Adaptive Virtual Fixtures for Machine-Assisted Teleoperation Tasks

Daniel Aarno, Staffan Ekvall and Danica Kragić
Centre for Autonomous Systems
Computational Vision and Active Perception Laboratory
Royal Institute of Technology (KTH), Stockholm, Sweden
Email: {bishop, ekvall, danik}@nada.kth.se

Abstract— It has been demonstrated in a number of robotic areas how the use of *virtual fixtures* improves task performance both in terms of execution time and overall precision, [1]. However, the fixtures are typically inflexible, resulting in a degraded performance in cases of unexpected obstacles or incorrect fixture models. In this paper, we propose the use of *adaptive virtual fixtures* that enable us to cope with the above problems. A teleoperative or human machine collaborative setting is assumed with the core idea of dividing the task, that the operator is executing, into several subtasks. The operator may remain in each of these subtasks as long as necessary and switch freely between them. Hence, rather than executing a predefined plan, the operator has the ability to avoid unforeseen obstacles and deviate from the model. In our system, the probability that the user is following a certain trajectory (subtask) is estimated and used to automatically adjusts the compliance. Thus, an on-line decision of how to fixture the movement is provided.

I. Introduction

In today's manufacturing industry, large portions of the operation has been automated. However, many processes are too difficult to automate and must rely on humans' supervisory control and decision making; in areas such as the identification of defective parts and process variations [2], or medical applications [3]. When such skills are required, humans still have to perform straining tasks. We believe that Human-Machine Collaborative Systems (HMCS) can be used to prevent ergonomic injuries and operator wear, by allowing cooperation between a human and a robotic system in a flexible way.

HMCS are gaining significant attention in areas such as, for example, medical surgery. In our previous work, [4], we investigated the possibility of segmenting complex tasks such as vitreo-retinal eye surgery into logical components or subtasks, given sensor traces of humans performing the task. It has been demonstrated that, if such segmentation can be obtained, it is possible to enhance the user's performance by choosing appropriate control modes and applying virtual fixtures. In [4], Hidden Markov Models (HMMs) were used for modeling subtasks where the states of the HMMs corresponded to motion primitives, or *gestemes*.

With recent changes in manufacturing such as just-in-time production, outsourcing and rapid process changes, it is necessary to be able to rapidly change the work-flow. Peshkin *et al.* [2] presents the concept of *Cobots* which are simple 2 DOF special purpose human-machine collaborative

manipulation systems that have been used for automotive assembly in the car industry. Although frequently used, the Cobots have the following limitations: i) they are specially designed for one single purpose or task - thus one Cobot is required for every single task, ii) when the assembly task changes, the Cobots have to be reprogrammed to fit the new task, and iii) if new assembly tasks are introduced, new Cobots have to be constructed.

In this paper, we will show how a more flexible design framework can be obtained. In particular, we borrow ideas from the area of Programming by Demonstration (PbD) to build a system where a human-machine collaborative manipulation system can be trained in a fast and easy way to increase human performance. We present a system that uses a standard robotic manipulator to assist humans in various assembly tasks. The ideas presented in this paper are, however, not limited to the area of assembly. We believe that there are natural extensions to other areas.

In our system, a high-level task is segmented to subtasks where each of the subtasks has a virtual fixture obtained from 3D training data. A state sequence analyzer learns what subtasks are more probable to follow each other. This is important for an on-line state estimator which estimates the probability of the user being in a particular state. A specific virtual fixture, corresponding to the most probable state, is then automatically applied.

A. Virtual Fixtures

Virtual fixtures are commonly defined as a task-dependent aid for teleoperative purposes, [5]. They are used to constrain the user's motion in undesired directions while allowing or aiding motion along the desired directions. It has been shown that applying virtual fixtures can enhance both the execution time, quality and precision of teleoperation tasks, [1], [5], [6].

Two major applications where virtual fixtures have been used to increase the performance are:

- **Assembly and High Operator Load Settings**
  Here, virtual fixtures have been used to improve the users' performance in cases where high operator loads are present, [2]. The high load makes the control difficult and may result in injury or long-term wear of the operator. Virtual fixtures are used to "remove" the weight and inertia of the object being manipulated.

