# Motion Intention Recognition in Robot Assisted Applications

Daniel Aarno and Danica Kragic *

*Computational Vision and Active Perception*
*Centre for Autonomous Systems*
*School of Computer Science and Communication*
*Royal Institute of Technology, Stockholm, Sweden*

**Abstract**

Acquiring, representing and modeling human skills is one of the key research areas in teleoperation, programming-by-demonstration and human-machine collaborative settings. The problems is challenging mainly because of the lack of a general mathematical model to describe human skills. One of the common approaches is to divide the task that the operator is executing into several subtasks or low-level subsystems in order to provide manageable modeling.

In this paper we consider the use of a Layered Hidden Markov Model (LHMM) to model human skills. We evaluate a gesteme classifier that classifies motions into basic action-primitives, or gestemes. The gesteme classifiers are then used in a LHMM to model a tele-operated task. The proposed methodology uses three different HMM models at the gesteme level: one-dimensional HMM, multi-dimensional HMM and multi-dimensional HMM with Fourier transform. The online and off-line classification performance of these three models is evaluated with respect to the number of gestemes, the influence of the number of training samples, the effect of noise and the effect of the number of observation symbols. We also apply the LHMM to data recorded during the execution of a trajectory tracking task in 2D and 3D with a mobile manipulator in order to provide qualitative as well as quantitative results for the proposed approach.

The results indicate that the LHMM is suitable for modeling teleoperative trajectory-tracking tasks and that the difference in classification performance between one and multi dimensional HMMs for gesteme classification is small. It can also be seen that the LHMM is robust with respect to misclassifications in the underlying gesteme classifiers.

*Key words:*
Layered Hidden Markov Models, human-machine collaboration, motion intention recognition

## 1 Introduction

Learning and recognizing human skills is an important research problem in teleoperation, programming-by-demonstration (PbD), Human-Machine Collaborative systems (HMCS) and human-computer interaction [1–12]. One of the strongest motivations for this work is that, if the intention can be recognized online in real-time, it is possible to improve the task execution by allowing the system to adapt to the operator's need by applying the correct control mode in the transfer step [3, 6, 7]. It has been widely recognized that the underlying system used for learning, representing, modeling and transferring skills have to deal with highly nonlinear relationships between the stimuli and responses (sensor/actuator systems). Learning human skills has been viewed as the problem of extracting specific skill characteristics given training data. In robotics community, hidden Markov models (HMMs) have been a popular method used for interpreting a human operator's intention during execution of a teleoperated or human-machine collaborative tasks, [1, 3–7]. HMMs have also been frequently and successfully used for speech recognition, [13]. Other areas where HMMs have shown good results are that of handwritten character recognition, [14] and gesture recognition for interpreting sign language, [11].

The problem studied in this paper is operator intention recognition in a teleoperated or human-machine collaborative system (HMCS). To be able to give the correct aid to the operator it is necessary for the system to be able to successfully interpret the operator's intent, online and in real-time. For example, robotics in medical surgery is receiving significant attention. Medical robots increase the performance with their superior precision but are still not capable of safe decision-making. To provide smooth and safe control in medical collaborative settings, provided that the intent of the operator can be recognized, the performance of the system can be increased by applying the correct type of force scaling and control modes [3]. In general, given a set of control primitives generated in the learning step, the online recognition step is responsible of choosing the most likely mental state/intention given a set of measurements. Hidden Markov models are suitable for modeling manipulation tasks because they capture the (often sequential) mental model that the operator uses. In addition,there is a natural correspondence between the states of the HMM and the subtasks most manipulation tasks can be decomposed of. HMMs are also suitable for modeling basic motion primitives because of their inherent sequential nature.

Although HMMs have been used successfully to model both motion primitives and complete teleoperative tasks, most work presented so far have only dealt with simple problems where either the number of primitives or tasks was low and they were very distinct. Our current work aims to extend motion intention recognition using

* Corresponding author.
*Email address:* bishop, dani@kth.se (Daniel Aarno and Danica Kragic).

HMMs to more complicated tasks in 3D with a larger set of primitives. Although a straightforward approach may be to model every high-level task with a single complex HMM, we are interested in learning hierarchical representations of tasks for which the low-level set of skill primitives is common. The particular motivation for this type of reasoning is our belief that ideas from teleoperative and HMCS are applicable and therefore very relevant for Programming by Demonstration (PbD) settings. In a PbD task, models are built by observing a human that performs it and a set of sensory-motor primitives is used to build a high level task. Therefore, our particular interest is on how to build complex tasks given a set of low-level primitives. The aim of this work is therefore to investigate which parameters determine the success of the HMM approach to motion intention recognition as well as present the Layered Hidden Markov Model (LHMM) approach we are currently investigating.

This paper is organized as follows: section 2 describes related work in the area of task modeling and intention recognition. In section 3 the concept of the layered hidden Markov model is introduced and some of the challenges are presented and section 4 describes the experimental setup used to evaluate the LHMM. Experimental results are presented in section 5 and finally the paper is summarized and conclusions given in section 6.

## 2 Related Work

Hidden Markov models have been used to interpret human intention in various human-machine collaborative settings. Li *et al.* used virtual fixtures for tracking a sine curve in two dimensions, [3]. A HMM 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 estimate, the virtual fixture was automatically switched on or off, enabling the user to avoid local obstacles. Nolin *et al.* demonstrated different ways of setting the compliance level, depending on how well the user was following the fixture, [15]. Three different compliance behaviors were evaluated: *toggle, fade* and *hold*. The results show that the *fade* behavior, which linearly decreases the compliance with the distance from the fixture, achieves best results (least tracking error) when using automatic detection of whether the user is following the fixture. Marayong *et al.* also investigated the effect of different compliance levels for a sine tracking task, [16]. In [6], Aarno *et al.* adjust the compliance in a similar way. However, instead of using the distance to the fixture, the probability that the user is following the fixture is used as a basis for setting the compliance. Yu *et al.* used HMMs to classify an operator's motion intention to three classes, path following, target alignment and obstacle avoidance, [7]. Each class is associated with a virtual fixture that assists the operator.

In our previous work a combination of k-means clustering, SVMs and HMMs was

used to automatically extract a set of virtual fixtures given sensor traces of an operator performing a task, segment the task into a number of subtasks, corresponding to a particular fixture and provide online assistance by applying the correct fixture during subsequent task executions, [6]. The output of the HMM was used to adjust the compliance of the virtual fixture so that the fixture was harder when the system was more certain about the current state. This allowed the system to handle task-deviations (i.e. none of the subtasks were executed) by lowering the stiffness of the fixture. However, the subtasks used in [6] were limited to straight lines. In [1], Hundtofte *et al.* used HMMs at the *gesteme* level as opposed to the task level. This means that basic interaction primitives are modeled by a HMM and the task is represented as a network of such HMMs. In our current work we combine gesteme classification with task-level modeling by the suggested LHMM approach so to handle more complicated types of tasks. This is an extension of the work presented in [6] where the SVM classifiers are replaced by the more expressive HMM classifiers.

