Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Hybrid HMM/SVM Model for the Analysis and Segmentation of Teleoperation Tasks

Andrea Castellani, Debora Botturi, Manuele Bicego, Paolo Fiorini
Department of Computer Science
University of Verona, Italy
{castellani,debora,fiorini}@metropolis.sci.univr.it, bicego@sci.univr.it

Abstract— **The automatic execution of a complex task requires the identification of an underlying mental model to derive a possible task control sequence. The model aims at analysing and segmenting the task in simpler sub-tasks. As an example of a complex task, in this paper we consider teleoperation where a person commands a remote robot. This paper presents a new modeling approach using Hidden Markov Models (HMM) and Support Vector Machines (SVM) to analyse the force/torque signals of a teleoperation task. The task is divided into simpler sub-tasks and the model is used to segment the signals in each sub-task. The segmentation gives informations on the system behavior identifying the changes of the model states. Peg in Hole force/torque data are used for testing the model. The results are consistent with the literature with respect to off-line analysis, whereas a significant increase of performance is achieved for on-line analysis.**

I. Introduction

In the last years, different teleoperation systems have been proposed to allow human operators to execute tasks in a variety of applications such as space operation, surgery and underwater maintenance. During task execution it is advisable that a supervisory algorithm analyses the teleoperation data as an additional safety measure. This algorithm should have the ability to monitor the system by using feedback signals. Because of the variability of complex teleoperation tasks (a sequence of simpler but different sub-tasks) the knowledge of the task state could help improving performance. For example, a single control algorithm may not be the most appropriate choice for every sub-task. A better choice could be to use a different control strategy for every sub-task, in this way each controller can be made more precise. To identify the various sub-tasks of a teleoperation, it is necessary to segment the teleoperation data in order to recognize the changes in the task state and to mark them as "jumps" from a sub-task to another.

In literature the problem of data segmentation has been addressed in different ways. In [1] a Hidden Markov Model (HMM) is used to carry out the task segmentation with a number of states equal to the teleoperation sub-tasks. The state transition is computed using the Viterbi algorithm, which returns the more probable state sequence of the HMM, and the parameters of the HMM are computed using the Baum-Welch algorithm or, equivalently, the Expectation Maximization method (EM). However this approach returns the segmentation only in off-line analysis. In [2] a partially Recurrent Neural Network (RNN) with fixed feedback is trained in order to segment the task on-line. This approach produced good results but the use of a neural network hides the use of prior information about the task. In [3], [4] auto-regressive models are presented where the segmentation or the jump between a state and the next is obtained using the Sequential Likelihood Ratio Test (SLRT) technique [5]. This technique is based on work on failure detection [6] and speech segmentation [7].

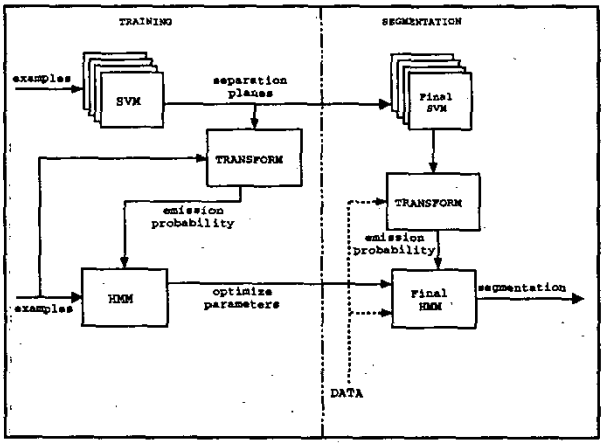

Fig. 1. Hybrid HMM/SVM scheme proposed in this paper, shown are the training phase and the on-line segmentation.

