This is because terrain properties directly affect navigability, odometry and localization performance. As part of our research, we are interested at using simple internal sensors such as accelerometers and actuator feedback information to help discover and identify terrain type. Real terrains can vary widely, contact forces vary with locomotion strategy (or gait, for a legged vehicle), and are difficult to model analytically. Therefore, the problem seems well suited to statistical data-driven approaches.

We approach the problem using unsupervised learning (clustering) of samples which represent sequences of consecutive measurement from the robot as it traverses the terrain, perhaps moving from one terrain type to another. Since those signals are generated through a physical system interacting with a continuous or piece-wise continuous terrain, time-dependency will be present between consecutive samples. The clustering algorithm we are proposing in this paper explicitly exploits this time-dependency. It is a single-stage batch method, eliminating the need for a moving time-window. The algorithm has been developed for noisy systems (i.e., overlapping clusters), as well as for systems that change state frequently (e.g., a robot traversing different terrain types in quick succession).

The paper is organized as follow. In Section II, we present an overview related work on the subject, pointing out some limitations with these methods. Our algorithm is then described in Section III, with theoretical justifications. In Section IV-A, Section IV-B and Section IV-C, we evaluate the performance of the algorithm on synthetic data with a linear separator classifier, a mixture of Gaussians classifier and *k*-NEAREST NEIGHBOR classifier, respectively. We compare our algorithm with a window-based method in Section IV-D. We then show in Section V how applying this method on data collected from a mobile robot enables robust terrain discovery and identification. This is followed by Section VI where we further point at differences between this algorithm and others.

II. RELATED WORK

Other techniques have been developed to exploit time dependencies for segmenting time-series or clustering data points. In Pawelzik *et al.* [1], an approximate probabilistic data segmentation algorithm is proposed on the segmentation algorithm on the assumption that the probability of having a transition within a sub-sequence is negligible, given a low switching rate between generating modes. Kohlmorgen *et al.* [3] present a method that uses mixture coefficients to analyze time-series generated by drifting or switching dynamics. In a similar fashion to the work we present here, these coefficients are found by minimizing an objective function that includes a squared difference between temporally adjacent mixture coefficients. The main argument behind their choice is that "solutions with a simple temporal structure are more likely than those with frequent change", an assumption we also exploit. In Kohlmorgen *et al.* [2], they present another segmentation method for time series. It is based again on minimizing a cost function that relates to the number of segments in the time series (hence minimizing the number of transitions), as well as minimizing the representation error of the prototype probability density function (*pdf*) of those segments. The distance metric used to compare *pdf*s is based on a L_2-Norm. Their simplified cost function has been designed to be computed efficiently using dynamic programming.

Lenser *et al.* [4] present both an off-line and on-line algorithm to segment time-series of sensor data for a mobile robot to detect changes in lighting conditions. The algorithm

works by splitting the data into non-overlapping windows of fixed size, and then populating a tree structure such that similar regions are stored close to each other in the tree, forcing them to have common ancestor nodes. The tree is built leaf by leaf, resulting in an agglomerative hierarchical clustering. If the number of clusters is known, information in the tree structure can be used to group the data together and form clusters. The distance metric used to compare regions correspond to the absolute distance needed to move points from one distribution to match the other distribution.

For terrain identification using a vehicle, several techniques have been developed (Weiss *et al.* [5] [6], Brooks *et al.* [7], DuPont *et al.* [8], Sadhukan *et al.* [9]). Features are extracted from acceleration measurements (i.e., spectrum of acceleration, multiple moments, etc) and supervised learning is used to train a classifier, such as a support vector machine (Weiss *et al.* [5]) or a probabilistic neural network (DuPont *et al.* [8]). Another work by Lenser *et al.* [10], a non-parametric classifier for time-series is trained to identify states of a legged robot interacting with its environment. These techniques require part of the data to be manually labelled, and thus cannot be employed in the current context of unsupervised learning.

A. Limitations of Window-Based Clustering Algorithms

As long as a system is switching infrequently between states and the distributions are well separated, there will be enough data points within a window of time to properly describe these distributions. Algorithms such as Lenser *et al.* [4] or Kohlmorgen *et al.* [2] will be able to find a suitable *pdf* to describe the distributions or to detect changes, and the clustering or segmentation will be successful.

As the system switches state more frequently however, the maximum allowable size for a window will be reduced. This has to be done in order to keep the probability of having no transition in a window reasonably low. With this in mind, two particular cases become difficult:

- Noisy systems with closely-spaced distribution, resulting in significant overlap. According to Srebro *et al.* [11], the difficulty in properly clustering data from two normally distributed classes with means μ_a, μ_b, and identical standard deviation $\sigma_a = \sigma_b$ is related to the relative distance $\frac{|\mu_a - \mu_b|}{\sigma_a}$.
- Complex distribution that cannot be characterized within a small window size. In this case, there is enough information within a time-window to classify samples, but if the distributions are unknown, there is not enough information to decide whether the samples belong to the same cluster.

III. Approach

The algorithm works as follow. Given that we have:

- a data set $\vec{X}$ of T time-samples of feature vectors $\vec{x}_i$, $\vec{X} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$ generated by a Markovian process with N_c states, with probability of exiting any state less than 50 *percent*,
- the sampled features $\vec{x}_i \in \vec{X}$ are representative, implying that *locally*, the distance between two samples is related to the probability that they belong to the same class,
- a classifier with parameters $\vec{\theta}$ used to estimate the probability $p(c_i|\vec{x}_t, \vec{\theta})$ that sample $\vec{x}_t$ belongs to class $c_i \subset C$, $|C| = N_c$,
- a classifier exploiting distance between data points $\vec{x}_i \in \vec{X}$ to compute probability estimates,
- a set of parameters $\vec{\theta}$ that is able to classify the data set $\vec{X}$ reasonably well.

