

# A Semi-Supervised Method for Learning the Structure of Robot Environment Interactions

Axel Großmann<sup>1</sup>, Matthias Wendt<sup>1</sup> and Jeremy Wyatt<sup>2</sup>

<sup>1</sup> Department of Computer Science  
Technische Universität Dresden  
Dresden, Germany  
{axg,mw177754}@inf.tu-dresden.de

<sup>2</sup> School of Computer Science  
The University of Birmingham  
Birmingham, UK, B15 2TT  
j1w@cs.bham.ac.uk

**Abstract.** For a mobile robot to act autonomously, it must be able to construct a model of its interaction with the environment. Oates *et al.* developed an unsupervised learning method that produces clusters of robot experiences based on the dynamics of the interaction, rather than on static features. We present a semi-supervised extension of their technique that uses information about the controller and the task of the robot to (i) segment the stream of experiences, (ii) optimise the final number of clusters and (iii) automatically select the individual sensors to feed to the clustering process. The technique is evaluated on a Pioneer 2 robot navigating obstacles and passing through doors in an office environment. We show that the technique is able to classify high dimensional robot time series several times the length previously handled with an accuracy of 91%.

## 1 Introduction

We would like our mobile robots to operate successfully in the real world. Independently of whether our aim is truly autonomous behaviour or just reliable and robust operation, this requires the robots to collect information about the interaction with the physical environment. In particular, we want an automatic technique for constructing a model of the world dynamics. Since our particular goal is to use such a model for execution monitoring [4, 5] at the level of reactive control, it should support predictions about the qualitative outcome of actions as well as help in explaining situations in which the actions had unintended effects.

As the interaction of a robot with the environment is complex, a description of it will be difficult to obtain. On the one hand, we would favour an unsupervised learning technique, e.g., the work by Oates *et al.* [14] on clustering robot-sensor data using dynamic time warping as similarity measure. On the other hand, learning a world model is fundamentally a supervised learning problem. As the entire world dynamics will be huge in any realistic application, it will be important to use the robot's goals to focus the model learning so that only useful and important aspects of the interaction

are represented [8]. We want to address this dilemma by devising a semi-supervised method.

The approach by Oates *et al.* has a number of deficiencies if it is to be applied to non-trivial robotic tasks. To our knowledge, it has been applied so far only to rather simple settings: a mobile robot moving along a straight line or turning in place, where the actions had durations between 2 and 8 seconds, only 4 sensors were used and these were selected manually. It is not clear how the technique performs for actions lasting up to 30 seconds or so, or whether it can deal with higher dimensional data.

We present a semi-supervised method for clustering robot experiences that can be seen as an extension of the work by Oates *et al.*. Our technique uses information about the reactive controller and the task of the robot to segment the stream of experiences and to optimise the final number of clusters as well as the contribution of the individual sensors to the clustering process. The technique is evaluated on an office-delivery robot passing through doors and navigating dynamic obstacles. Eventually, the robot can reliably distinguish situations in which it was able to pass through a door or avoid an obstacle successfully from situations in which it failed to do so.

The paper is organised as follows. In Section 2, we introduce the main ideas and concepts of learning models of a reactive controller. In Section 3, we describe the individual steps of the learning algorithm. In Section 4, we evaluate the performance of the algorithm in experiments with an office-delivery robot. We conclude in Section 5 and outline future work in Section 6.

## 2 Obtaining Models of a Reactive Controller

Designing a robot controller for navigation tasks in dynamic environments is hard. In general, we cannot foresee and consider all the possible situations in the interaction of the robot with the environment that might make a difference to success or failure of an action. However, given an existing controller, we can collect a reasonable amount of data and pose this problem as a typical knowledge discovery application [7].

Without any considerable loss of generality, we focus on behaviour-based controllers in the following. This type of controller consists of several interacting, task-specific programs that are referred to as low-level behaviours. Each behaviour program takes the current sensor readings and the state information and computes target values of the robot's actuators. Individual behaviours can overwrite the output of other behaviours. Moreover, behaviours can be temporarily deactivated. The activation context of a behaviour may be specified in terms of the sensory inputs, the state information, or the output of other behaviours. Examples of low-level behaviours for navigation tasks are obstacle avoidance, travelling to a target location, or corridor following.