- **Surgical Applications**

  Robotics in medical surgery is receiving significant attention and medical robotic applications are becoming more frequent, [3]. Medical robots increase the performance with their superior precision but are still not capable of safe decision-making. The use of virtual fixtures allows the surgeon to move restricted by the fixtures, protecting blood vessels and delicate tissue. Typically, the virtual fixture is generated automatically from 3D-data, obtained from, for example, a CAT-scan.

B. Task Segmentation and State Estimation

In our work, the fixtures are made flexible and adaptive by dividing the task into subtasks. In particular, each subtask is considered to be a straight line and the fixture associated with it is simply the direction of the line. In order to apply the correct fixture, it is then important to estimate what line the operator is currently following, i.e., what is the current *state*. A combination of K-means clustering, Hidden Markov Models and Support Vector Machines are used for state generation, state sequence analysis and associated probability estimation. The segmentation is achieved automatically under the assumption that the task can be divided into states consisting of line tracking. An on-line state estimator then applies a virtual fixture with a guidance factor determined by the probability of the user being in that state. The use of the HMM/SVM approach is motivated by the good generalization over similar but not identical tasks.

This paper is organized as follows. We start with discussion about methods and applications related to virtual fixtures, in Section II. In Section III we describe the theoretical basis for the methods used in the system and in Section IV we explain how these methods are applied. Experimental results are presented in Section V. In Section VI, we conclude and define directions for our future work.

II. RELATED WORK

Virtual fixtures have been shown to enhance users' performance in both teleoperated and HMCS settings on micro- as well as macro-scale. Li *et al.* [7] used virtual fixtures for tracking a sine curve in two dimensions. In [7], a Hidden Markov Model approach was used to estimate whether the user was i) doing nothing, ii) following the curve, or iii) not following the curve. Based on this estimation, the virtual fixture was automatically switched on or off, enabling the user to avoid local obstacles.

In many applications, the fixtures are generated from a predefined model of the task. These approaches work well as long as the trajectory to be followed in the real world is exactly as described by the model. However, for robotic applications, the system must be able to deal with model errors. There may, for example, exist a generic model of a task but in an on-line execution stage, the task at hand will in some way differ from the generic model. In other words, we are interested in performing the same type of tasks in terms of sequencing but the length (type) of each subtask may vary. Therefore, an adaptive approach in which the trajectory is decomposed into straight lines is evaluated in this work. The system constantly estimates which state the user is currently in and aids the execution.

To estimate the state, it is necessary to i) decompose the task into several subtasks, or in our case to different lines, ii) have the ability to recognize the subtasks on-line, and iii) handle deviations from the learned task in a flexible manner. For this purpose, HMMs have been used in various ways to model and detect state changes corresponding to different predefined subtasks, [7], [8]. However, in most cases only one- or two-dimensional input have been considered. In our system, the subtasks are automatically detected given the assumption of motions along straight lines in 3D. Hence, a hybrid HMM/SVM automata is constructed for on-line state probability estimation. The state estimation is then used to choose a virtual fixture.

III. THEORETICAL BACKGROUND

When using adaptive virtual fixtures, it is important to detect the current state in order to apply the correct fixture. A combination of HMMs and SVMs are used for state sequence analysis and probability estimation. This section describes the theoretical background of virtual fixtures and the two methods used for state estimation.

A. Virtual Fixtures

Virtual fixtures act as a guidance that aids the user in controlling the robot. The fixtures constrain the user's motion in undesired directions while allowing motion along desired directions. Virtual fixtures can be applied in several ways. One common approach, that is used in this paper, is to define a *desired* direction $\mathbf{d} \in \mathbb{R}^3$, $\|\mathbf{d}\| = 1$, sometimes also defined as the *span* of the task, [4]. The user's input, which may be of several types such as force, position or velocity control, is transformed to a desired velocity $\mathbf{v_{user}}$. The user's input velocity is then divided into components parallel and perpendicular to the desired direction and scaled by an guidance factor k as shown by (1). The guidance factor determines the compliance of the system. A high value ($\simeq 1$) of k means that the fixture is *hard*, i.e. only motion in the direction of the fixture is allowed. A value of $k = 0.5$ is in our notation equivalent to no fixture at all and supporting isotropic motion. The output velocity $\mathbf{v}$ of the robot is then obtained by scaling $\hat{\mathbf{v}}$ to match the input speed. As mentioned, this is similar to defining *span* and *kernel* operators as proposed in [4], [6].

$$\hat{\mathbf{v}} = \mathbf{proj_d}(\mathbf{v_{user}}) \cdot k + \mathbf{perp_d}(\mathbf{v_{user}}) \cdot (1 - k) \qquad (1)$$

B. Hidden Markov Models

The main idea behind Hidden Markov Models (HMMs) is to integrate a simple and efficient temporal model and the available statistical modeling tools for stationary signals into a mathematical framework. The majority of applications have

1140

been in speech recognition [9] but successful results are also reported in many other fields.