The teleoperation task and the signals produced during a teleoperation task are very unpredictable. They strongly depend on the operator and they are also quite variable when the same operator executes the same task. For these reasons we propose to use the HMM approach, since HMM describe very well signal variability and the sequential aspect of a teleoperation task [8]. A difficulty in using HMM is the choice of probability distribution, typically a parametric distribution. The assumption of parametric distribution can decrease the performances of HMM [9]–[11] because the real distribution is hidden and the choice of a parametric distribution is a strong hypothesis on the model. This probability must be computed outside the HMM framework. For this reason we use a HMM where the emission probability distributions are computed using a Support Vector Machine (SVM) Classifier [12]. SVM represent an instrument that have been intensively

used and that have shown a good classification properties for multidimensional data.

Another key feature of the approach proposed here is that the segmentation can be obtained on-line, as shown in Fig. 1. This is important since the analysis of a teleoperation task can be used to identify the operator performance (computed earlier off-line) but also to recognize the sub-task in execution. This would give an important information to the control algorithm and would be possible only if the segmentation is obtained on-line. The model is tested using the force/torque data of Peg in Hole tasks, performed at NASA-JPL and used also in [1], [2]. It produced good results directly comparable with those of [1], [2]. In particular our HMM/SVM hybrid model returns the average correct segmentation rate of about 100% (off-line) and of about 84% (on-line).

The rest of the paper is organized as follows. Section 2 briefly introduces the two tools that are used in the paper, i.e. HMM and SVM. Section 3 describes our HMM/SVM hybrid model. Section 4 reports the analysis results, and finally in Section 5 the final remarks are summarized.

II. Segmentation Tools

In this Section we introduce the two tools used in our approach: Hidden Markov Models and Support Vector Machine Classifiers.

A. Hidden Markov Models

In a Markov model each state corresponds to an observable event. In many problems this model is too restrictive to be applicable: an HMM is a Markov random process that can not be observed directly. In other words an HMM can be seen as an extension of the Markov model where the states are not observable, and the observation is a probabilistic function of the state. The resulting model is a doubly embedded stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observations. An HMM is formally defined by the following elements [8]:

- A set $S = \{S_1, S_2, ..., S_N\}$ of (hidden) states;
- A state transition probability distribution, also called transition matrix $A = \{a_{ij}\}$, representing the probability to go from state S_i to state S_j

$$a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$$

with $1 \leq i, j, \leq N$, $a_{ij} \geq 0$ and $\sum_{j=1}^{N} a_{ij} = 1$.
- A set $V = \{v_1, v_2, ..., v_M\}$ of observation symbols.
- An observation symbol probability distribution, also called emission matrix $B = \{b_j(k)\}$, indicating the probability of emission of symbol v_k when system state is S_j

$$b_j(k) = P(v_k \text{at time t} \mid q_t = S_j)$$

with $1 \leq j \leq N$, $1 \leq k \leq M$, $b_i(k) \geq 0$ and $\sum_{j=1}^{M} b_j(k) = 1$.
- An initial state probability distribution $\pi = \{\pi_i\}$ representing probabilities of initial states

$$\pi_i = P(q_1 = S_i)$$

with $\pi_i \geq 0$ and $\sum_{i=1}^{N} \pi_i = 1$.

For convenience, we denote an HMM as a triplet $\lambda = (A, B, \pi)$.

The traditional approach in using a HMM is to select the topology type (Fig. 2) and compute the parameters (A, B, π) using the Baum-Welch algorithm [8]. Then, the Markov model is completely defined.

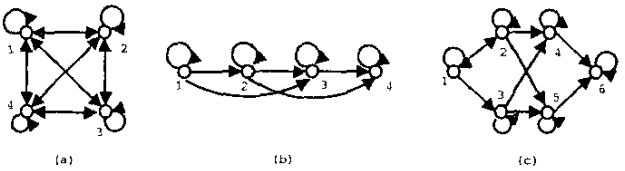

Fig. 2. Three possible types of HMM. (a) A 4-state ergodic model. (b) A 4-state left-right model (c) A 6-state parallel path left-right model.