As we are concerned with learning a model of the world dynamics, we can decide to build such a model for the reactive controller as a whole or for the task-specific low-level behaviours individually. In the first case, we consider the reactive controller to be a black box with a given set of inputs (the current sensor readings and state information obtained by the localisation and object-detection modules) and outputs (the actuator settings) and an evaluation function specifying a task-specific performance measure. In the second case, we model each low-level behaviour as a black box. As each behaviour

is supposed to achieve a specific sub-task, the set of inputs and outputs is expected to be smaller than in the first case, and the evaluation function simpler and more specific. In either case, the inputs and the performance values can be represented as multivariate time series.

The overall approach to learning a model of a reactive controller we propose consists of four steps: (1) recording sensory, status, and performance information of the controller during the execution of high-level actions, (2) finding qualitatively different outcomes by clustering the robot’s experiences, (3) finding frequently occurring patterns in selected experiences, and (4) building a model the world dynamics for specific purposes such as the detection of exceptional situations and significant events, execution monitoring, or object detection. In our paper, we specifically address steps (1) and (2).

### 3 Classification of Sensory Data

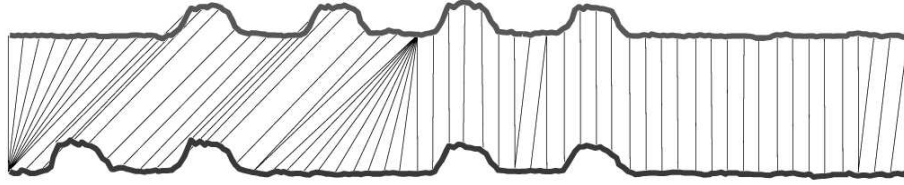
The clustering algorithm for robot experiences presented in this section is an extension of classical clustering algorithms to handle finite sensor histories. The main challenge thereby is to devise a means for comparing time series that is suitable for the robotics domain.

**Representing signals.** At the level of the sensors, the basic objects to be compared are not sequences but are continuous functions of the continuous parameter time. The values of the functions are usually multidimensional, i.e., we obtain readings from the set of sensors. In practise, the continuous functions are converted into discrete sequences by some sampling process. In addition, the time series are segmented by high-level actions of the robot. We assume that the robot can only execute a single high-level action a time.

Let  $E$  denote an experience, represented as a multivariate time series containing  $n$  measurements from a set of sensors recorded during the robot’s operation such that  $E = \{\mathbf{e}_t \mid 1 \leq t \leq n\}$ . The  $\mathbf{e}_t$  are vectors of values containing one element for each sensor. We do not distinguish between raw sensor readings obtained from physical sensors and preprocessed information from virtual sensors.

As the values for each experience  $E$  were recorded over the course of engaging in a single high-level action, we store the corresponding high-level action and information about the specific outcome of this action. Suppose an office delivery robot is navigating in an environment. Some delivery actions will be completed successfully while the robot fails at others. We distinguish three possible outcomes: success, failure, and timeout. For each high-level action, we obtain a set of  $m$  experiences. For each sensor, the range of the measurements is determined. As a further preprocessing step, the feature values are transformed to the interval  $[0, 1]$ .

**Temporal abstraction.** Ideally, we want to find subsets of qualitatively similar experiences. We consider two experiences as qualitatively similar if, in the respective time intervals, the robot’s actions had similar effects on the environment as seen from the robot’s sensors. Oates *et al.* [14] argued that qualitatively similar experiences may be quantitatively different in at least two ways: (1) they may be of different lengths and (2) the rate at which progress is made can vary non-linearly within a single time series. This is, the similarity measure for time series must support temporal abstraction. The



**Fig. 1.** Given two one-dimensional time series that have an overall similar shape, but are not aligned in the time axis, DTW can efficiently find an alignment between them.

standard approach of embedding the time series in a metric space and using Euclidean distance of similarity measure does not meet this requirement.

Rosenstein and Cohen [15] and Höppner [7] have obtained temporal abstraction by using a pattern alphabet for each type of sensor. That is, the experiences are represented in a unit- and scale-independent manner as sequences of feature vectors, where the feature values are pattern labels. By specifying a time window, the pattern alphabet can be learnt. However, this does not seem possible in our case given the little *a priori* knowledge available about the controller and its interaction with the environment. We need a method that does not impose any constraint on the length of the experiences.