The work presented above has been mostly concerned with modeling motion primitives in teleoperated/HMCS tasks. In our current work we are integrating the gesteme classification with the higher level task modeling. Hierarchical Hidden Markov Models (HHMMs) and layered hidden Markov models (LHMMs) have been used to model various phenomena that exhibit stochastic structures at several different levels in areas such as speech and text recognition, modeling of group actions in meetings and extracting context from video, [12, 17–20]. Zhang *et al.* used a two-layer HMM to model individual and group actions during meetings in [18]. An I-HMM was used to model individual actions. The recognized individual actions was then passed along to the G-HMM that was used to classify group actions. In [12], a LHMM was used to recognize different types of activity in an office environment. In [19] Xie *et al.* used a HHMM to automatically segment a soccer game into two classes, *pause* and *play* in an unsupervised setting.

In this work, we are considering a LHMM approach to user intention recognition where the sensory information is sampled motion data from a pose measuring sensor. The LHMM is preferable over the HHMM since in the HHMM the states contain another HMM and thus represents a time sequence of the raw signals. On the other hand, in the LHMM there are several HMMs running in parallel at any given level of the hierarchy, where each HMM corresponds to a different “concept”.

## 3 Theoretical Background

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. An HMM, [21], 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 $A$ that determines the probability of moving from state $i$ to state $j$, the observation probability matrix $B$ that gives the probability of observing a specific observation symbol given that the system is in state $i$ (i.e $P(o|i)$) and the initial state probability vector $\pi$.

### *3.1 Layered Hidden Markov Models*

Hidden Markov models can be used on two levels for modeling human actions. A HMM can be used to recognize the operator's motion primitives, or *gestemes* as in [1] or to model the mental stages of the operator performing a teleoperation task as in [22]. A gesteme-level HMM is used to recognize a primitive motion sequence and a task-level HMM is used to recognize a complete task.

In our approach, a layered hidden Markov model (LHMM) consists of $N$ levels of HMMs where the HMMs on level $N+1$ corresponds to observation symbols or probability generators at level $N$. Every level $i$ of the LHMM consists of $K_i$ HMMs running in parallel, Fig. 1. At any given level $L$ in the LHMM a sequence of $T_L$ observation symbols $\mathbf{o}_L = \{o_1, o_2, ..., o_{T_L}\}$ can be used to classify the input into one of $K_L$ classes, where each class corresponds to each of the $K_L$ HMMs at level $L$. This classification can then be used to generate a new observation for the level $L-1$ HMMs. At the lowest layer, i.e. level $N$, primitive observation symbols $\mathbf{o}_p = \{o_1, o_2, ..., o_{T_p}\}$ would be generated directly from observations of the modeled process. For example, in a trajectory tracking task, the primitive observation symbols would originate from the quantized sensor values. Thus at each layer in the LHMM, the observations originate from the classification of the previous layer, except for the lowest layer where the observation symbols originate from measurements of the observed process.

It is not necessary to run all levels at the same level of granularity. For example, it is possible to use windowing at any level in the structure so that the classification takes the average of several classifications into consideration before passing the results up the layers of the LHMM. Instead of simply using the winning HMM at level $L+1$ as an input symbol for the HMM at level $L$, it is possible to use it as a probability generator by passing the complete probability distribution up the layers of the LHMM. Hence, instead of having a "winner takes all" strategy where the most probable HMM is selected as an observation symbol, the likelihood $L(i)$ of observing the $i$-th HMM can be used in the recursion formula of the level $L$ HMM to account for the uncertainty in the classification of the HMMs at level $L+1$. Thus, if the classification of the HMMs at level $n+1$ is uncertain, it is possible to pay more attention to the a-priori information encoded in the HMM at level $L$.

A LHMM could in practice be transformed into a single layered HMM where all the different models are concatenated together. Some of the advantages that may be

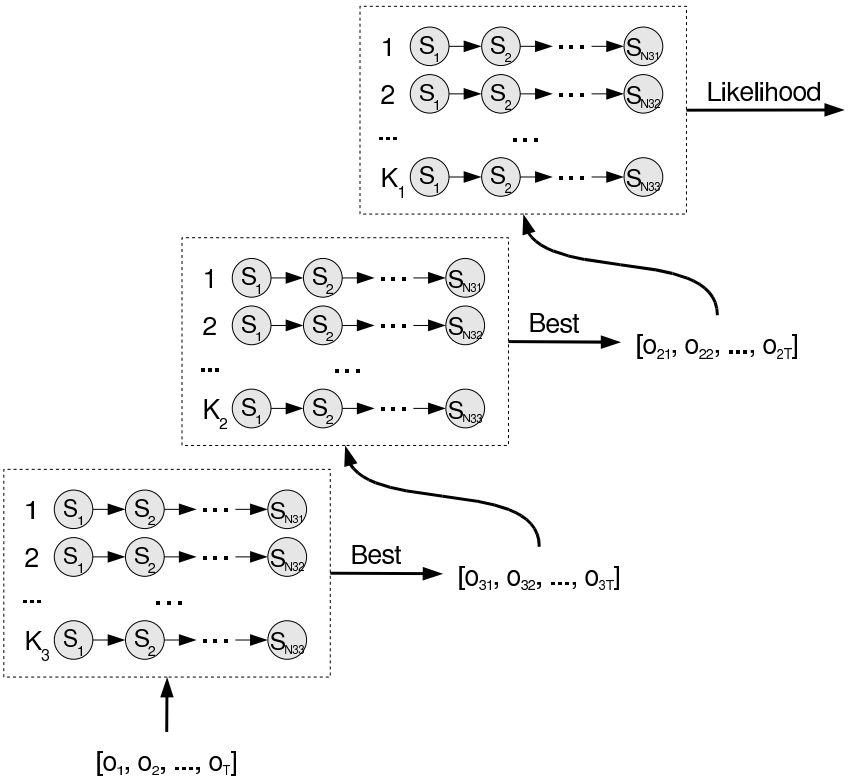


Fig. 1. A layered hidden Markov model.

expected from using the LHMM over a large single layer HMM is that the LHMM is less likely to suffer from over-fitting since the individual sub-components are trained independently on smaller amounts of data. A consequence of this is that a significantly smaller amount of training data is required for the LHMM to achieve a performance comparable of the HMM. Another advantage is that the layers at the bottom of the LHMM, which are more sensitive to changes in the environment such as the type of sensors, sampling rate etc, can be retrained separately without altering the higher layers of the LHMM.

Here, a LHMM with two levels is considered. At level 1 a single HMM is used to model the task, where each state in the HMM corresponds to a sub-task. At level 2 there is a HMM for each of the $K_2$ possible gestemes that may occur during execution of the task. The observation sequence for the level 2 HMMs is generated from the quantized motion direction of the trajectory recorded during task operation. The index of the HMM with highest likelihood among the $K_2$ HMMs at level 2 is then taken to be the the observation symbol for the level 1 HMM. The level 1 HMM is then used to compute the probability of a certain state as a function of time given the observation sequence produced by the HMMs at level 2. Since each state in the level 1 HMM corresponds to a mental stage of the teleoperation task this information can be used to understand the operator's intention. The proposed structure is outlined in Fig. 2. Here, the winning HMM at level 2, i.e. the one with the highest likelihood, is chosen and an observation symbol corresponding to this gesteme is

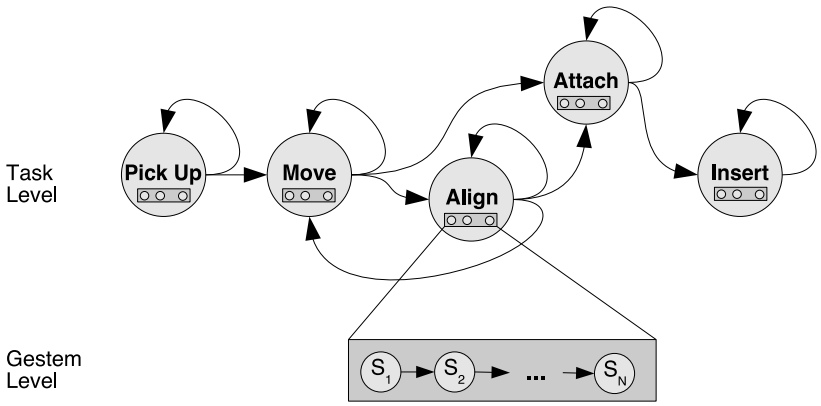


Fig. 2. A two level layered hidden Markov model, modeling gestemes at level 2 and a task at level 1.

generated for the level 1 HMM. The alternative would be to use the complete probability distribution and have the HMMs at level 2 act as a probability estimator for the level 1 HMM. However, according to [12] using the complete distribution does not give any apparent advantage over the simpler winner takes all model.