B. Support Vector Machines

The Support Vector Machines, introduced at the end of '70 [13], [14], are classifiers that have been intensively used [15]–[20]. They have several strengths: fast training by using specific algorithms [21], [22], accurate classification and, at the same time, high performance of generalization, i.e. the ability to learn the trend and the regularity of the data. A nice introduction of SVM for pattern recognition is reported in [12].

The basic objective of the SVM training is to find the optimal separation hyper-plane that minimizes the expected classification error, which is equal to maximizing the distance of the points from the margin (Fig. 3).

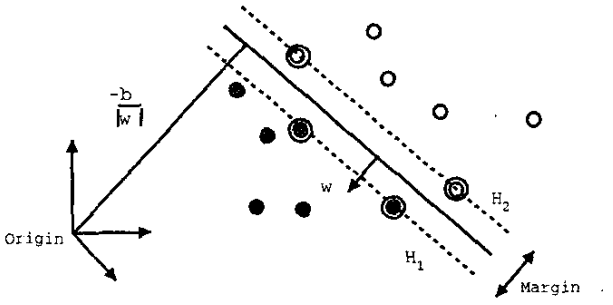

Fig. 3. Separation plane of two example sets (black and white points).

The use of the so called "Kernel trick" [12] permits to define linear separation surfaces in a larger dimensional space that becomes highly non linear in the original space. In practice, it defines a "map" Φ that transforms the vectors to a space where they can be more easily separable by a linear classifier, by using the Kernel $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y})$. The use of this

Kernel permits to obtain a highly non linear separation surface with the same computational cost of a linear separation. The classification function is represented by a linear combination of Kernels K applied to the training data $\{\mathbf{x}_i\}$ with class labels $\{y_i\}$:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x_i}) + b; \qquad (1)$$

where α_i are the Lagrange multipliers. The elements $\mathbf{x}_i$ with non null multiplier are called *Support Vectors* (SV) and are the only ones that concur to the definition of the separation surface, represented with circles around points in Fig. 3.

A SVM is a binary classifier. In the case of more classes, two differents strategies are possible: "one vs. one" and "one vs. all". In the first case one SVM for each pair of classes is constructed; an element $\mathbf{x}_i$ belongs to the class that produces the most positive output. In the second case one SVM for each class is constructed, in order to separate one class from the others. Once the training is completed the estimate of function (1) permits to decide the proper class of each data point. As described more in detail in [12], [23] the number of SV, generally, is low with respect to the number of examples. There are however non trivial cases, in which such set is not minimal and it can further reduced to benefit the testing speed; this is important in our case during the segmentation phase by Viterbi algorithm. For this purpose we have used the method described in [24] that we introduce in the following.

Suppose we train an SVM classifier with pattern vectors $\mathbf{x}_i$ and that r of these are determined to be SV with separation surface described by equation (1). We want to find all the SV linearly dependent, with the correspondent dependency coefficients c_i, to remove them from the SV set and to update the Lagrange multipliers of the remaining SV. In practice equation (1) becomes:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{\substack{i=1 \\ i\neq k}}^{r} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + \alpha_k y_k \sum_{\substack{i=1 \\ i\neq k}}^{r} c_i K(\mathbf{x}, \mathbf{x}_i) + b \\ &= \sum_{\substack{i=1 \\ i\neq k}}^{r} \alpha_i (1+\gamma_i) y_i K(\mathbf{x}, \mathbf{x}_i) + b \\ &= \sum_{\substack{i=1 \\ i\neq k}}^{r} \alpha_i^{\dagger} y_i K(\mathbf{x}, \mathbf{x}_i) + b \qquad (2) \end{aligned}$$

where $\gamma_i = \frac{\alpha_k y_k c_i}{\alpha_i y_i}$, $\alpha_i^{\dagger} = \alpha_i(1+\gamma_i)$ and k is the index of the linearly dependent SV.