The algorithm searches for the parameters $\vec{\theta}$ that minimize:

$$\arg\min_{\vec{\theta}} \sum_{i=1}^{N_c} \frac{\sum_{t=1}^{T-1} \left(p(c_i|\vec{x}_{t+1}, \vec{\theta}) - p(c_i|\vec{x}_t, \vec{\theta})\right)^2}{var(p(c_i|\vec{X}, \vec{\theta}))^2} \quad (1)$$

In our context of terrain identification, $\vec{X}$ represents a time-series of vehicle sensory information affected by the terrain e.g., acceleration measurements.

Roughly speaking, the cost in Eq. 1 tries to strike a balance between minimizing variations of classifier posterior probabilities *over time*, while simultaneously maintaining a wide distribution of posterior probabilities ($var(p(c_i|\vec{X}, \vec{\theta}))$). This is all normalized by the number of samples in each class (approximated as $var(p(c_i|\vec{X}, \vec{\theta}))$), thus preventing the algorithm from clustering all samples into a single class. A more thorough derivation of this cost function is provided in Section III-A.

An important feature of this algorithm is that it can employ either parametric or non-parametric classifiers. For example, if two classes that can be modelled with normal distributions, a linear separator is sufficient. On the other hand, a four-class problem requires a more complex classifier, such as mixture of Gaussians. If the shape of the distributions is unknown, *k*-NEAREST NEIGHBOR can be used.

A. Derivation of the Cost Function using Fisher Linear Discriminant

Let us assume that two classes, a and b, are normally distributed with means μ_a and μ_b with identical variance $\sigma_a = \sigma_b$. In *Linear Discriminant Analysis* (LDA), the data is projected on a vector $\vec{\omega}$ that maximizes the Fisher criterion $J(\omega)$:

$$J(\omega) = \frac{(\mu_{a\omega} - \mu_{b\omega})^2}{\sigma_{a\omega}^2 + \sigma_{b\omega}^2} \quad (2)$$

with $\mu_{a\omega}$, $\mu_{b\omega}$, $\sigma_{a\omega}$, $\sigma_{b\omega}$ being the means and variances of the data after projection onto $\vec{\omega}$. Data labels are required in order to compute Eq. 2. For an unlabelled data set $\vec{X}$ containing two normally distributed classes a and b, Eq. 2 can be approximated *if* the probability that consecutive samples belong to the same class is greater than 0.5. The within-class variance C_{var} projected on $\vec{\omega}$ is approximated by the average squared difference between consecutive time samples $\vec{x}_t$ projected on $\vec{\omega}$:

$$C_{var}(\vec{\omega}, \vec{X}) = \frac{1}{T-1} \sum_{t=0}^{T-1} (\vec{\omega} \cdot \vec{x}_{t+1} - \vec{\omega} \cdot \vec{x}_t)^2 \quad (3)$$

Its expected value is:

$$E\{C_{var}(\vec{\omega},\vec{X})\} = \sigma_{a\omega}^2 + \sigma_{b\omega}^2 + (\mu_{a\omega} - \mu_{b\omega})^2 P_{trans} \quad (4)$$

The between-class variance C_{dist} can be estimated by the variance of the projected data. Provided that each class has the same prior probability,

$$E\{C_{dist}(\vec{\omega},\vec{X})\} = E\{var(\vec{\omega}\cdot\vec{X})\} =$$

$$\frac{(\mu_{a\omega} - \mu_{b\omega})^2}{4} + \frac{\sigma_{a\omega}^2 + \sigma_{b\omega}^2}{2} \quad (5)$$

Dividing Eq. 5 by Eq. 4 and letting the probability of a transition $P_{trans} \to 0$, we get:

$$E\{\frac{C_{dist}(\vec{\omega},\vec{X})}{C_{var}(\vec{\omega},\vec{X})}\} \to \frac{1}{2} + \frac{(\mu_{a\omega} - \mu_{b\omega})^2}{4(\sigma_{a\omega}^2 + \sigma_{b\omega}^2)} \quad (6)$$

Minimizing the inverse of Eq. 6 corresponds to finding the Fisher criterion $J(\omega)$:

$$\arg\min_{\vec{\omega}} \frac{C_{var}(\vec{\omega},\vec{X})}{C_{dist}(\vec{\omega},\vec{X})} = \arg\max_{\vec{\omega}} \frac{C_{dist}(\vec{\omega},\vec{X})}{C_{var}(\vec{\omega},\vec{X})} \approx \arg\max_{\vec{\omega}} J(\vec{\omega}) \quad (7)$$

with

$$\frac{C_{var}(\vec{\omega},\vec{X})}{C_{dist}(\vec{\omega},\vec{X})} = \frac{\sum_{t=0}^{T-1}(\vec{\omega}\cdot\vec{x}_{t+1} - \vec{\omega}\cdot\vec{x}_t)^2}{var(\vec{\omega}\cdot\vec{X})} \quad (8)$$

The probability that sample $\vec{x}_t$ belongs to class c, for a linear separator classifier, can be expressed by a sigmoid function:

$$p(c|\vec{x}_t) = \frac{1}{1+e^{-kd}} \quad (9)$$

where d is the distance between $\vec{x}_t$ and the boundary decision, and k is a parameter that determines the "sharpness" of the sigmoid. Given a sufficiently small k and significant overlap of the clusters, most of the data will lie within the region $d \ll \frac{1}{k}$. Eq. 9 can then be approximated by:

$$p(c|\vec{x}_t) \approx \frac{d}{k} \quad (10)$$

and Eq. 8 approximated as:

$$\frac{C_{var}(\vec{\omega})}{C_{dist}(\vec{\omega})} \approx \frac{\sum_{t=0}^{T-1}(p(c|\vec{x}_{t+1},\vec{\theta}) - p(c|\vec{x}_t,\vec{\theta}))^2}{var(p(c|\vec{X},\vec{\theta}))} \quad (11)$$