The HMM, denoted by $\lambda = (A, B, \pi)$, is defined by three elements over a collection of N states and M discrete observation symbols. The elements are: the state transition probability matrix A, the observation probability matrix B and the initial state probability vector π.

C. Probability Estimators for HMM

One of the problems with using HMMs is the choice of the probability distribution for estimating the observation probability matrix B. With continuous input, a parametric distribution is often assumed because of issues related to the Baum-Welch method on data sets where $M \gg N$, [10]. This may decrease the performance of the HMM since the real distribution is hidden and the assumption of a parametric distribution is a strong hypothesis on the model [8].

Using probability estimators avoids this problem. These estimators compute the observation symbol probability [11] instead of using a look-up matrix or parametric model. Another advantage with probability estimators is the possibility to use continuous input instead of discrete observation symbols for the HMM. Successful use of probability estimators using multi layer perceptrons (MLP) and Support Vector Machines (SVM) are reported in [11]–[13]. Naturally, the estimators cannot be trained with the Baum-Welch algorithm, but have to be trained separately. In this work, SVMs are used to estimate the observation probabilities.

D. Support Vector Machines

Support Vector Machines (SVM) have been used extensively for pattern classification in a number of research areas [14]–[16]. SVMs have several appealing properties such as fast training, accurate classification and good generalization [17], [18]. In short, SVMs are binary classifiers that separate two classes by an optimal separation hyperplane.

SVMs work with linear separation surfaces in a Hilbert space [17]. However, the input patterns are often not linearly separable, or even defined in such a dot-product space. To overcome this limitation, a "kernel trick" is used to transform the input pattern to a Hilbert space [19]. A map $\phi : \chi \to \mathcal{H}, x \to \mathbf{x}$ is defined for the patterns x from the domain χ. The Hilbert space $\mathcal{H}$ is commonly called the feature space [17].

SVMs try to estimate a function $f : \chi \to \{\pm 1\}$ that classifies the input $x \in \chi$ to one of the two classes ± 1 based on input-output training data. To avoid over-fitting it is imperative to restrict the class of functions that f is chosen from, [17]. Consider the class of hyperplanes $\mathbf{w} \cdot \mathbf{x} + b = 0$, $\mathbf{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$ with the corresponding decision function $f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$. Among all such hyperplanes there exists a unique one that gives the maximum margin of separation between the two classes [17], that is:

$$\max_{\mathbf{w}, b} \left(\min(\|\mathbf{x} - \mathbf{x}_i\| : \mathbf{x} \in \mathbb{R}^N, \ \mathbf{w} \cdot \mathbf{x} + b = 0) \right) \quad (2)$$

The optimal hyperplane may be computed by solving an optimization problem. Using the Lagrangian dual to solve for the maximum margin of separation; it can be shown [17] that the hyperplane decision function can be written as:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{m} y_i \cdot \alpha_i (\mathbf{x} \cdot \mathbf{x}) + b \right) \quad (3)$$

which implies that the solution vector $\mathbf{w}$ has an expansion in terms of a subset of the training samples. The subset is formed by the training samples with a non-zero Lagrange multiplier, α_i. The samples with a non-zero Lagrange multiplier are known as the *support vectors*.

IV. TRAJECTORY ANALYSIS

This section describes the implementation of the virtual fixture learning system. The virtual fixtures are generated automatically from a number of demonstrated tasks. The overall task is decomposed into several subtasks, each with its own virtual fixture.

The first step is to filter the input data. Then, a line fitting step is used to estimate how many lines are required to represent the demonstrated trajectory. An observation probability function learns the probabilities of observing specific 3D-vectors when tracking a certain line. Finally, a state sequence analyzer learns what lines are more probable to follow each other. In summary, the demonstrated trajectories results in a number of support vectors, a HMM and a set of virtual fixtures. The support vectors and the HMM are then used to decide when to apply a certain fixture.