From this equation we see that the linearly dependent vectors are not required for the representation of the separation surface and therefore in the testing phase. We see, instead, that the Lagrange multipliers must be modified in order to obtain such a simplified representation. However, this is a very simple modification that can be applied to every linearly dependent SV. Thanks to this method, it is possible to reduce the number of SV without modifying the performance of the classifier.

During a teleoperation task, signals are sequentially generated. These signals are measured by the sensors on the robot and give a great amount of information about the teleoperation state: force/torque contact data, position/velocity/acceleration of the robot joint, images record by a camera and so on. The SVM are good classifier but do not account for temporal information, since they are static classifiers. For this reason, we can not use this classifiers to segment teleoperation signal sequences. In this paper we propose to develop a hybrid model that use both HMM and SVM.

III. HMM/SVM HYBRID MODEL

We choose to define the HMM with one state for every sub-task in which the teleoperation task can be partitioned. The segmentation is obtained from the analysis of the passage between a state and the other of the HMM automaton (Viterbi algorithm). In practice the current state of HMM defines the sub-task in execution. The main points of the our hybrid model refer to the use of the SVM and the definition of an algorithm that gives back the segmentation during the task execution, and not off-line, at the end of data sequence, like standard HMM.

A. Emission Probability with SVM

Here we have used SVM classifiers to generate the HMM emission distribution probability in the training phase and in the segmentation phase. Firstly, we train one SVM for every sub-task signal (one vs. all). This produces one separation surface (equation 1) for each sub-task.

The function $f(\mathbf{x})$ that describes the separation plane, measures the distance of the element $\mathbf{x}$ from the margin. To produce a distribution probability from this function two considerations are important:

1) The function $sign(f(\mathbf{x}))$ defines whether the pattern $\mathbf{x}$ belongs or not to the class ($f(\mathbf{x}) > 0 \Rightarrow \mathbf{x} \in class$ and $f(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \notin class$).
2) The distance from the margin is proportional to the probability that the element belongs to this class. If the point is near the margin, then the probability of belonging to the class is low. If the point is far from the margin then it has a higher probability of belonging to the class.

For this reason we have used a sigmoid (Fig. 4) to transform the distance measure $f(\mathbf{x})$ into the conditional probability $P(class \mid \mathbf{x})$ and then, using Bayes' theorem, in the HMM emission probability $P(\mathbf{x} \mid class)$:

$$\begin{aligned} P(class \mid element) &= P(j \mid \mathbf{x}) \\ &= \frac{1}{1+e^{-kf_j(\mathbf{x})}} \\ P(element \mid class) &= P(\mathbf{x} \mid j) \\ &= \frac{P(j \mid \mathbf{x}) \cdot P(\mathbf{x})}{P(j)} \qquad (3) \end{aligned}$$

where $P(j)$ is the probability of the class j and f_j is the function that identifies the class j from the others and that returns the distance from the margin.

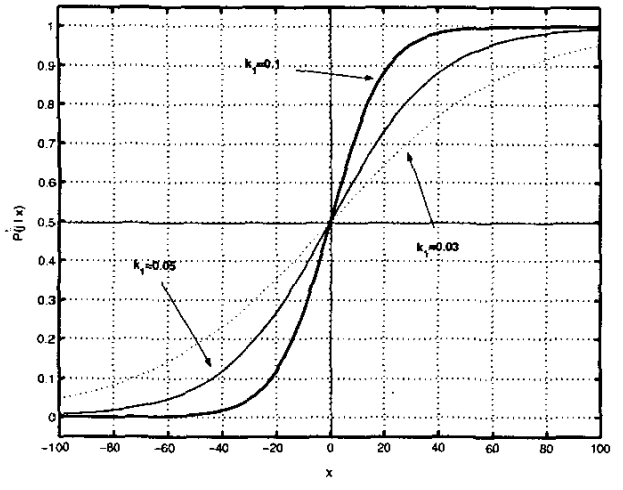

Fig. 4. Sigmoid function used to transform SVM measure into the conditional probability $P(j \mid \mathbf{x})$. k describes the trend of the sigmoid function.