Where $\vec{\theta}$ correspond to the linear separator parameters. Eq. 11 is then normalized by $p(c_a|\vec{X},\vec{\theta})p(c_b|\vec{X},\vec{\theta})$ to reflect the probability of leaving a given state c. This normalizing factor can be approximated by $var(p(c|\vec{X},\vec{\theta}))$, and the final cost function is:

$$E(\vec{X},\vec{\theta}) = \frac{\sum_{t=0}^{T-1}(p(c|\vec{x}_{t+1},\vec{\theta}) - p(c|\vec{x}_t,\vec{\theta}))^2}{var(p(c|\vec{X},\vec{\theta}))^2} \quad (12)$$

B. Optimization: Simulated Annealing

The landscape of the cost function being unknown, simulated annealing was used to find the classifier parameters $\vec{\theta}$ that minimize $E(\vec{X},\vec{\theta})$ described in Eq. Eq. 12. Although very slow, it is necessary due to the presence of local minima. For classifiers with few (less than 20) parameters, one parameter was randomly modified at each step. For *k*-NEAREST NEIGHBOR classifiers, modifications were made to a few points and a small random number of their neighbors. This strategy improved the speed of convergence. The cooling schedule was manually tuned, with longer schedules for more complex problems. Three random restart runs were used, to avoid being trapped in a deep local minimum.

IV. TESTING THE ALGORITHM ON SYNTHETIC DATA SETS

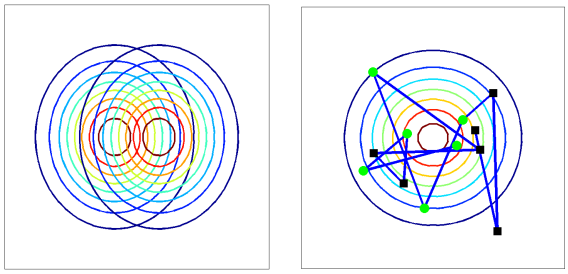
The algorithm was first evaluated using three different classifiers (linear separator, mixture of Gaussians, and *k*-NEAREST NEIGHBOR) on synthetic data sets. This was done to demonstrate the range of cases that can be handled, as our real robot data sets cannot cover some of those cases (e.g., complex-shaped distributions in Section IV-C.5). These sets were generated by sampling a distribution s times (our so-called *segment length*), then switching to the next distribution and drawing s samples again. This segment length determined the amount of temporal coherence present in the data. This process was repeated until a desired sequence length was reached. Fig. 1(b) shows a sequence of 12 samples drawn from two Gaussian distributions, with a segment of length 3. For the test cases presented in this section, the segment lengths were relatively short (between 3 and 5), to demonstrate how the algorithm is capable of handling signals generated from a system that changes state frequently. Results for these synthetic distributions are shown in the following subsections.

A. Linear Separator Classifier with Two Gaussian Distributions

For linear separators, we used two closely-spaced two-dimensional Gaussian distributions (Fig. 1(a)) with identical standard deviation $\sigma_{x\{1,2\}} = 0.863$, $\sigma_{y\{1,2\}} = 1$, and the distance between the means was 1. This simulated cases where features are extremely noisy. Without labels, the combination of the distributions is radially symmetric (Fig. 1(b)). The optimal Bayes classification rate for a single data point for these distributions was 70.7 *percent*, an indication of the difficulty of the problem. The linear separator was trained using Eq. 1, with probabilities computed from Eq. 9. A value of $k = 3$ was chosen, although empirically results were similar for a wide range of k values. 100 time sequences of 102 samples were randomly generated. The average classification success rate was 68.5 ± 6.8 *percent*, which is close to the Bayes classification rate. Fig. 2 shows an example of the classifier posterior probability over time after cost minimization.

B. Mixture of Gaussians Classifier with Three Gaussian Distributions

For *mixture of Gaussians* classifiers, three normal distributions in two dimensions (see Fig. 3) were used. The classi-



(a) Individual distributions (b) Combined Distributions

Fig. 1. Contour plot 1(a) of the two normal distributions used in testing the algorithm with a linear separator. Their combination 1(b) is resembles a single, radially-symmetric Gaussian distribution. A synthetic sequence of 12 data samples with segment length of 3 is also shown in 1(b), with a line drawn between consecutive samples.

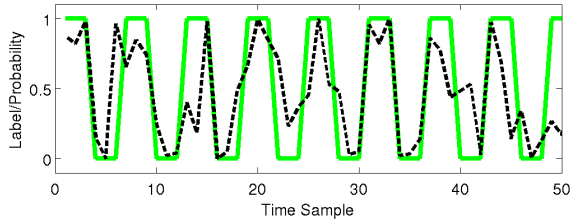
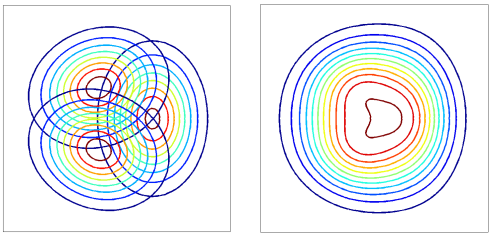