### *3.2 The gesteme HMM*

The goal of the gesteme HMMs is to distinguish between different motion primitives. For example, there can be gesteme HMMs to recognize motion along lines with different direction or circles with different radii and orientation in space. In our case, the gestemes can be any arbitrary motion in 2D or 3D. The observations for the gesteme HMMs are extracted from motion data. The trajectory is recorded, normalized and differentiated in order to compute the motion directions which are then mapped to corresponding observation symbols as described later in this section.

For the gesteme HMMs we evaluate the following types of models:
**One-dimensional HMM (OD)**: Here, the observation symbols are taken from a set $O = \{O_1, O_2, ..., O_K\}$ of $K$ discrete symbols. The **B** matrix is used to store the probability of observing the $j$th symbol in state $i$, $\mathbf{B_{i,j}} = P(O_j|\text{state } i)$. The symbols are generated by k-means clustering of all the training directions. The number of cluster centers is 25 in all experiments, if not stated otherwise. This number was chosen by an offline examination of the data. One observation is that using too few clusters makes it hard to distinguish between different motion directions while using too many makes the generalization difficult.

**Multi dimensional HMM (MD)**: The MD HMM assumes independence between the different dimensions of the input data. Thus, there is a **B** matrix for each dimension of the input data. This means that for a $D$ dimensional HMM the observation symbols are also $D$ dimensional where each dimension $d$ contains values from a

finite enumerated set. Each dimension is split into 10 equally sized bins and the input directions are projected into these bins generating the observation symbols.

**Multi dimensional HMM with FT (MD-FT)**: MD-FT is similar to the MD except that instead of mapping the raw motion directions to symbols, each dimension of the raw input directions are pre-processed by applying the Fourier transform to small overlapping windows, similar to that reported in [7]. In this work a Hamming window [23] of size 6 was used with 50% overlap.

### *3.2.1 Restarting the Gesteme Classifiers*

In the online approach presented in [3] all models are active in parallel. There are also two special states added to each HMM, the initial starter state and the resetting state. When the system is initialized the probability of starting in state 1 is 1 for all models whereas the probability of starting in any other state is 0.

The probability of a given model is then computed by the standard recursion formula [13], with the exception that $\alpha_t(1)$ is computed according to (1), where $\alpha_{t,m}(i)$ is the probability of being in state $i$ at time $t$ given the $m$:th HMM. This means that the probability that the model is in the initial state (i.e is reset) is the average of the probabilities that the $M$ models are in their resetting state.

$$\alpha_t(1) = \frac{\sum_{m=1}^{M} \alpha_{t,m}(N)}{M} \tag{1}$$

One potential problem with using the online approach presented in [3] is if one model $\lambda_1$ is very different from the other models it may happen that the observed symbols only occur in that model. If that is the case, the probability of any other model will be zero. If there is an observed symbol that should never occur in any state of $\lambda_1$ and has zero probability of being seen in state 1 in any other model, the probability of all models will become zero forever. The normal approach to deal with this issue is to assigning a small (non-zero) value of observing any symbol in any state, even if no such symbol was visible during training.

We have developed an alternative approach. By monitoring the most probable HMM it is possible to detect a sharp drop in probability when that model becomes invalid. Thus by low-pass filtering the derivative of the likelihood of the most probable model and applying a simple cumulative sum (CUSUM) RLS [24] test it is possible to get a good estimate of the change-time. The test statistic $g_t$ of the CUSUM test for the input $y_t$ is updated according to (2). Note that the negative sign of $y_t$ stems from the fact that we expect a negative change.

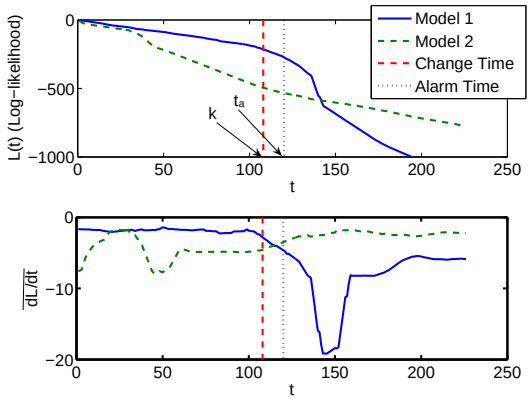


Fig. 3. Change time detection based on CUSUM RLS test on averaged model likelihood.

$$g_{t+1} = g_t - y_t - \nu - \hat{y}_t \tag{2}$$

$$g_t = 0, \text{ and } k^* = t \text{ if } g_t < 0 \tag{3}$$

$$g_t = 0,\ t_a = t \text{ and alarm if } g_t > h$$

$$\hat{y}_{t+1} = \hat{y}_t \cdot \frac{T-1}{T} + \frac{y_t}{T},\ T = t - t_0$$

The two parameters of the CUSUM test, the drift $\nu$ and the threshold $h > 0$ can be chosen to balance the robustness and the delay for detection of the test. The time at which the test triggers ($g_t > h$) is called the alarm time, $t_a$, and is the time when the algorithm recognizes a change in the signal $y_t$. The change time $k$ is then estimated as $k^*$, given by (3), and consequently the delay for detection is $\Delta_d = t_a - k^*$ (see Fig. 3).

When the alarm is triggered ($g_t > h$), the HMM estimates are restarted at time $k^*$, which marks the point in time where the currently selected model became invalid. Thus the HMM algorithm must maintain a memory of at least $\Delta_d$ old samples of the input.

### *3.3 The Task HMM*

The task HMM, or the level 1 HMM in the LHMM structure, encodes the task sequencing. Both levels of the LHMM work on the same time granularity and for each observation generated from the motion data the likelihood of the gesteme (level 2) HMMs are computed. The gesteme HMMs are enumerated and the index of the most likely gesteme HMM is used as an observation for the task level (level 1) HMM. Each of the states in the task level HMM corresponds to a sub-task in the operator's mental model and the most likely state can be computed in order to, for example, aid the operator with the execution of that sub-task. It should be noted that there need not be a one-to-one mapping between a state in the task level HMM and a gesteme HMM. Rather a specific gesteme can correspond to different states

depending on the previous state (the Markov assumption). Furthermore there may be several gestemes that can appear in a single state.