**Dynamic time warping.** Suppose that two time series have approximately the same overall component shapes, but these shapes do not line up in the time axis. To find the similarity between such sequences, we must ‘warp’ the time axis of one series to achieve a better alignment. Dynamic time warping (DTW) is a technique for efficiently achieving this warping. It has been used successfully in many domains including speech processing [16], data mining [12], and robotics [14]. See Figure 1 for an example.

DTW appears to be well suited for clustering experiences in a velocity-independent fashion. However, there is also an associated cost. The time complexity of the algorithm is  $O(n_1 n_2)$  for comparing two experiences  $E_1$  and  $E_2$ , of length  $n_1$  and  $n_2$ , respectively. Recently, techniques have been developed that address this problem. For example, it is possible to speed up DTW by one to three orders of magnitude using segmented DTW [12] provided that the time series can be approximated by a set of piecewise linear segments. The classical versions of DTW may show the undesired behaviour of mapping a single point on one time series onto a large subset of another time series. This problem, referred to as singularities, can be avoided by using the (estimated) local derivatives of the data instead of the raw data itself [13].

**Local distance measure.** DTW requires the definition of a distance function between the values of the sequence at a given time frame. In the following, this is referred to as local distance measure. As in most high-dimensional applications, the choice of the distance metric is not obvious. There is very little literature on providing guidance for choosing the correct distance measure which results in the most meaningful notion of proximity between two records [1]. We have used the following generic local distance function:

$$d(x, y) = \left[ \sum_{i=1}^N (w^i |x^i - y^i|)^k \right]^{1/k}$$

where  $x$  and  $y$  are the  $N$ -dimensional feature vectors, and  $w$  is an  $N$ -dimensional weight vector. We obtain Manhattan distance if  $k = 1$ , Euclidean distance if  $k = 2$ , and fractional distance if  $k < 1$ . Aggarwal *et al.* [1] argued that the use of fractional distance may lead to improvements in high-dimensional data-mining problems.

Given two experiences  $E_1$  and  $E_2$ , DTW uses dynamic programming to find a warping that minimises the area of the local distance function  $d$  in time for  $E_1$  and  $E_2$ . The area is used as a measure of similarity between the two experiences.

**Sensor weighting.** For any given situation, only a small subset of the robot's sensors may contribute to the time-series patterns that characterise the situation. Instead of specifying the contributing sensors *a priori*, a learning algorithm should weigh the importance of each sensor automatically. This may be possible, if the clustering process does not only take into account the sensor readings, but also information about the task to be performed.

The contribution of the individual sensors to the local distance measure is determined by the weight vector  $w$ . In order to find suitable parameters for the sensor weights, we propose the following three-step strategy: (1) We try to reduce the dimensionality  $N$  by using preprocessed information (virtual sensors) instead of raw sensor readings. Suppose a robot is equipped with five sonar sensors in forward direction. Instead of using the raw readings from all five sensors, we compute estimates of the distances to the closest obstacles for only three areas (front-right, front-centre, and front-left). (2) We exclude individual sensors using knowledge about the task and the implementation of the controller or the low-level behaviour. (3) We optimise the weight parameters automatically using the experiences and information about the outcome of the actions (success or failure). The weights in this paper are binary, and selected by searching through the space of combinations of features. DTW and hierarchical clustering are carried out for each combination of features, and the performance evaluated. If the current feature set being tried outperforms the previous best feature set then the clustering associated with the new best replaces the previous best.

**Hierarchical clustering.** Given the set of  $m$  experiences and an instance of the local distance function, we compute a complete pairwise distance matrix. Using a symmetrised version of the algorithm, this means invoking DTW  $m(m - 1)/2$  times. Hierarchical clustering is an appealing approach to problems in which there is no single 'obvious' partition of the input space into well-separated clusters. Agglomerative algorithms work by beginning with singleton sets and merging them until the complete input set is achieved. Hence, they produce a hierarchical, binary cluster tree. Given the tree, one can conveniently trade off between the number of clusters and the compactness of each cluster.

Oates *et al.* [14] used the average-linkage method together with DTW. They found that robot experiences with any noise were grouped together and distinguished from those that had none. In fact, most clustering methods are biased toward finding clusters possessing certain characteristics related to size, shape, or dispersion: Ward's method tends to find clusters with roughly the same number of members in each cluster; average linkage is biased toward finding clusters of equal variance; and single linkage performs well for elongated or irregular clusters [3]. Due to the lack of prior knowledge, we selected all three for the experiments.