A. Filtering the Input Samples

The input data consist of a set of 3D-coordinates describing the position. From these samples, movement directions are extracted. However, the input samples are quite noisy and have to be filtered. The filtering is achieved by using a dead-zone of 2 cm, the approximate noise level of our input device (see Section V), that makes sure that random noise is not captured. That is, the movement direction is not sampled until the end-effector has moved at least 2 cm since the last stored sample, thus small variations in position are not captured.

B. Finding Straight Lines in the Demonstrated Trajectories

Once the task has been demonstrated, the filtered input data is quantized in order to find straight lines. Since the input data consists of normalized 3D-vectors, K-means clustering [20] may be used to find the lines. The position of a cluster center is equal to the direction of the corresponding line.

Given a trajectory, it is not known beforehand of how many different lines that are suitable for its representation. A search method was implemented that evaluates the result for different number of clusters and then chooses the quantization with the best results. Prior to clustering, two thirds of the data points are stored for validation. These are used to measure how well the current clusters represent unseen data. What is searched for is an optimal number of clusters that maximizes the validation score for the unseen data. In detail, the algorithm first starts with a single cluster. It then gradually increases the number of clusters by one as long as the validation score increases.

When the validation score stops increasing, a local maximum is found. As an extra precaution, the effect of two additional clusters is also evaluated. If the result is still worse, it is assumed that the optimal number of clusters and their positions have been found.

C. Estimating Observation Probabilities Using SVMs

As mentioned previously, SVMs are used for observation probability estimation, instead of using the B matrix. The benefit is that the method does not require discrete observation symbols or an assumption on the probability distribution.

For each state detected by the clustering algorithm, a SVM is trained to distinguish it from all the others (one-vs-all). In order to provide a probability estimation for the HMM, the distance to the margin, from the sample to be evaluated is computed as [8]:

$$f_j(\mathbf{x}) = \sum_i \alpha_i \cdot y_i \cdot \mathbf{x} \cdot \mathbf{x_i} + b \tag{4}$$

where $\mathbf{x}$ is the sample to be evaluated, $\mathbf{x}_i$ is the i-th training sample, $y_i \in \{\pm 1\}$ is the class of $\mathbf{x}_i$ and j denotes the j-th SVM. The distance measure $f_j(\mathbf{x})$ is then transformed to a conditional probability using a sigmoid function (5), [8]. The probability for a state i given a sample $\mathbf{x}$ can then be computed as:

$$P(\text{state } i|\mathbf{x}) = \frac{1}{1 + e^{-\sigma \cdot f_i(\mathbf{x})}} \cdot \prod_{j \neq i}(1 - \frac{1}{1 + e^{-\sigma \cdot f_j(\mathbf{x})}}) \tag{5}$$

The value of σ was empirically chosen to 0.5 for the sigmoid transfer function.

Given the above and applying Bayes' rule, the HMM observation probability $P(\mathbf{x}|\text{state } i)$ may be computed. The SVMs now serve as probability estimators for both the HMM training and state estimation. A radial basis function (RBF) with $\sigma = 2$ was used as the kernel. It is widely known that the standard SVMs do not cope well with outliers. To overcome this, a modified version of SVMs as proposed in [21] is used. The SVMs are trained using the SVM Matlab toolbox [22].

D. State Sequence Analysis Using Hidden Markov Models

We have started by the assumption that the task consists of following a number of lines in a certain order. However, in an on-line execution step, the lines may have different lengths and the lengths may also differ from those provided by the training data. For the sake of simplicity, when a certain line is followed, it is assumed that a corresponding *line state* is active. Thus, there are equally many states as there are line directions.

Given that the user is following a certain line direction, some line types will be more likely to follow after that line type, depending on the task. For this purpose, Hidden Markov Models are used [9] to analyze the transition probabilities for these states.

A fully connected Hidden Markov Model was used to model the task. The number of states is equal to the number of line types found in the training data. The A-matrix is initially set to have probability 0.7 to remain in the same state and a

uniformly distributed probability to switch state. The π vector is set to uniformly distributed probabilities, meaning that all states are equally probable at the start time. For training, the Baum-Welch algorithm is used until stable values are achieved.