As mentioned previously, the states of the task level HMM are supposed to correspond to the mental states of the operator. As a consequence, it is not possible to use the Baum-Welch algorithm to train the task level HMM, because it will optimize the HMM parameters $\lambda$ in order to maximize $P(Q|\lambda)$ for the state sequence $Q$. The approach taken in this work is to have the operator manually segment the trajectory into sub-tasks corresponding to the mental model of the operator. The gesteme HMMs are trained using the Baum-Welch algorithm. Using the gesteme HMMs to classify the training data a new observation sequence $\mathbf{o}'$ is obtained. From the observation sequence $\mathbf{o}'$ the $\mathbf{B}$ can be computed by counting the occurrences of each symbol in every state and then normalizing the rows of $\mathbf{B}$. The task level HMM can now be trained by a modified version of the Baum-Welch algorithm where the $\mathbf{B}$ matrix is kept constant.

## 4 Experimental Setup and Data Generation

To better analyze and reproduce the results, we first carry out experiments on synthetic data. The goal is to first evaluate the three gesteme models: one-dimensional, multi-dimensional and multi-dimensional HMM with Fourier transform, with respect to the number of gestemes, the influence of the number of training samples, the effect of noise and the effect of the number of observation symbols. We also perform similar experiments with a robotic manipulator to verify the simulated results. There, the data is taken from a trajectory-tracking task where the end-effector of the manipulator is force controlled by a human operator.

A synthetic reference task consists of a sequence of motion primitives randomly generated from two groups of motion primitives. The first group contains straight lines of varying directions and lengths and the second group is made up of circle segments with varying starting and ending angles as well as orientations and radii. Fig. 4 shows example trajectories. These trajectory types may seem simple, but they were chosen because we believe that there exists several relevant tasks in areas such as medical surgery or automotive assembly that can be decomposed into a sequence of linear and circular motions.

The simulated trajectories are created in the following way. Given a reference trajectory $T_r$ a target point $\mathbf{p}$ is selected on $T_r$ so that the distance to $\mathbf{p}$ from the current position $\mathbf{q}$ is larger than some threshold $\xi$. A direction of motion $\mathbf{d}$ is then computed as the average between the direction towards $\mathbf{p}$ from $\mathbf{q}$ and the current direction of motion. A random error $\mathbf{e}_d$ is then added to $\mathbf{d}$ where each element of $\mathbf{e}_d$ is generated independently according to

$$\mathbf{e}_d(i) = \kappa \cdot \Gamma \tag{4}$$

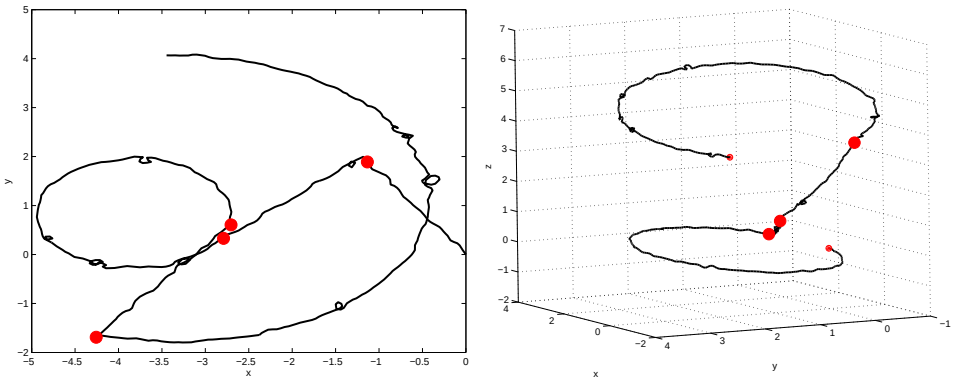

Fig. 4. Example trajectories in 2D (left) and 3D (right). The red dots marks the change from one primitive to the next.

where $\Gamma$ is generated from a normal distribution $(\mu = 1, \sigma = 1)$ and $\kappa$ determines the noise level. The value of $\kappa$ was set to 0.2 for all experiments if not otherwise stated.

Finally, the current position $\mathbf{q}$ is updated by taking a step of size $\delta \cdot (1 + 2\kappa \cdot \Gamma)$ in the direction of $\mathbf{d}$ where $\delta$ determines the step-size, which was set to 0.05 in all experiments. The value of $\kappa$ was set to 0.15 for all experiments if not otherwise stated. Here, three classes of reference trajectories are used. They are referred to as *line*, *circle* and *mixed* trajectories in 2D respectively 3D. The line trajectories are made up of a sequence of linear segments, the circle trajectories are comprised of circle segments and the mixed trajectory type consists of a mixture of linear and circular segments.

## 5 Experimental Evaluation

For the LHMM to be successful there must be a robust underlying gesteme classifier. Furthermore the LHMM and gesteme classifiers must be able to produce good results online with only partial observation sequences. It is also necessary to detect positions at which one gesteme ends and another begins in order to restart the gesteme classifiers at the correct instance in time - something that is required by the recursion equations if the HMM is to yield good performance. The experimental evaluation in this work consists of evaluating the HMM gesteme classifier for the three HMM types described in Section 4 with respect to the number of gestemes, the influence of the number of training samples, the effect of noise on classification performance and the online behavior. The experimental evaluation also considers the LHMM itself as well as the proposed method for restarting the gesteme classifiers.

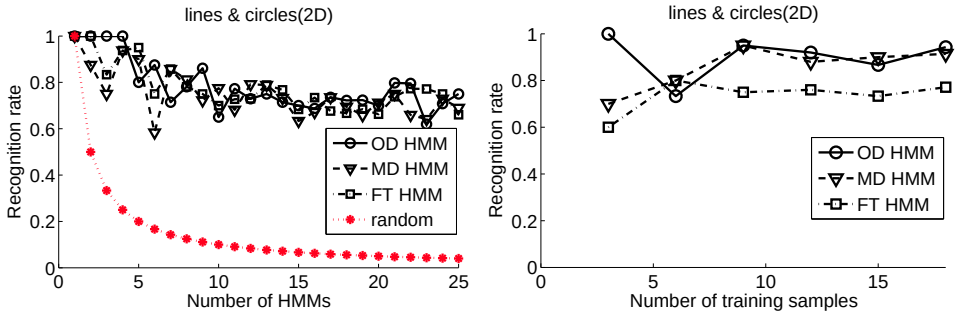


Fig. 5. **Left**: Typical deterioration of classifier performance as function of the number of HMMs. **Right**: The effect of the number of training samples on the HMM classification.

### *5.1 The Gesteme Classifier*

The first experiment evaluated the off-line performance of the HMM gesteme classifiers with respect to the number of different gestemes. If the gestemes are not generated at random but chosen from some set of gestemes that are constructed to be easy to distinguish between (such as the letters of the alphabet) the performance could be expected to be better than that reported here. As it can be seen in Fig. 5 (left), the recognition performance drops almost linearly from 100% to about 70% for 25 gestemes. The type of HMM or gesteme type (circles, lines or mixture) appears to have no statistical significance on the recognition performance. However, for three dimensional data the classification performance is a bit better but that can be explained with the fact that the individual gestemes are less likely to be similar.

It is well known that HMMs can be successfully trained with only a small amount of training data. Especially if there are few outliers such as in our training data where the motion is perfect except for the introduced white noise. It can be seen in Fig. 5 (right) that the recognition rate is quite high even for only two training runs. This is a good feature of the HMM gesteme classifier since in many settings extensive training is not possible. When the type of noise changes and outliers are introduced the necessary number of training sequences will increase some in order to be able to capture the larger variations that occurs. However, preliminary results indicate that in practice the necessary number of training sequences is actually quite low as long as the training sequences are representative for what will occur during execution.