**Search for suitable distance functions.** The stopping criterion used by Oates *et al.* [14] is based on distance information. Specifically they do not merge clusters when the mean inter-cluster distance and the mean intra-cluster distance are different and that difference is statistically significant at 5% as measured by a t-test. In this paper we use the same criterion to stop the clustering process. However, we perform multiple clusterings using different sensor weightings, trying to maximise the evaluation function  $q = k_1 \frac{1}{C} + k_2 \frac{A}{B}$ , where  $C$  is the final number of clusters,  $A$  is the number of correct classifications, and  $B$  is the number of misclassifications. An experience is classified correctly if it has the same action outcome as the majority of elements of the cluster it belongs to. The parameters  $k_1$  and  $k_2$  can be tuned for the specific application.

---

Given a robot controller, a selected high-level action  $a$ , the set of low-level behaviours  $\mathcal{B}$  implementing  $a$ , the set of  $N$  sensors  $\mathcal{S}$  used as inputs by  $\mathcal{B}$ , a set of  $m$  experiences for  $a$ , and a method for selecting individual sensors from  $\mathcal{S}$ .

- (0) Let  $r = 0$  and  $q_0 = 0$ . Create an initial instance of the distance function,  $d_0$ , with  $w_i = 0$  for  $i = 1 \dots N$ .
- (1) Let  $r = r + 1$ .
- (2) Select a single sensor  $s_k$  from  $\mathcal{S}$  and create a new instance  $d_r$  from  $d_{r-1}$  by setting  $w_k = 1$ .
- (3) Compute the distance matrix by invoking DTW  $m(m-1)/2$  times using  $d_r$ .
- (4) Perform hierarchical, agglomerative clustering and obtain a set of clusters  $\mathcal{C}$ .
- (6) Evaluate the current partitions by computing  $q_r = q(\mathcal{C})$ .
- (7) If  $q_r > q_{r-1}$  keep  $w_k = 1$ , set  $w_k = 0$  otherwise.
- (8) If there are still sensors to select go to (1), terminate otherwise.

---

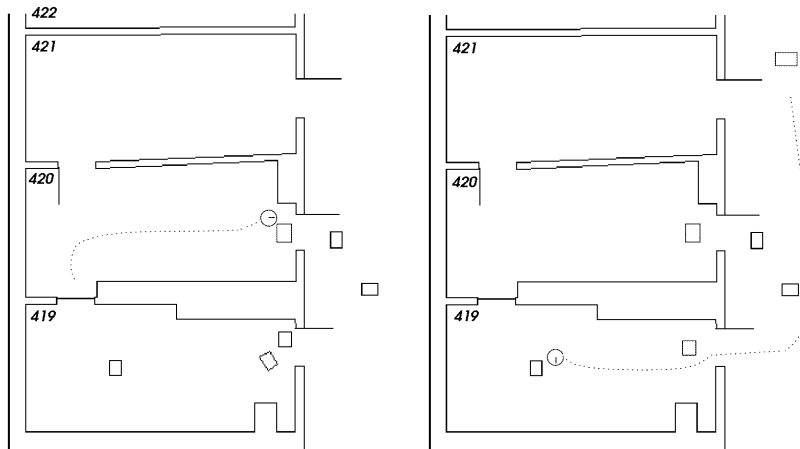
**Fig. 2.** The semi-supervised algorithm for clustering robot experiences.

The complete high-level algorithm is given in Figure 2. After obtaining a suitable partition of the experiences, we have several options on how to proceed. We can select or compute cluster prototypes [14], or we can search for reoccurring patterns within the experiences of each cluster and use them to represent differences between the clusters. In this way, we could predict the outcome of robot actions.

## 4 Experimental Results

The aim of the experiments is to investigate the usability of DTW as similarity measure and the performance of the proposed learning algorithm in a realistic and sufficiently relevant robotics application.

**Application scenario.** We have used a system specifically developed for offce-delivery tasks [5]. It was used on a Pioneer 2 robot, equipped with a camera, a ring of sonar sensors, and a laser-range-finder. The control software of this system consists of a behaviour-based reactive controller, a position-tracking system, a path planner, and a high-level planning and reasoning system. We replaced the reactive controller on



**Fig. 3.** Typical trajectories in the office environment. *Left:* In Room 420, the low-level behaviour *GoToPos* fails as the way is blocked by an obstacle. *Right:* On the way to Room 419, the robot successfully avoids two obstacles.