Fig. 2. Classifier posterior probability over time for the two classes drawn from Gaussians distributions depicted in Fig. 1(a), after optimization (dashed line). Ground truth is shown as solid green line. The segment length is 3. The first 50 samples are shown.

fier itself had 9 free parameters: 6 for the two-dimensional Gaussian locations, and 3 for the standard deviation (the Gaussians were radially-symmetric). The three distributions had covariance equal to:

$$\sigma_1 = \begin{pmatrix} .5 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_2 = \begin{pmatrix} .8 & .1 \\ .1 & .6 \end{pmatrix}, \sigma_3 = \begin{pmatrix} .8 & -.1 \\ -.1 & .6 \end{pmatrix}$$

The distribution centers were located at a distance of 0.95 from the $(0, 0)$ location and at 0, 120 and 240 *deg* angles. The optimal Bayes classification rate for these distributions is 74 *percent*. 100 time sequences of 207 samples with segment length of 3 were generated. The average classification rate was 69.2 ± 9.3 *percent*.



(a) Individual distributions (b) Combined Distributions

Fig. 3. Contour plot of the three normal distributions 3(a) and their sum 3(b) used to test the algorithm with a mixture of Gaussians classifier.

C. K-Nearest-Neighbor Classifier with Uniform Distributions

As an instance-based learning method, *k*-NEAREST NEIGHBOR [12] has the significant advantage of being able to represent complex distributions. A major drawback associated with this classifier is the large number of parameters (proportional to the number of points in the data set). This results in lengthy computation time in order to find the parameters that minimizes the cost function. Five different test cases (few data points, unequal number of samples per class, overlapping distributions, six-class distributions and complex-shaped distributions) were designed to test the performance of the algorithm using this classifier.

1) Few Data Points: For each test sequence, only 36 samples were drawn from 3 square, non-overlapping uniform distributions, with a segment length of 3. The classifier used $k = 10$ neighbors with a Gaussian kernel $\sigma = 0.8$. The mean classification success rate over 100 trials was 93.7 ± 6.1 *percent*. Fig. 4 shows these results in more detail.

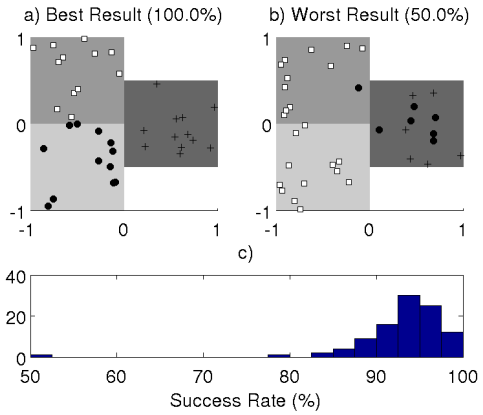


Fig. 4. Results of clustering method applied on a data set of 36 points drawn from three equal distributions. The best a) and worst b) results of clustering are shown, with distributions shown as background grey boxes. c) shows the histogram of success rates over 100 trials.

2) Unequal Number of Samples per Class: Three uniform rectangular distributions of equal density, but different area were sampled with a segment length is 3. In total, 84 samples were drawn from the smallest, 168 from the medium and 252 from the largest distribution. The classifier used $k = 20$ neighbors with a Gaussian kernel $\sigma = 0.8$. The average classification success rate over 80 trials was 87.8 ± 11.6 *percent*. Fig. 5 shows these results in more details.

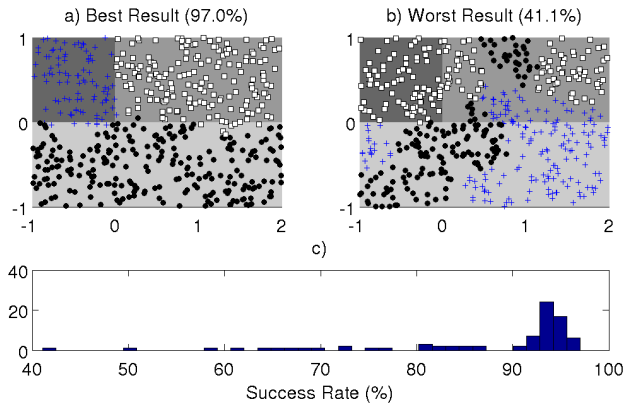


Fig. 5. Results of clustering method applied on a data set of 504 points, drawn from three distributions of different sizes. The best a) and worst b) results of clustering are shown, with distributions shown as background grey boxes. c) shows the histogram of success rates over 80 trials.

3) Overlapping Distributions: Two distributions with significant overlapping (40 *percent*) were used to generate the data. The overlapping regions were selected so non-overlapping regions within a class would be of different sizes. The classifier used $k = 20$ neighbors with a Gaussian kernel $\sigma = 0.8$. 57 test sequences of 504 points, with a segment length of 3 were randomly generated. The average classification success rate was $76.3 \pm .6.4$ *percent*, not too far from the maximum possible rate of 80 *percent*.

4) Six-Classes Distributions: Six square uniform distributions were used in these tests. The classifier used $k = 10$ neighbors with a Gaussian kernel $\sigma = 0.8$. 100 test sequences of 306 points, with a segment length of 3 were randomly generated, with detailed results shown in Fig. 6. The average classification rate was 90.0 ± 5.7 *percent*.