The HMM is able to handle a large amount of noise as long as the noise is consistent during training and classification. To evaluate what amount of noise the gesteme classifiers can handle, we tested the classification performance with several synthetic runs generated by varying the value of $\kappa$ from 0.05 to 0.55. For the proposed methods to work in the intended setting it is required to obtain good results with only a limited amount of training samples. Therefore only five training samples where used for the experiments in this section, if not otherwise stated. Fur-

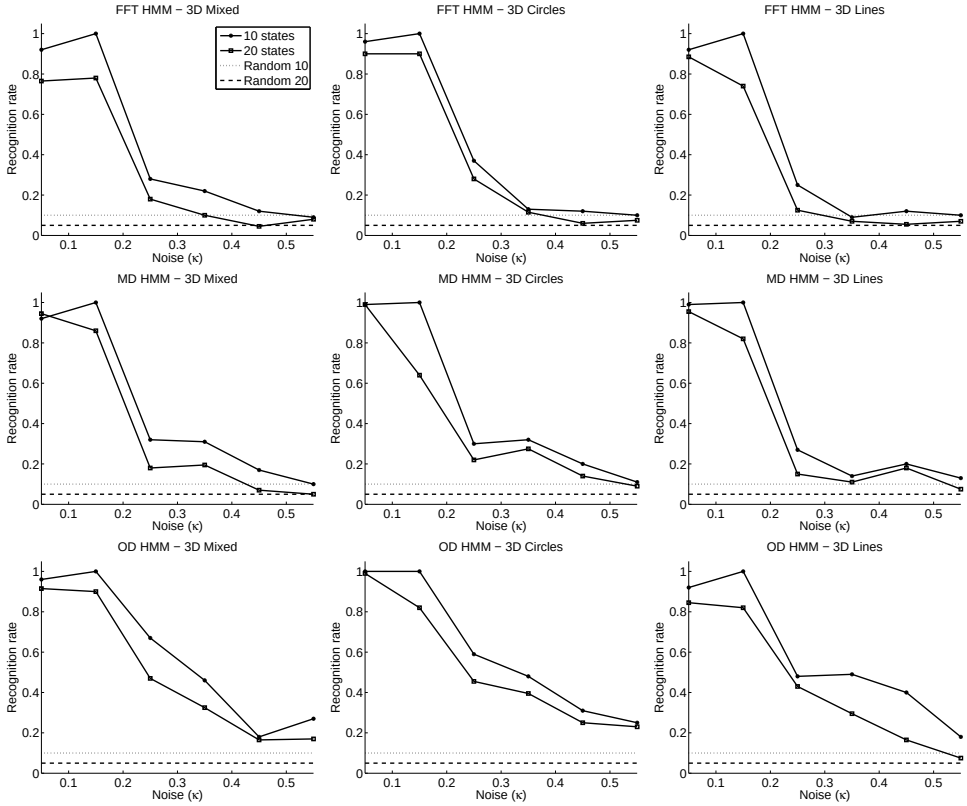


Fig. 6. Classification performance as a function of noise.

thermore, the results presented in this section are the average of 10 independent trials, if not explicitly stated.

Fig. 6 and 7 shows the classification performance as a function of the noise, $\kappa$ for 2D and 3D data. We can conclude that a reasonable value for the noise parameter $\kappa$ is less then $0.2-0.25$. For the remainder of the experimental results on synthetic data the value of $\kappa$ is therefore set to 0.15 unless explicitly specified. Note that a value of $\kappa \in [0.3, 0.5]$ is almost as bad as guessing. By examining the individual runs, it can be seen that the noise sensitivity is highly affected by the similarity of the gestemes. If the gestemes are similar, the performance decreases almost linearly with increased noise. If the gestemes contains few common symbols, the classification performance remains relatively unaffected until the noise starts to dominate (i.e is large compared to the nominal motion). One interesting result is that the OD HMM appears to have better performance with respect to noise sensitivity. We believe that the reason for this is the low dimensionality and that the k-means clustering of the pre-processing step helps with generalization since the cluster centers are affected by the actual training data instead of using pre-defined bins.

The next experiment evaluates the effect of the number of gestemes on classification when considering noise. For every gesteme there is a corresponding HMM. As it can be seen in Fig. 8 and 9, the performance drops almost linearly from 100% to about 60% for 25 gestemes for the medium noise case where $\kappa = 1.5$. It is again in-

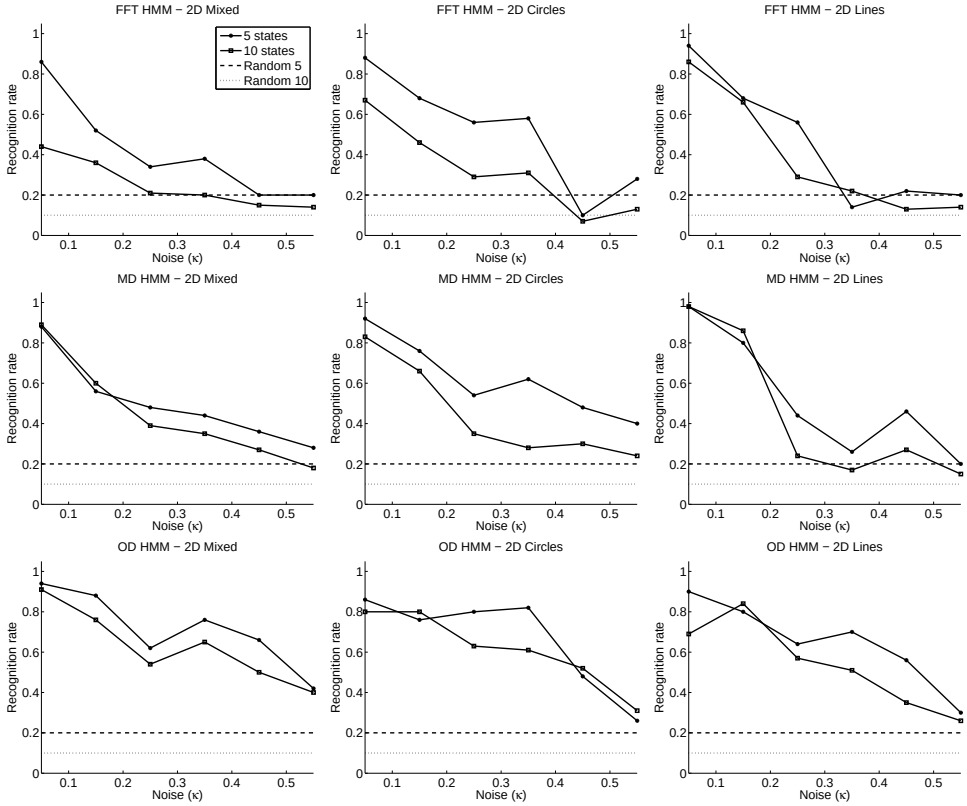


Fig. 7. Classification performance as a function of noise.

teresting to note that the OD HMM appears to have better performance w.r.t noise. The classification performance for the 3D data is a bit better but that can be explained with the fact that the individual gestemes are less likely to be similar.