The use of a sigmoid transforms a distance measure into a probability distribution and permits to satisfy the consideration 1,2 described above. For this reason, this function is used in several studies, an example can be found in [25].

B. Segmentation

The Viterbi algorithm segments the data when the whole data sequence is available. The interest in real time segmentation of telerobotics tasks has lead to the definition of an algorithm that computes the most probable state (sub-task) during the execution of the task. This algorithm samples the normal Viterbi algorithm (Viterbi Standard (VS)), computing every t samples the partial result of the process without backtracking (Viterbi Sampled (VC)). In the VS algorithm the variable $\delta_t(i)$ represents the probability that the model is in the state j at the time t. The VC algorithm, at the time t, chooses the state (sub-task) q_t of maximum probability:

$$q_t = arg \max_{1 \le i \le N} [\delta_t(j)]. \quad (4)$$

VS and VC are not equivalent and generally produce different results. This is because VS computes the optimal sequence of states at the end of the sequence, by back-tracking on the whole data-set. This is not equivalent to consider, for every t, the maximum δ_t. In fact, VC finds the optimal solution at each time t. For this reason VS is generally more precise, while the performances of VC strongly depends on the HMM and SVM training phases.

IV. Data used for testing

In order to test our HMM/SVM hybrid model we have used a typical *"Peg in Hole"* telerobotic task. Such task consists of inserting and subsequently extracting a peg from a hole. The force and torque data used have been collected from two previous experiments described in [1], [2] and whose data are shown in Fig. 5. The experiments were carried out at NASA-JPL using a PUMA manipulator equipped with a "Smart" Hand, and a Force Reflecting Hand Controller. Some of experiments and the equipment setup is described in [1], [2].

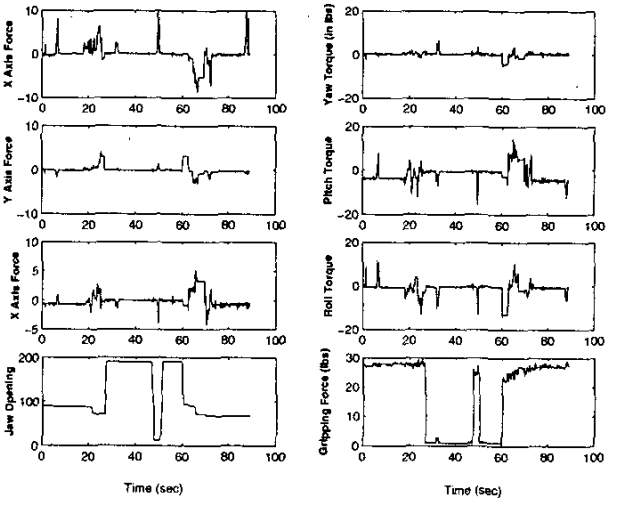

Fig. 5. Force and Torque data during an example of Peg in Hole task. The data are the forces in x, y, z directions, the torque in $x, y, , z$ directions, the jaw opening and the gripping force.

The task can be subdivided into a sequence of four sub-tasks (*move, tap, insert, extract*). This subdivision can be observed by analyzing the force signal in the x direction in Fig. 6. The complete sequence of sub-task is *move, tap, move, insert, move, tap, move, extract, move, tap, move* also used in [1]. We observe that in correspondence to different sub-tasks the force signals are quite different.

V. Results

In order to test the model, we have implemented the HMM algorithms with *MatLab* [26], whereas for SVM we have used a free *SVM MatLab* toolbox [27].