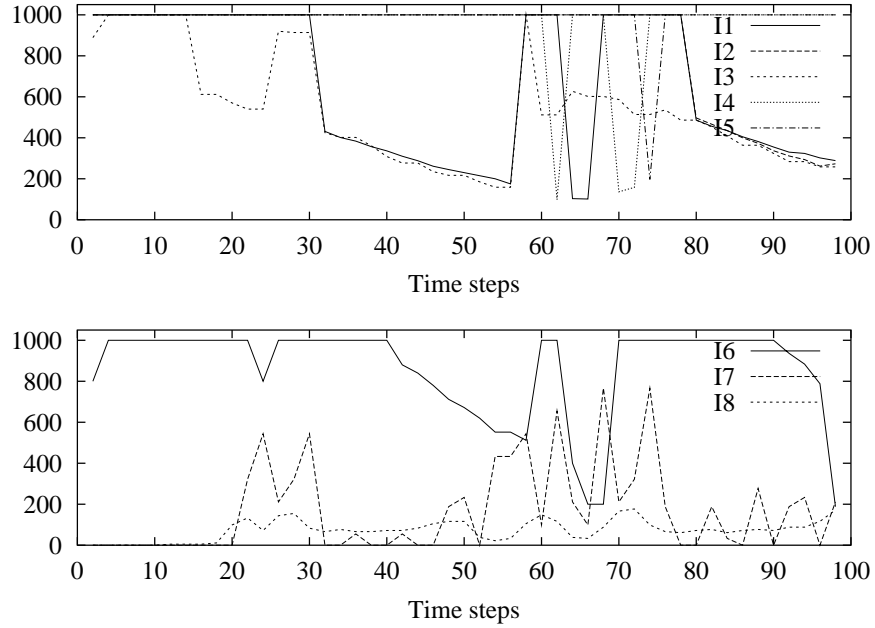
purpose with a version that had a number of deficiencies. Although the robot was able to pass through open doors, its performance at avoiding dynamic obstacles was rather poor and it did not know how to handle doors that were closed. Typical robot trajectories are shown in Figure 3.

During the execution of a high-level go-to action that takes the robot from its current location to a given goal position, we distinguish the following qualitatively different situations: (A) the robot is operating in open space (no obstacles detected), (B) the robot is avoiding an obstacle successfully, (C) the robot gets stuck at an obstacle, (D) the robot is travelling through a door successfully, and (E) the robot bumps into a door and gets stuck. We want the robot to learn to identify these situations based on its sensory data. In this way, it would be possible to predict the outcome of the robot's actions or to use this information to improve the current implementation of the robot controller.

**Robot experiences.** We have used the following set of sensors: the distance to the closest obstacle detected in the sonar-sensor buffers front/left (I1), front/right (I2), front (I3), side/right (I4), and side/left (I5), the translational velocity (I6), the rotational velocity (I7), and the angular deviation from the optimal path (I8). These eight data streams correspond to the inputs to the obstacle avoidance behaviour used in the reactive controller (I1-I5) and to state information of the controller (I6-I8), respectively.

As the experiences are recorded during the execution of high-level plans for office delivery tasks, many different locations are visited during the robot's operation. The durations of the experiences for the go-to action are between 5 and 40 secs. Sequences of about 20 secs are the most frequent. The sampling rate of the sensor values in the given application is 4 Hz.

A single robot experience for the go-to action is shown in Figure 4. At the start, the robot does not detect any obstacles in its sonar buffer and moves at maximum speed. At  $t = 20$ , it performs a 90 deg turn. At  $t = 35$ , it detects an obstacle situated ahead and begins to slow down. At  $t = 55$ , it starts navigating around the obstacle. This procedure



**Fig. 4.** Robot experience with a duration of 25 secs.

is finished at  $t = 70$ . Eventually, the robot stops at the target position. Please note the effect of the sonar distance readings (I1 – I5) on the speed values (I6, I7). As the robot gets very close to the obstacle, it is very hard to distinguish situations in which the obstacle-avoidance procedure is going to succeed or fail. **Iterative clustering.** We have applied the algorithm in Figure 2 to a set of 314 experiences recorded during the execution of high-level go-to actions. In fact, we have used the average-linkage method, binary sensor weights  $w_i$ , and an evaluation function  $q$  with  $k_1 = 0$  and  $k_2 = 1$ . The results are given in Table 1. We obtained the best classification results using the inputs I6, I7, and I8. Using a fractional distance measure, we can predict the outcome of the go-to actions correctly in about 90 % of the cases.