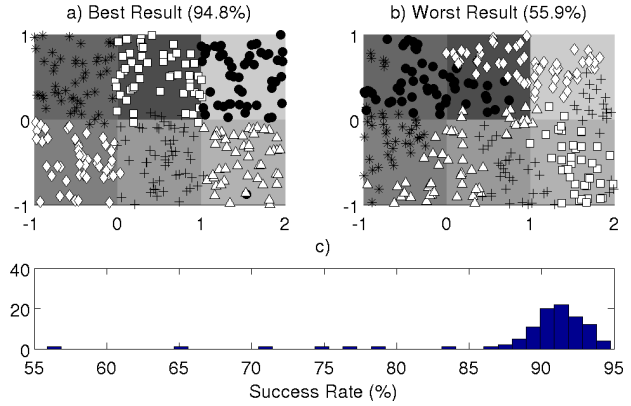


Fig. 6. Clustering results for a data set of 306 points drawn from 6 distributions of equal sizes, with segment length of 3. The best a) and worst b) results are shown, with distributions shown as background grey boxes. c) shows the histogram of success rates over 100 trials.

5) Complex-Shaped Distributions: Complex-shaped distributions were simulated using two, two-dimensional spirals. Data was generated according to the following equations:

$$x_1 = (tt + d_\perp) * cos(\phi + \phi_0), \quad x_2 = (tt + d_\perp) * sin(\phi + \phi_0) \tag{13}$$

with $\phi = d_{arc}\sqrt{rand\{0..1\}}$ the arc distance from the center, $d_\perp = N(0, \sigma_{SRnoise})$ a perpendicular, normally distributed distance from the arc, and ϕ_0 equal to 0 for the first distribution and π for the second. 5,000 data points were drawn for each trial, with segment length of 5 using $d_{arc} = 15$ and $\sigma_{SRnoise} = 0.9$ for the distributions. Fig. 7 shows time sequences of 10 and 50 samples from the test sequence used. One can see that the shape of the distributions cannot be inferred from short sequences, even when data labelling is provided.

The k-NEAREST NEIGHBOR classifier used $k = 40$ neighbors, with a Gaussian kernel of $\sigma = 0.4$. Fig. 8 shows classification success rate achieved for the 19 test cases generated. Fig. 9 shows a successful and unsuccessful case of clustering. If we exclude the 3 unsuccessful cases, considering them as being outliers due to the simulated annealing getting stuck in a local minimum, the average value of classification success was 92.6 ± 0.4 *percent*.

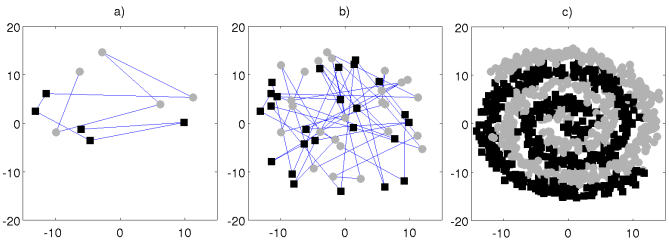


Fig. 7. Time sequence shown in feature space for a) ten samples and b) fifty samples drawn randomly from the distributions described in Eq. 13 and shown in c), with segment length of 5 samples. The spirals are not visible in a) and barely discernible in b).

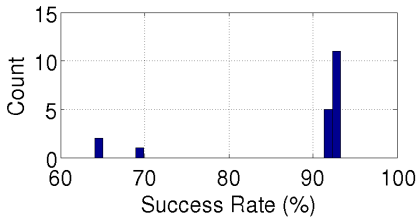


Fig. 8. Histogram showing the distribution of the 19 classification success rates after clustering. The majority of results were located around 92 *percent*, with three cases failing to completely identify the underlying structure.

D. Comparison with a Window-based Method

Performance of this algorithm was compared to the segmentation algorithm described in Lenser *et al.* [4]. The latter algorithm was run until only two clusters were left. The data used was generated from a simpler version of the two-spiral distributions, with $d_{arc} = 5$ and $\sigma_{SRnoise} = 0.9$., with 250 samples per test case. A test case is shown in Fig. 10 a), without temporal information for clarity. For long segment lengths (over 60 samples), success rates are similar for both methods. As expected, Fig. 10 b) shows that shorter segment lengths negatively affect the window-based method, with larger windows being affected most. This can be explained by the merger of the two clusters through windows containing data from both distributions. The number of such windows is greater for a larger window size and a smaller segment length.

V. TESTING ALGORITHM ON ROBOT SENSOR DATA: AUTONOMOUS TERRAIN DISCOVERY

The vehicle used to collect various ground surfaces data (Fig. 11) [14] was a hexapod robot specifically designed for amphibious locomotion. It was previously shown in Giguere *et al.* [15] to be able to recognize terrain types using supervised learning methods. The robot was equipped with a 3-axis Inertial Measurement Unit (3DM-GX1^{TM}) from Microstrain. The sensors relevant for terrain identification are: 3 accelerometers, 3 rate gyroscopes, 6 leg angle encoders and 6 motor current estimators. Each sensor was sampled 23 times during a complete leg rotation, thus forming a 23-dimensions vector. Multiple vectors could be concatenated together to improve detection, forming an even higher dimensionality feature vector for each complete leg rotation. Dimensionality was subsequently reduced by applying Principal Component Analysis (PCA). In the following experiments, only the two first main components

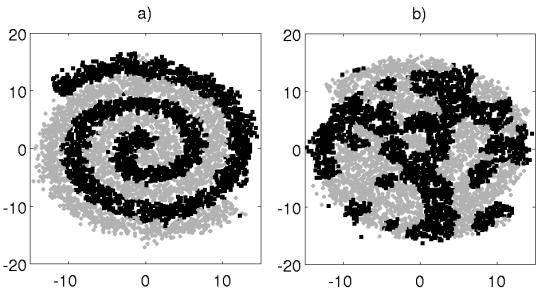


Fig. 9. Successful (92 *percent*) a) and unsuccessful b) clustering for a two-class problem of 5,000 points generated according to Eq. 13. The segment length was 5.