The number of observation symbols is not crucial but have to be set reasonably. If too few symbols are used the HMM can not distinguish between different directions leading to poor classification. At the same time, using too many symbols will prevent the HMM from generalizing, leading to poor classification because none of the models will correspond well with the training sequences. Fig. 10 and 11 show the classification performance as a function of the number of observation symbols. Remember that the observation symbols are defined differently between the OD and MD HMMs and values are thus not comparable. For the OD HMM the observation symbols correspond to the cluster centers obtain from the k-means clustering of the nominal motion directions of the training data, whereas for the MD HMMs the observation symbols are taken from $M \cdot D$ predefined bins of size $1/M$ giving a total of $M^D$ different possible observations, where $M$ is the number of discrete observation symbols and $D$ is the dimensionality of the MD HMM.

The above experiments have been conducted off-line considering that the whole gesteme was available in the recognition stage. In order to be used in the intended setting, the LHMM and gesteme classifiers must be made to work online with only partially observed gestemes. The next experiment evaluated the gesteme classifiers

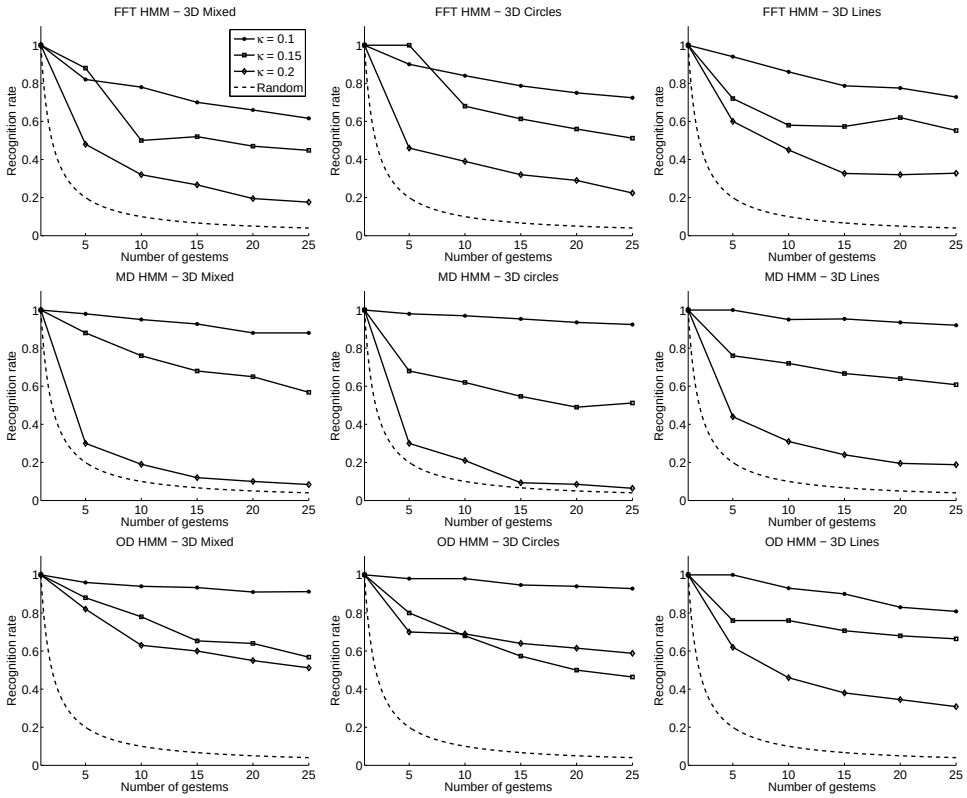


Fig. 8. Classification performance as a function of the number of gestemes.

online performance. Figure 12 shows that the gesteme classifiers can produce good results after observing only a small fraction (10%-20%) of the gesteme. The results here will depend strongly on the similarity between the first part of the gestemes and the success will thus vary depending on the type of task. Another important aspect for online classification is the exact time at which the HMM recursion starts. In this case the times where known due to the fact that the test data was synthetically generated. If the change time for switching between gestemes are off there is a large risk to observe very unlikely observation symbols and thus the correct HMM can be severely penalized in beginning of the classification and in worst case never recover.

### *5.2 The LHMM*

Examples of 2D trajectories that contain four gestemes, $G = \{l_1, l_2, l_3, c_1\}$ are shown in Fig. 13. The "mental model" of the first example task is that the gestemes should be performed in a SLR fashion with the $c_1$ gesteme appearing twice so the task should go through the five different states $S_1, ..., S_5$ and thus execute the gestemes in the following order: $l_1, c_1, l_2, c_1, l_3$. The gesteme is exactly the same in $S_2$ and $S_4$ so one cannot differentiate between these states by simply monitoring the output from the four gesteme classifiers.

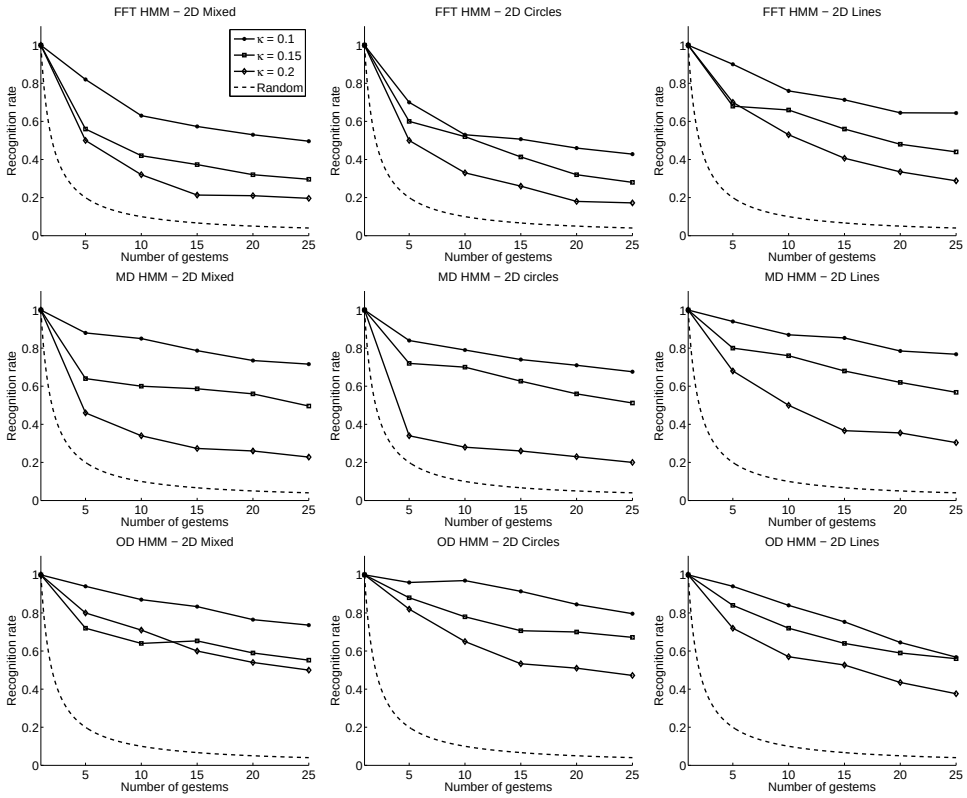


Fig. 9. Classification performance as a function of the number of gestemes.

A task level HMM is now trained on the output of the gesteme classifiers. That is, the trajectory is classified by the gesteme classifiers (online) and the sequence of winning gestemes are used as input to the task-level HMM which is trained in order to extract the task-model. Figure 13 (right) shows result of classification obtained by the gesteme classifiers.