For analysis and segmentation we have used only the force and torque signals in x direction. Table I shows the SVM training results (4 SVM, one for every different sub-task). The last column in the table represents the final number of linearly independent Support Vectors with a reduction of the 12.21% respect to the whole set of Support Vectors ("N. Example" are the number of training examples, "Time" represent the length of training in seconds, "N. SV" and "Min." represent respectively the total number of Support Vectors and the numbers of linearly independent Support Vectors). For all the training runs we have used a Gaussian "Radial Basis Function" Kernel (Rbf) with $\sigma = 2$:

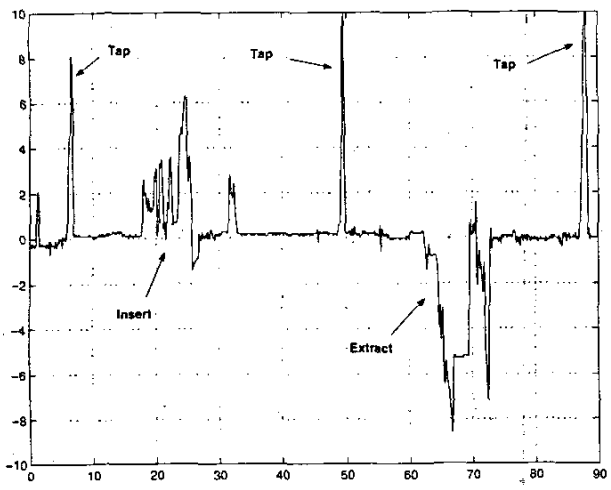

Fig. 6. Force signal in x direction and the identification of *tap*, *insert* and *extract* sub-task.

$$K(\mathbf{x}, \mathbf{y}) = e^{\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}. \quad (5)$$

This type of Kernel is used in several SVM applications, as described in [16], [17], [28].

TABLE I
SVM TRAINING.

Sub-task	N. Example	Time	N. SV.	Min.
move	766	715	295	286
tap	716	540	167	128
insert	760	935	450	406
extract	744	618	403	352

Once the SVM training is completed, the second step is to execute the HMM training. For HMM we have used a 11-state left-right model [1] corresponding to the Peg in Hole subdivision.

The results of the segmentation with VS (off-line segmentation) and VC (on-line segmentation) algorithms for a few examples are shown in Table II. VS obtains an exact segmentation in all cases (off-line), whereas VC obtains 84% of correct segmentation (on-line). The results of segmentations computed by VS and VC algorithms using only data in the x force direction are shown in Fig. 7.

These results can be directly compared with the results presented in [1], [2].

- We achieved equal results in off-line segmentation, but in [1] no on-line segmentation is performed.
- In on-line segmentation HMM/SVM returns an average correct segmentation of about 84% versus the 64% obtained in [2].

Moreover, the VC algorithm returns in real time a control value (*Guard*) that represents how the hidden Markov model

TABLE II
SEGMENTATION RESULTS ON FIVE EXAMPLES.

Sequence	Seq. length	VS	VC
first	889	100%	92.58%
second	249	100%	89.16%
third	259	100%	88.42%
fourth	369	100%	89.16%
fifth	449	100%	63.47%
Exact Segmentation		100%	84%

follows the teleoperation data. This is equivalent to the value of P defined in [8]. This value represents a good index to estimate operators training, and a good safety value to pass to a control algorithm (the more the data are far away from the model and the more this value decreases). In Fig. 8 we see the behavior of the *Guard* in an example with and without noise: we could note that this parameter changes if we introduce a noise sequence.