To investigate the performance of the algorithm in more detail, we have manually selected 97 from the original set of 314 experiences. The experiences in this subset correspond to the situation sequences AA, AB, AC, AD, AE, or DA. The performance of the algorithm for these experiences using average linkage, a fractional distance measure, and several subsets of sensors is shown in Figure 5. The clustering process starts with 97 singleton clusters. For most sensor subsets, it terminates at 11 clusters. Using the sensors I6, I7, I8, we can predict the action outcome, again, in about 90 percent of the cases correctly. The 9 cases of misclassifications all refer to situation sequences AB and AC, i.e., the robot failed to identify whether a dynamic obstacle in front of it can be avoided successfully using the given controller. Please note, we have not yet attempted

Distance function	Selected sensors $I_i$ ( $w_i = 1$ )	Number of clusters	Number of misclassifications	Percentage of misclassifications
	6	20	61	19.6
Euclidean ( $k = 2$ )	6, 8	12	103	33.0
	6, 7, 8	36	35	11.3
	1, 2, ..., 8	27	48	15.1
Manhattan ( $k = 1$ )	6, 8	19	106	34.0
	6, 7, 8	44	29	9.3
	1, 2, ..., 8	12	110	35.0
Fractional ( $k = 0.3$ )	6, 8	15	74	23.7
	6, 7, 8	23	28	9.0
	1, 2, ..., 8	23	45	14.4

**Table 1.** Classification results for a set of 314 experiences

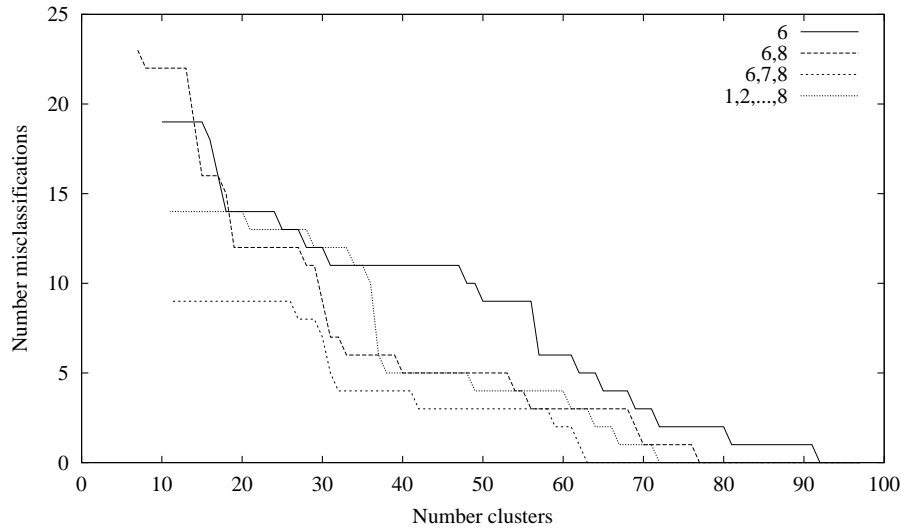
to use the results of the clustering in connection with execution monitoring, e.g., to stop the robot before it bumps into a closed door or dynamic obstacle.

## 5 Conclusions

We have presented a semi-supervised method for clustering robot experiences that is applicable to realistic robot tasks. This extends the method of Oates [14] by (i) using the controller to partially segment the time series, (ii) using the labelled sequences to assess the clustering quality, (iii) using the clustering quality to select when to stop clustering and which sensors to use in the DTW process. The experimental results show that this extends the usefulness of Oates’ approach so that significantly longer time series of double the dimensionality can be handled; and that accurate classifications can be generated.

Finding a suitable clustering of robot experiences is certainly an important first step in building a model of the world dynamics. Given the clusters of experiences, it would be desirable to be able to identify short sequences of sensor data which can be used for the prediction of success or failure of high-level actions. Thus we face the problem of segmenting the time series at a finer level than currently, and detecting dependencies between the subsequences and the high-level action’s outcomes.