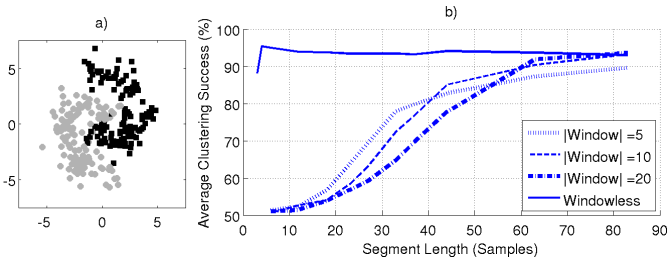


Fig. 10. Average clustering success rate for the windowless algorithm and the segmentation algorithm in Lenser *et al.* [4] for time-window sizes of 5, 10 and 15 samples. When transitions are infrequent (corresponding to segment length over 60), success rates are similar for both methods. For shorter segment lengths, the window-based method fails to identify the two clusters and instead simply merge them together.

were used. Even though some information is discarded, our previous results in [15] indicated that this was sufficient to distinguish between small numbers of terrains. If more discrimination is needed, other components can be added.

These experiments were restricted to level terrains, with small turning maneuvers. Changes in terrain slopes or complex robot maneuvers impact the dynamics of the robot, and consequently affects how a particular terrain is perceived by the sensors. Discarding data when the robot performs a complicated maneuver or when the slope of the terrain crosses a threshold would mitigate this issue. Terrains were selected to offer a variety of possible environments that might be encountered by an amphibious robot. They were also different enough that, from a locomotion point of view, they are distinct groups, and thus form classes on their own.

Fig. 11. The hexapod robot, shown equipped with semi-circle legs for land locomotion. The vehicle moves forward by constantly rotating legs in two groups of three, forming stable tripod configurations.

A. *Overlapping Clusters with Noisy Data*

The first data set was collected in an area covered with grass, with a section that had been recently tilled. The robot was manually driven in a straight line over the grass, crossing eventually to the tilled section. The robot was then turned around, and manually driven towards the grass. Eight transitions were collected in this manner. The problem was made more challenging by using only the pitch angular velocity vector. Using more sensor information would have reduced the noise and increased the relative separation between the two clusters.

Two classifiers were used in the clustering algorithm for this data set. The first one was a linear separator with a constant $k = 3.0$. Classification success after clustering was 78.6 *percent* (see Figs. 12 and 14(a)), a value certainly close to the Bayes error considering the overlap between the classes. The second classifier used for clustering was a k-NEAREST NEIGHBOR classifier with $k = 10$ and $\sigma = 1.0$ for the kernel. As expected, k-NEAREST NEIGHBOR had slightly inferior results, with a classification success rate of 73.9 *percent* (see Figs. 13 and 14(b)). The larger number of parameters (454 compared to 2 for the linear separator) makes this classifier prone to over-fit the data, potentially explaining the difference in performance.

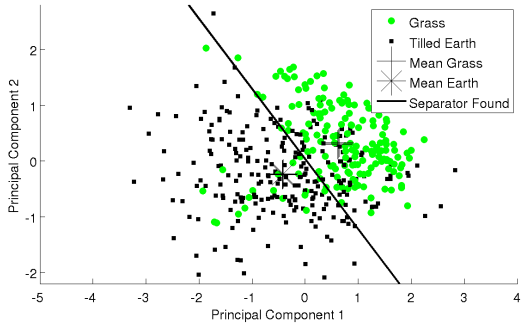


Fig. 12. Sensor data set collected for a robot walking on grass and tilled earth, with eight transitions present in the data. The solid line represents the separator found on the unlabelled data using the algorithm with a linear separator as classifier. Even though the clusters have significant overlap, the algorithm still managed to find a good solution. Notice how the separator is nearly perpendicular to a line joining the distribution centers, an indication that the solution is a close approximation to *LDA* on the labelled data.

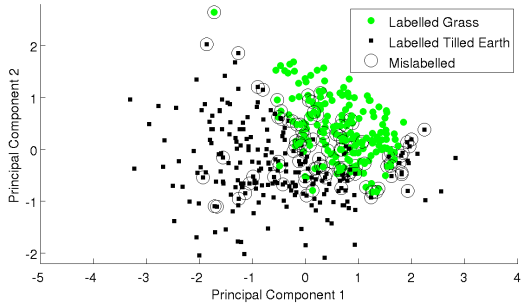
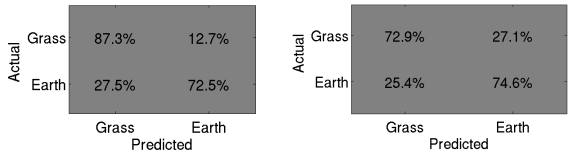


Fig. 13. Same data set as Fig. 12, clusterized with the algorithm using a k-NEAREST NEIGHBOR classifier. The circled data points are wrongly labelled. Most of them are located either at the boundary between the two distributions, or deep inside the other distribution.



(a) Linear Separator (b) k-NEAREST NEIGHBOR

Fig. 14. Confusion matrix for classification of two-terrain data obtained after clustering using a) a linear separator and b) the k-NEAREST NEIGHBOR.

B. Fast-Switching Semi-Synthetic Data

Another data set was collected over five different terrains, and a high-dimensionality feature vector for each complete leg rotation was generated using 12 sensors (all 3 angular velocities, all 3 accelerations and all 6 motor currents). As in the previous case, only the two first principal components were used. Of all sensors, the motor currents were the most informative.