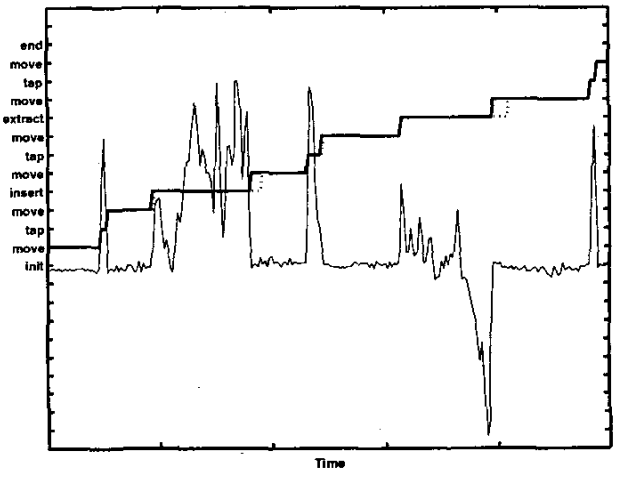

Fig. 7. Peg in Hole segmentation example. Solid line represent the VS segmentation while dotted line represent the VC segmentation, each step indicates a state transition (sub-task).

As shown in Fig. 8, the insertion of wrong data, for example in case of operator execution error, decreases drastically the value of the *Guard* because the signal does not follow the HMM model. This can be a good warning to detect signal changes and operator performance. The idea to define "*guards*" based on the analysis of the data measured from the sensors can be seen also in [29] ("*sensory-guard*").

VI. CONCLUSIONS

In this paper, an approach aimed at modeling and segmenting teleoperation tasks is presented. A new HMM/SVM hybrid model is defined and tested on a Peg in Hole teleoperation task. In detail:

- We use a HMM model to describe a typical sequential teleoperation task.

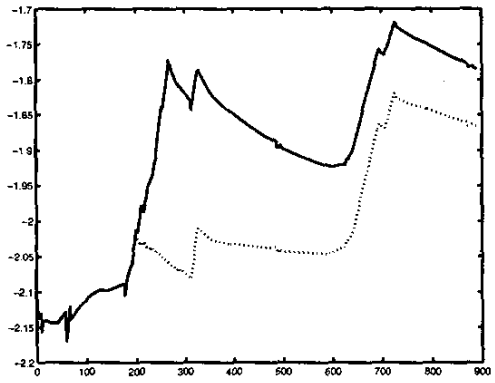

Fig. 8. The *Guard* progress. Solid line represent the *Guard* on real signal while the dotted line represent the *Guard* on modified signal (we have exchanged the real signal with noise in a central interval, from [200, 299]).

- We have overcome limitations of the HMM emission distribution probability using SVM as "probability generators".
- We use dimensional reduction [24] to decrease the number of support vectors to speed up the classification phase and then the segmentation of signals.
- We have developed a new algorithm, based on the standard Viterbi algorithm, to segment experimental data online (VC).

The results can be compared with those presented in the literature, since we have used the same data sets. In particular VS can be compared with the HMM in [1]: in both cases the correct off-line segmentation is 100%. VC can be confronted with the Neural Network in [2] that produced correct online segmentation of about 65% versus 84% of the new HMM/SVM hybrid model.

The long-term objective of this research is to improve our model using more complex HMM and a new method [23] to increase the SVM classification performance (*Virtual Support Vector method*). Finally, we propose to use our model to analyse more complex tasks, especially in robotic surgery. Among these tasks, a special interest will be placed on applying our hybrid model to the *suture* surgical task, both as input for a possible teleoperation control algorithm, and as the basis for a future automatic task execution.

ACKNOWLEDGMENT

The authors would like to thank NASA-JPL for the use of the experimental data of the Peg in Hole experiments.

REFERENCES

[1] B. Hannaford and P. Lee, "Hidden Markov Model Analysis of Force/Torque Information in Telemanipulation," *The International Journal of Robotics Research*, vol. 10, no. 5, pp. 528–539, October 1991.

[2] P. Fiorini, A. Giancaspro, S. Losito, and G. Pasquariello, "Neural Networks for the Segmentation of Teleoperation Tasks," *Presence. Teleoperators and Virtual Environments*, vol. 2, no. 1, pp. 1–13, 1993.