The original purpose of the DTW algorithm was to compare one-dimensional time series. In the current approach, we use the rather ad-hoc solution of using the local distance function on the multi-dimensional data items to project the multi-dimensional time series to a one-dimensional one. It is clear that this causes a loss of information, and we have reasons to believe that the most interesting patterns, i.e., those with a good prediction rate, are interdependencies between the different sensory data. The relationship between the distance estimates to obstacles in the different directions yield possibly useful information about the situation the robot is in. Thus, we should also



**Fig. 5.** Hierarchical clustering with a set of 97 experiences using a fractional distance with  $k = 0.3$ . The clustering process moves from right to left. Clustering stopped when the termination criterion based on the difference between the mean intra-cluster and inter-cluster distances was satisfied.

look for alternative ways to deal effectively with the problem of multi-dimensionality of our input data.

## 6 Future Work

We will investigate the problem of segmenting a multi-dimensional numerical time series, incorporating somehow the DTW distance measure. The data mining algorithms capable of dealing with all of these properties are rather rare. A promising approach however seems to be the combination of algorithms for finding frequent sequences in categorical data, like SPADE [17] or FreeSpan [6] and an appropriate categorisation of the numerical multi-dimensional time series. This would provide an easy solution to the problem of the multi-dimensionality, which can be handled in the categorical case. However, this imposes a restriction on the categorisation of the time series used.

The discretised representation of the time series should support a distance measure lower bounding the DTW distance, otherwise there is no possibility of combining the categorical algorithm and the DTW distance measure. There have been some efficient discretisation techniques proposed, among them piecewise constant and piecewise linear approximations [9–11] which despite their simplicity have some remarkable properties in lower bounding several distance measures.

Another approach could be a combination of the discretisation technique used by Das *et al.* [2], sliding a window over the time series and clustering the subsequences

using DTW. This yields a discretised version of the time series, which then can be searched for frequent patterns.

It will have to be subject for future research which of the discretisation techniques proposed in the literature can be used in combination with categorical algorithms for frequent sequence detection to correctly classify the recorded sensor data.

## References

1. C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behaviour of distance metrics in high dimensional space. In *Proc. of 8th Int. Conf. on Database Theory (ICDT-2001)*, pages 4200–434, 2001.
2. G. Das, K.-I. Lin, H. Mannila, et al. Rule discovery from time series. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD-98)*, pages 16–22, 1998.
3. B. S. Everitt. *Cluster analysis*. John Wiley, 1993.
4. G. D. Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Principles of Knowledge Representation and Reasoning*, pages 453–465, 1998.
5. A. Großmann, A. Henschel, and M. Thielscher. A robot control system integrating reactive control, reasoning, and execution monitoring. In *Proceedings of the First International Workshop on Knowledge Representation and Approximate Reasoning (KRAR-2003)*, Olsztyn, Poland, May 2003.
6. J. Han, J. Pei, B. Mortazavi-Asl, et al. FreeSpan: Frequent pattern-projected sequential pattern mining. In *Proc. of the 6th Int. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, pages 20–23, 2000.
7. F. Höppner. Discovery of temporal patterns. Learning rules about the qualitative behaviour of time series. In *Proc. of the 5th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD-2001)*, pages 192–203, 2001.
8. L. P. Kaelbling, T. Oates, N. H. Gardiol, and S. Finney. Learning in worlds with objects. In *Working Notes of the AAAI Stanford Spring Symposium on Learning Grounded Representations*. AAAI, 2001.
9. E. J. Keogh. A fast and robust method for pattern matching in time series databases. In *Proc. of the 9th Int. Conf. on Tools with Artificial Intelligence (ICTAI-97)*, pages 578–584, 1997.
10. E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *Proc. of the IEEE Int. Conf. on Data Mining*, pages 289–296, 2001.
11. E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification clustering and relevance feedback. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD-98)*, pages 239–243, 1998.
12. E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proc. of the 3rd European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD-1999)*, pages 1–11, 1999.
13. E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proc. of the 1st SIAM Int. Conf. on Data Mining (SDM-2001)*, Chicago, IL, USA., 2001.
14. T. Oates, M. D. Schmill, and P. R. Cohen. A method for clustering the experiences of a mobile robot that accords with human judgements. In *Proc. of the 17th National Conf. on Artificial Intelligence (AAAI-2000)*, pages 846–851, 2000.
15. M. T. Rosenstein and P. R. Cohen. Continuous categories for a mobile robot. In *Proc. of the 16th National Conf. on Artificial Intelligence (AAAI-99)*, pages 634–641, 1999.
16. D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. CLSI Publications, 3rd edition, 1999.
17. M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.