It can be seen from Fig. 13 that even tough the gesteme classifiers are sometimes confusing gesteme $l_1$ and $c_1$ the task-level HMM is still capable of determining the correct state. This is because the miss-classifications of the gesteme classifiers are consistent with training data and thus the task-level HMM expects some miss-classifications. Furthermore the discriminant power of the LHMM is much better than that of the HMM, i.e. the difference between the most probable and the second most probable state is in general much larger for the LHMM.

### *5.3 Experiments with a Robot Manipulator*

In order to verify the validity of the proposed approach and to show that the quantitative simulated results are relevant, we have performed a number of qualitative experiments with a robot manipulator. The manipulator used is a made from a number of PowerCube elements and passive links and it is mounted on a mobile base

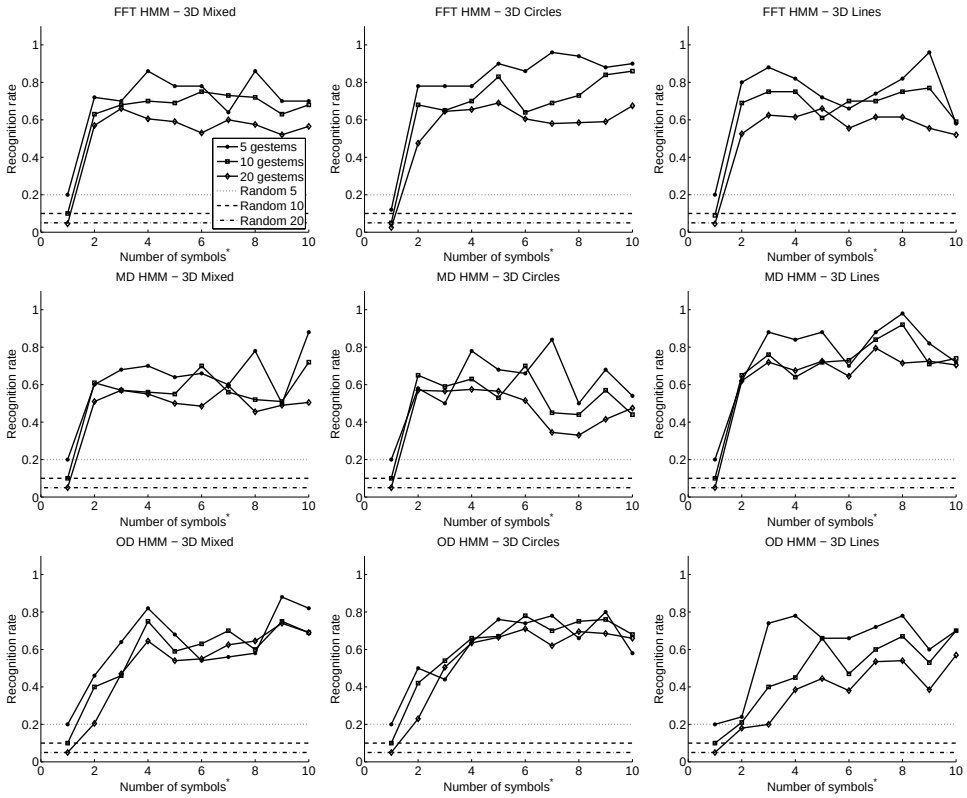


Fig. 10. Classification as a function of the number of symbols. Note that the number of symbols have different meaning for OD and MD HMMs.

as shown in Fig. 14.

The manipulator has a JR3 force-torque sensor mounted between the end-effector and the last link, providing 6 DOF force-torque measurements at the end-effector. The force-torque sensor provides decoupled data at 8 kHz per channel, which is low-pass filtered with the bandwidth 30 Hz (-3 dB) by a DSP. The (filtered) force-torque data is used to control the position and orientation (pose) of the end-effector by driving the end-effector with a velocity proportional to the forces and torques. Due to the kinematics of the manipulator, large motions (of the joints) are sometimes required to realize small changes in orientation of the end-effector. This can make control of the manipulator more difficult than for a PUMA-like robot (6 rotary DOF with the 3 DOF of the wrist intersecting a single point). In all experiments the mobile platform is stationary and the operator guides the manipulator by applying forces to the end-effector.

Two trajectory tracking tasks are used in the experiments. The first task consists of tracking a sequence of lines and circle segments on a planar 2D surface, very similar to the simulated trajectories used previously. The second task consists of tracking a trajectory on an object in 3D without touching the object. Two representative trajectories are shown in Fig. 15.

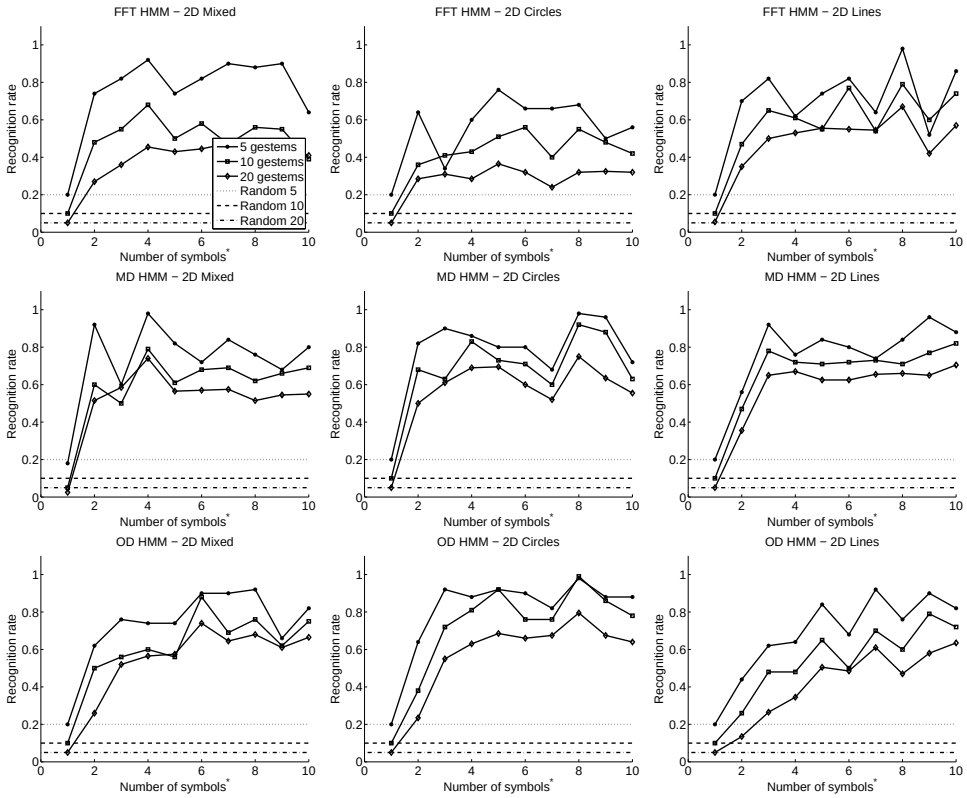


Fig. 11. Classification as a function of the number of symbols. Note that the number of symbols have different meaning for OD and MD HMMs.

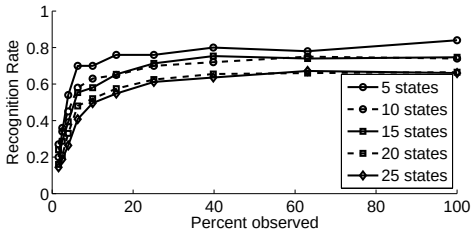


Fig. 12. Online recognition performance of MD HMM with 3D data consisting of lines and circle segments.