E. Using Virtual Fixtures For Line Tracking

With each line, there is an associated virtual fixture defined by the direction of the line. In order to apply the correct fixture, the current state has to be estimated. The system continuously updates the state probability vector $\mathbf{p}$, where $p_i = P(\mathbf{x_k}, \mathbf{x_{k-1}}..., \mathbf{x_1}|\text{state } i)$ is calculated according to (6). The state s with the highest probability p_s is chosen and the virtual fixture corresponding to this state is applied with the guidance $k = \max(0.5, \ p_s \cdot 0.8)$, where $p_s = \max_i\{p_i\}$. As shown in (1), the guidance factor describes how the virtual fixture will constrain the operator's motion. Thus, when unsure which state the user is currently in, the user has full control. On the other hand, when all observations indicate a certain state, the guidance k is 0.8. This automatic adjustment of the guidance factor allows the user to leave the fixture and move freely without having a special "not-following-fixture"-state.

$$
\begin{aligned}
\hat{p}_i &= \begin{cases} \pi_i \cdot P(\mathbf{x}|\text{state } i) & \text{if } \mathbf{p}^{\text{last}} = \mathbf{0} \\ P(\mathbf{x}|\text{state } i) \cdot \sum_j^{Nstates} A_{ij} \cdot p_j^{last} & \text{otherwise} \end{cases} \\
p_i &= \frac{\hat{p}_i}{\sum_k \hat{p}_k}
\end{aligned} \tag{6}
$$

V. EXPERIMENTAL EVALUATION

The system was evaluated with three experiments. The first experiment was a simple trajectory tracking task in a workspace with obstacles, shown in Fig. 1. The second was similar to the first, but the workspace was changed *after* training, in order to test the algorithm's automatic adjustment to similar workspaces. In the last experiment an obstacle was placed right in the path of the trajectory, forcing the operator to leave the fixture. This experiment tested the guidance adjustment as well as the algorithm's ability to cope with unexpected obstacles. A PUMA 560 was used in all experiments.

A. Input Device

In the experiments, a magnetic tracker called Nest of Birds [23] was used. The Nest of Birds consists of an electronics unit, a transmitter and four pose measuring sensors. The sensors measure transmitter-generated magnetic fields. The electronic unit controls the transmitted signals and directs the sensor measurement. From signals measured by the sensors, Nest of Birds calculates the position and orientation of each sensor. Thus, each sensor has six degrees of freedom. Only one sensor was used, to track the hand position of the user. However, the system works equally well with other input modalities. For instance, we have also used a force sensor mounted on the end effector to control the robot.
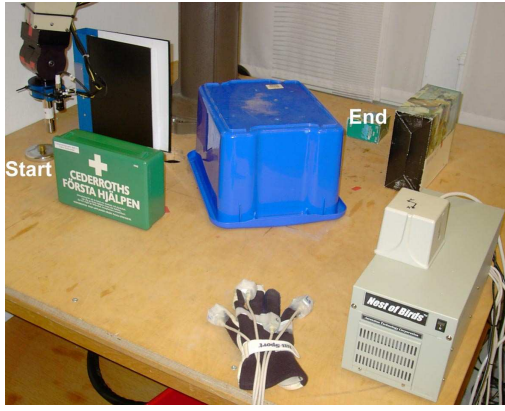
Fig. 1. The workspace used in the first experiment. The Nest of Birds is visible in the lower right corner of the image.

B. Experiment 1: Pick and Place

The first experiment was a simple pick-and-place task in a narrow workspace. The user had to avoid obstacles and move along certain lines to avoid collision. At start, the operator demonstrated the task five times. The system learned from training data, illustrated in Fig. 2(a). From the training data four states were automatically identified.

(a) A training example demonstrated by the user.

(b) Following trajectory using virtual fixtures.

(c) Modified workspace (obstacle size).

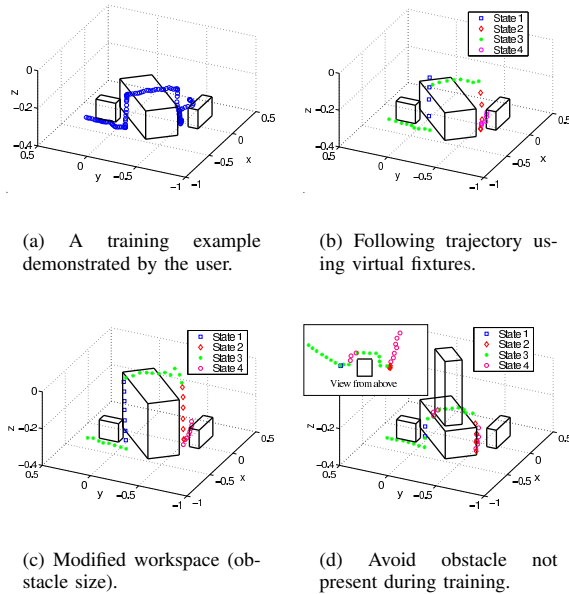(d) Avoid obstacle not present during training.