[3] B. Eberman and J. K. Salisbury, "Application of Change Detection to Dynamic Contact Sensing," *The International Journal of Robotics Research*, vol. 13, no. 5, pp. 369–394, October 1994.

[4] B. Eberman, "A Model-Based Approach to Cartesian Manipulation Contact Sensing," *The International Journal for Robotics Research*, vol. 16, no. 4, 1997.

[5] M. Bassèville, "Detecting changes in signals and systems - a survey," *Automatica*, vol. 24, no. 3, pp. 309–326, 1988.

[6] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, no. 6, pp. 601–611, 1974.

[7] R. Andre-Obrecht, "A new statistical approach for the automatic segmentation of continuous speech signals," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 36, no. 1, pp. 29–40, January 1988.

[8] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proceedings of the IEEE*, vol. 77(2), February 1989, pp. 257–286.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood Estimation from Incomplete Data," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1997.

[10] G. McLachlan, *The EM Algorithm and Extensions*. John Wiley, 1997.

[11] R. A. Redner and H. F. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review*, vol. 26, no. 2, pp. 195–239, 1984.

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

[13] V. Vapnik, "Estimation of Dependences Based on Empirical Data," *Nauka, Moscow. (English translation: 1982, Springer Verlag, New York)*, 1979.

[14] ——, "The Nature of Statistical Learning Theory," *Springer-Verlag*, 1995.

[15] V. Blanz, B. Schülkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter, "Comparison of View-Based Object Recognition Algorithms using Realistic 3D Models," *Lecture Notes in Computer Science: Artificial Neural Networks*, vol. 1112, 1996.

[16] E. Osuna, F. Girosi, and E. Freund, "Training Support Vector Machines: an application to Face Detection," in *In IEEE Conference on Computer Vision and Pattern Recognition*, 17-19 June 1997, pp. 130–136.

[17] P. Clarkson and P. J. Moreno, "On the use of Support Vector Machines for Phonetic Classification," in *Proceedings ICASSP*, vol. 2, Civic Plaza, Hyatt Regency - Phoenix, Arizona, May 15-19 1999, pp. 585–588.

[18] M. Schmidt, "Identifying Speaker with Support Vector Networks," in *In Interface '96 Proceedings*, Sydney, 1996.

[19] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization," in *Proceedings of the Seventh International Conference on Information and Knowledge Management*, Washington D.C, November 1998, pp. 148–155.

[20] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proceedings of the European Conference on Machine Learning*, vol. 1398, Chemnitz, DE, 1998, pp. 137–142.

[21] C. Cortes and V. Vapnik, "Support Vector Network," *Machine learning*, vol. 20, pp. 273–297, 1995.

[22] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft Research, Technical Report MSR-TR-98-14, Tech. Rep., April 21 1998.

[23] C. J. C. Burges and B. Schoelkopf, "Improving speed and accuracy of Support Vector Learning Machines," *Advances in Neural Information Processing System*, vol. 9, pp. 375–381, 1997.

[24] T. Downs, K. E. Gates, and A. Masters, "Exact Simplification of Support Vector Solutions," *Journal of Machine Learning Research*, vol. 2, pp. 293–297, December 2001.

[25] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist Probability Estimators in HMM Speech Recognition," *IEEE Transaction on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–175, January 1994.

[26] T. M. Inc., "Matlab: The language of technical computing. version 6.0.0.88 release 12," http://www.mathworks.com, 22 September 2000.

[27] S. R. Gunn, "SVM Toolbox for Matlab," http://www.isis.ecs.soton.ac.uk/isystems/kernel/, March 1998.

[28] M. Pontil and A. Verri, "Support Vector Machines for 3D Object Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 637–646, June 1998.

[29] M. Brooks, "Sensor-Based Task Control for Telerobotic Manipulation," in *Proceedings of the International Symposium on Advanced Robot Technology*, March 1991.