No rapid terrain changes were present in the original data set, so segments of data were selected according to the random state change sequence shown in Fig. 15. This artificially decreased the average segment length to a value of 6, with individual states having average segment lengths between 3.5 and 9.8. A classifier with a mixture of five radially-symmetric Gaussians was used in the clustering algorithm. Fig. 16 shows the labelling of the data set and Fig. 17 shows the confusion matrix. Overall, 91 *percent* of the data was grouped in the appropriate class. Even though some of the distributions are elongated (for example *linoleum*), the combination of symmetric Gaussians managed to capture the clusters. Standard clustering techniques struggled with these distributions, with average classification success rates of 68.2 and 63.0 *percent* for mixture of Gaussians and K-means clustering, respectively.

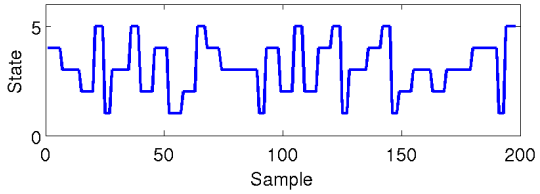


Fig. 15. State (from 1 to 5) sequence used to generate the time sequence of data in the clustering problem shown in Fig. 16. The shortest segment length is 3, and the average segment length is 6.

VI. DISCUSSION

A. Comparing Cost Functions from Previous Works

In [3], the cost function to be minimized is:

$$E(\Theta) = \sum_{t=1}^{T}(y_t - \sum_{s=1}^{N} p_{s,t} f_s(\vec{x_t}))^2 + C \sum_{t=1}^{T-1} \sum_{s=1}^{N} (p_{s,t+1} - p_{s,t})^2 \quad (14)$$

with $f_s(.)$ as a kernel estimator for data $\vec{x_t}$ for state s, $p_{s,t}$ as the mixing coefficient for state s at time t, $\theta = p_{s,t} : s = 1, ..., N; t = 1, ..., T$ as the set of coefficients to be found, and C as a regularization constant. In [2], the cost function is:

$$o(\vec{s}) = \sum_{t=W}^{T} d(p_{s(t)}(\vec{x}), p_t(\vec{x})) + Cn(\vec{s}) \quad (15)$$

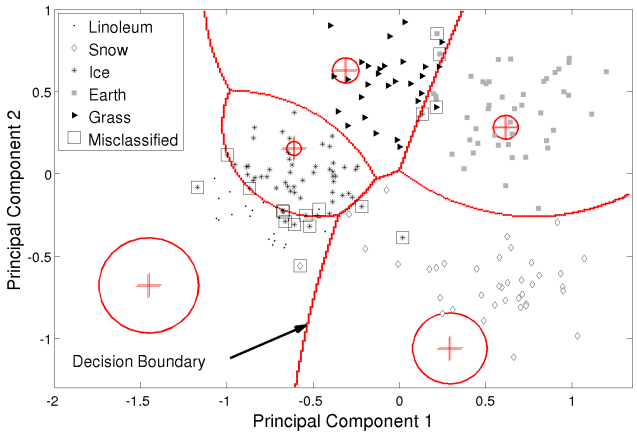


Fig. 16. Applying the clustering algorithm on a data set of five environments, with a mixture of five Gaussians as a classifier. Labels shown are the ground truth. The samples surrounded with a small square are wrongly labelled. The five red cross-hairs are the location of the Gaussians, the thick circles representing their standard deviation. The decision boundary is represented by the red curved lines.

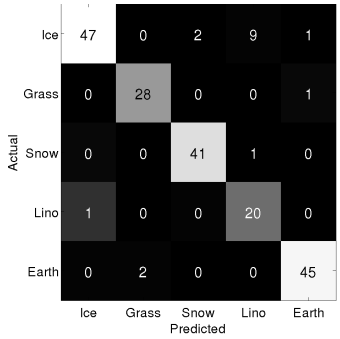


Fig. 17. Confusion matrix for the clustering shown in Fig. 16.

With $d(.,.)$ being a L_2-Norm distance metric function, $\vec{s}$ a state sequence, $p_t(\vec{x})$ the *pdf* estimate within the window at time t, $p_{s(t)}(\vec{x})$ the prototype *pdf* for state $s(t)$, $n(\vec{s})$ is the number of segments and C a regularization constant.

Eq. 14 and Eq. 15 can be separated in two parts: the first part minimizes representation errors, and the second part minimizes changes over time. A regularization constant C is needed to balance them, and selecting this constant is known to be a difficult problem. In contrast, our cost function (Eq. 12) does not take into account the classifier fit to the actual data $\vec{X}$. Instead we solely concentrate on the variation of the classifier output over time, thus completely eliminating the need for a regularization constant and associated stabilizer. This is a significant advantage. The fact that the classifier fit is not taken into account by our algorithm can be seen in the mixture of Gaussian result case shown in Fig. 16: the location of the Gaussians and their width (marked as circled red cross-hairs) do not match the position of the data they represent. Only the decision boundaries matter in our case. If the actual *pdf*s are required, a suiting representation can be fitted using the probability estimates found.

B. Distances and Window Size for Complex Distributions

Algorithms relying on distances between *pdf*s described within small windows such as the one used by Kohlmorgen *et*

al. [2] succeed if the distance between a sample and other members of its class is much smaller than its distance to members of the other class. For complex distributions such as the one used in the spirals case, this is not the case: the average distance between intraclass points and interclass points is almost identical: 13.4 vs 13.9. This can be seen in Fig. 18, were the distribution of distances are almost overlapping. Our algorithm with a *k*-NEAREST NEIGHBOR classifier succeeds because it concentrates on nearby samples collected over the whole duration, and these have the largest difference between intraclass and interclass distances (distance less than 2 in Fig. 18).