Fig. 2. End effector position in example workspace. The different symbols (colors) corresponds to the different states recognized by the HMM.

The user then performed the task again, this time aided by the virtual fixtures generated from the training data. The path taken by the PUMA is shown in Fig. 2(b).

For clarity, we also present the state probabilities estimated by the SVM and HMM during task execution, in Fig. 3.

The guidance factor used is also shown. This example clearly demonstrates the ability of the system to successfully segment and repeat the learned task, allowing a flexible state change.
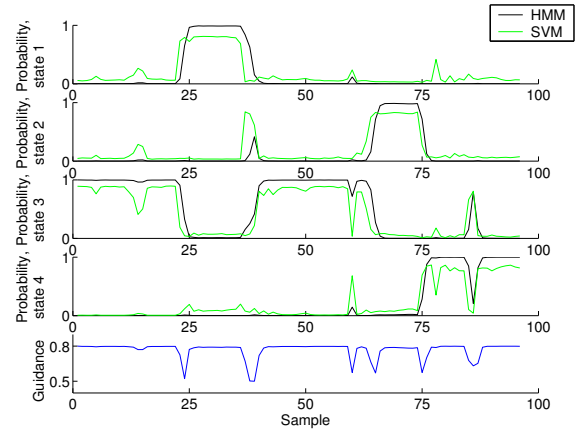

Fig. 3. Estimated probabilities for the different states in experiment 1. Estimates are shown for both the SVM and HMM, the guidance is also shown.

Initially, the end-effector is moving along the y-axis, corresponding to the direction of state 3. Because of deviations from the state direction, the SVM probability is a bit noisy, however the HMM probability remains steady. At sample 24, the user switches direction and starts raising the end-effector. The guidance decreases with the probability for state 3, simplifying the direction change. Then, the probability for state 1, corresponding to movement along the z-axis, increases. In total, the user performed 4 state transitions in the experiment.

C. Experiment 2: Changed Workspace

This experiment demonstrates the ability of the system to deal with a changed workspace. The same training trajectories as in the first experiment are used but the workspace was changed after training.

As it can be seen in Fig. 2(c), the size of the obstacle the user has to avoid has been changed so that the end-effector had to be raised twice as high as in the first experiment. The trajectory generated from the on-line execution shows that this does not introduce any problem for the control algorithm since appropriate guidance is provided at each state. This clearly justifies the proposed approach compared to the work previously reported in [2].

D. Experiment 3: Unexpected Obstacle

The final experiment was conducted in the same workspace as the first one. However, this time a new obstacle was placed right in the path of the learned trajectory, forcing the operator to leave the fixture. Once again, the same training examples as in the previous experiments were used. Figure 2(d) illustrates the path taken in order to avoid the obstacle. The system always identifies the class which corresponds best with input data. However, as seen in Fig. 4, for unseen directions the
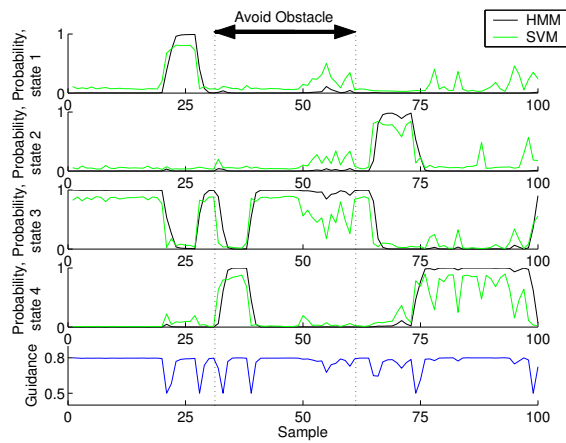
1143

Fig. 4. Estimated probabilities for the different states in the obstacle avoidance experiment 3. Estimates are shown for both the SVM and HMM estimator.

guidance decreases, which enables the operator to regain control and safely avoid the obstacle.

Now, it can be seen that the overall task has changed and that new states were introduced in terms of sequencing. The proposed system not only provides the possibility to perform the task, but can also be used to design a new task model by demonstration if this particular task has to be performed several times.

VI. CONCLUSIONS

In this paper, a system based on the use of *adaptive virtual fixtures* has been proposed. It is widely known that one of the important issues for teleoperative systems is the ability to divide the overall task into subtasks and provide the desired control in each of them. In particular, we have shown that it is possible to use a HMM/SVM hybrid state sequence analyzer on multi-dimensional data to obtain an on-line state estimate that can be used to apply a virtual fixture. Furthermore, the process is automated to allow construction of fixtures and task segmentation from demonstrations, making the system flexible and adaptive by separating the task into subtasks. Hence, model errors and unexpected obstacles are easily dealt with.

In the current design, the algorithm automatically estimates the number of subtasks required to divide the training data. If instead it is possible for the user to manually select the number of states, the algorithm may be expected to perform even better. Such an approach may be, for example, used in medical applications since surgical tasks are expected to be well-defined and known in advance [4]. The experimental results have further shown that two straight lines with almost the same directions are classified as the same straight line. This, however, is not a severe problem as the fixture for this line is close to the optimal fixtures for the individual lines.

Our future work will evaluate this approach in terms of more complex tasks as well as integrate the use of visual feedback to allow further guidance for the user.

REFERENCES

[1] A. B. Kuang, S. Payandeh, B. Zheng, F. Henigman, and C. MacKenzie, "Assembling virtual fixtures for guidance in training environments," in *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 367–374, 2004.

[2] M. A. Peshkin, J. E. Colgate, W. Wannasuphoprasit, C. Moore, R. B. Gillespie, and P. Akella, "Cobot architecture," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 377–390, 2001.

[3] R. H. Taylor and D. Stoianovici, "Medical robotics in computer-integrated surgery," in *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 765 – 781, 2003.

[4] D. Kragić, P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, "Human-Machine Collaborative Systems for Microsurgical Applications," in *International Symposium on Robotics Research*, 2003.

[5] S. Payandeh and Z. Stanisic, "On application of virtual fixtures as an aid for telemanipulation and training," in *10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 18–23, 2002.

[6] P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, "Spatial Motion Contraints: Theory and Demonstrations for Robot Guidance Using Virtual Fixtures," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1954–1959, 2003.

[7] M. Li and A. Okamura, "Recognition of Operator Motions for Real-Time Assistance Usng Virtual Fixtures," in *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2003.

[8] A. Castellani, D. Botturi, M. Bicego, and P. Fiorini, "Hybrid HMM/SVM Model for the Analysis and Segmentation of Teleoperation Tasks," in *IEEE International Conference on Robotics and Automation*, 2004.

[9] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proc. of the IEEE, vol 77, no. 2*, pp. 257–286, 1989.

[10] A. Elgammal, V. Shet, Y. Yacoob, and L. S. Davis, "Learning Dynamics for Exemplar-Based Gesture Recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 571–578, 2003.

[11] H. Bourlard and N. Morgans, "A Continuous Speech Recognition System Embedding MLP into HMM," in *Advances in Neural Information Processing Systems*, vol. 2, 1990.

[12] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[13] A. Ganapathiraju, J. Hamaker, and J. Picone, "Hybrid SVM/HMM Architectures for Speech Recognition," in *Speech Transcription Workshop*, 2000.

[14] D. Roobaert, *Pedagogical Support Vector Learning: A Pure Learning Approach to Object Recognition*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Sweden, 2001.

[15] M. Rychetsky, S. Ortmann, and M.Glesner, "Support Vector Approaches for Engine Knock Detection," in *International Joint Conference on Neural Networks*, 1999.

[16] K. H. P. Haesun, "Prediction of protein relative solvent accessibility with support vector machines and long-range interaction 3D local descriptor," *Proteins: Structure, Function, and Bioinformatics*, pp. 557–562, 2004.

[17] P.-H. Chen, C.-J. Lin, and B. Schlkopf, "A Tutorial on ν-Support Vector Machines." http://www.csie.ntu.edu.tw/˜cjlin/papers/nusvmtutorial.pdf.

[18] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[19] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[20] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *5-th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

[21] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] S. R. Gunn, "MATLAB Support Vector Machine Toolbox." http://www.isis.ecs.soton.ac.uk/isystems/kernel/.

[23] "Nest of Birds." http://www.ascension-tech.com.