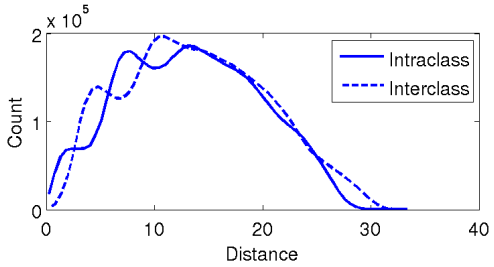


Fig. 18. Distribution of distances between samples belonging to the same class (intraclass) and samples belonging to different classes (interclass), for the spirals data set used in Fig. 9. The distributions almost completely overlap, except at shorter distances.

Moreover, to avoid transitions in their time-windows, these algorithms would have to limit their time-window sizes. Bearing in mind that for the spirals data set in Fig. 9 the segment length was 5, this would result in window sizes of 2 to 3 samples. These distributions cannot be approximated in a satisfactory manner with so few points. From simple visual inspection one can see that it requires, at a minimum, 10 to 20 points. An advanced dimensionality reduction technique such as *Isomap* [16] could be used to simplify this type of distribution. If data points bridging the gap between the spirals were present at regular intervals however, *Isomap* would be of little help.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented a new clustering method based on minimizing a cost function related to the probability output of classifiers. Synthetic test cases were used to demonstrate its capabilities over a range of distribution types. The same method was employed on data collected with a walking robot, clustering its sensor data in terrain categories successfully.

As future work, we intend to quantify the impact of segment lengths, overlap and number of points in sequence on the clustering process. We are also looking at methods to estimate the number of classes present in the data set, so it would not need to be known beforehand. Since merging classes together does not affect the cost but splitting a valid class does, a rapid increase of cost-per-data point as we increase the number of clusters in the algorithm is a potential indicator that we have found the proper number of classes.

A major drawback of using simulated annealing to find the minimum cost is its prohibitive computing time, in the order of hours on a dual CPU 3.2 GHz Xeon machine for the most complex distributions. There are indications that other optimization techniques could be employed to drastically reduce the amount of computation required.

We are looking at applying this algorithm outside the scope of terrain discovery. Many segmentation problems exhibit continuity in their sets, such as texture segmentation (spatial continuity) or video (temporal continuity) segmentation. Feature ranking for continuous processes is another possible application, where the cost function could be employed to evaluate each feature individually.

REFERENCES

[1] Pawelzik, K., Kohlmorgen, J. and Muller, K.-R. (1996). *Annealed competition of experts for a segmentation and classification of switching dynamics*. Neural Computation, 8(2), 340–356.

[2] J. Kohlmorgen and S. Lemm. *An on-line method for segmentation and identification of non-stationary time series*. In NNSP 2001, pp. 113-122.

[3] J. Kohlmorgen, S. Lemm, G. Rätsch, and K.-R. Müller. *Analysis of nonstationary time series by mixtures of self-organizing predictors*. In Proceedings of IEEE Neural Networks for Signal Processing Workshop, pages 85-94, 2000.

[4] S. Lenser and M. Veloso. *Automatic detection and response to environmental change*. In Proceedings of the International Conference of Robotics and Automation, May 2003

[5] Weiss C., Fröhlich H., Zell A. *Vibration-based Terrain Classification Using Support Vector Machines*. In Proceedings of the IEEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October, 2006.

[6] Weiss C., Fechner N., Stark M., Zell A. *Comparison of Different Approaches to Vibration-based Terrain Classification*. in Proceedings of the 3rd European Conference on Mobile Robots (ECMR 2007), Freiburg, Germany, September 19-21, 2007, pp. 7-12.

[7] Brooks C., Iagnemma K., Dubowsky S. *Vibration-based Terrain Analysis for Mobile Robots*. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005.

[8] Dupont, E. M., Moore, C. A., Collins, E. G., Jr., Coyle, E. *Frequency response method for terrain classification inautonomousground vehicles*. In Autonomous Robots, January, 2008.

[9] Sadhukan, D., Moore, C. (2003). *Online terrain estimation using internal sensors*. In Proceedings of the Florida conference on recent advances in robotics, Boca Raton, FL, May 2003.

[10] S. Lenser, M. Veloso, *Classification of Robotic Sensor Streams Using Non-Parametric Statistics*, Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, October, 2004, pp. 2719-2724 vol.3

[11] N. Srebro, G. Shakhnarovich, S. Roweis, *When is Clustering Hard?*. PASCAL Workshop on Statistics and Optimization of Clustering, July 2005.

[12] Cover, T.M. and Hart, P.E., 1967. *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory, 13(1), 21-27.

[13] U. Saranli, M. Buehler, D.E Koditschek *RHex: A simple and highly mobile hexapod robot*, The International Journal of Robotics Research, vol. 20, no.7, pp.616-631, 2001.

[14] G. Dudek, M. Jenkin, C. Prahacs, A. Hogue, J. Sattar, P. Giguere, A. German, H. Liu, S. Saunderson, A. Ripsman, S. Simhon, L. A. Torres-Mendez, E. Milios, P. Zhang, I. Rekleitis *A Visually Guided Swimming Robot*, Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1749-1754, 2005.

[15] P. Giguere, G. Dudek, C. Prahacs, and S. Saunderson. *Environment Identification for a Running Robot Using Inertial and Actuator Cues*. In Proceedings of Robotics Science and System (RSS 2006), August, 2006.

[16] J. Tenenbaum, V. de Silva, J. Langford. *A Global Geometric Framework for Nonlinear Dimensionality Reduction*. Science 22 December 2000: Vol. 290. no. 5500, pp. 2319 - 2323.