The trajectories are normalized so that samples are 1 cm apart (this is a reduction of data of about 90%). From the normalized data the sequence of motion directions is computed and k-means clustering is used to identify 10, for 2D data, or 25, for 3D data, cluster centers used as the symbols in a one-dimensional HMM. The sequence of motion directions is then transformed to a sequence of observation symbols. A total of five trajectories were recorded and three of them were used for training and the results presented here were evaluated on two trajectories. The reason for using only 3 trajectories is that one important aspect of the proposed system is to provided good results even with little training data, which should be possible given the previously presented results.

Figure 16 shows the results of the online classification of the gestemes for the

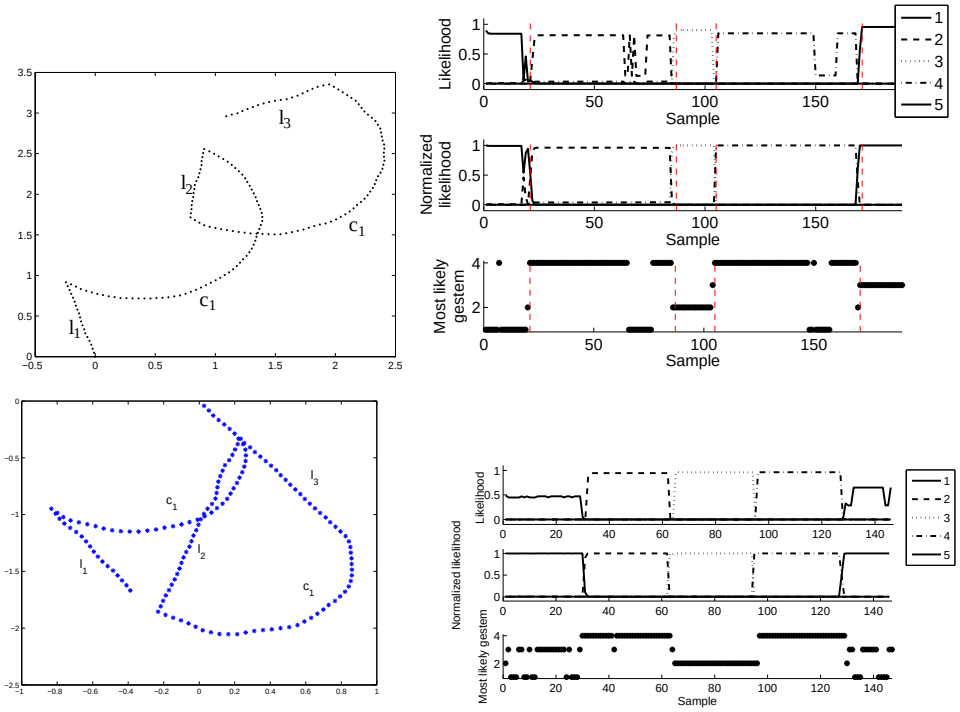


Fig. 13. Left: Example trajectories of a task with 5 states and 4 gestemes, and Right: LHMM Classification - the top plot shows the likelihood of each state and the plot in the middle shows the normalized likelihood. The bottom plot shows the (online) classification of the motion by the gesteme classifiers and is the input to the task-level HMM.

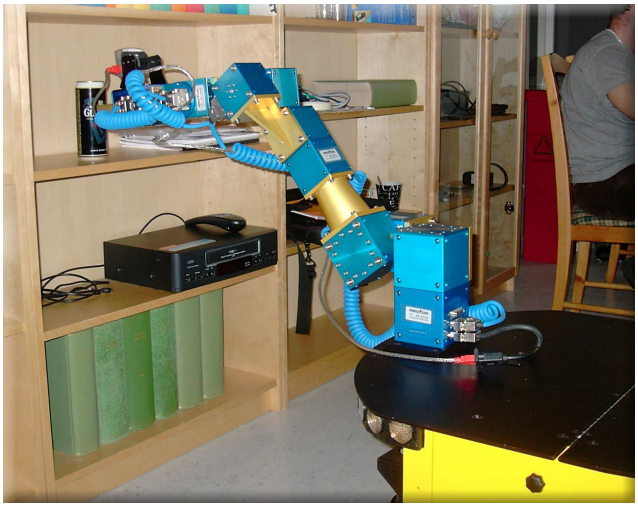

Fig. 14. The mobile manipulator used for the experimental validation.

3D trajectory in Fig. 15. The CUSUM test described previously was used to reset the classification. Figure 16 shows the low-passed filtered derivative of the HMM probability used for the test as-well as the test statistic and the estimated and true change times. As it can be seen the classification is good even tough there were only three training trajectories available. One of the reasons for this is that it is the same person that performs the training and testing sequences. For operator independent

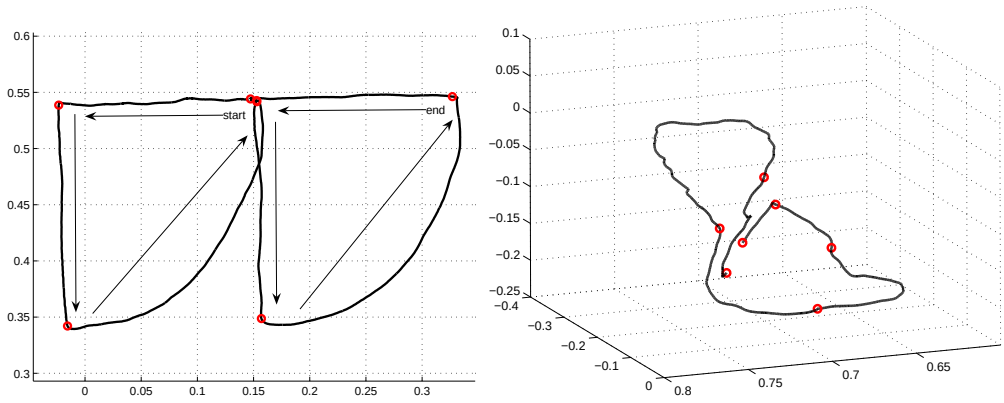


Fig. 15. Representative trajectories for the two trajectory tracking tasks.

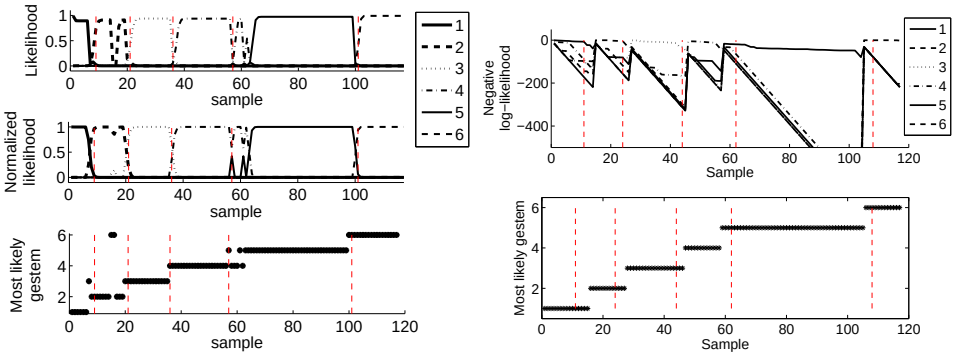


Fig. 16. **Left:** State probability given a mental model of a task. **Right:** Online classification of robot trajectories. The solid green lines are the manually estimated change times, the dashed red lines are the change times estimated by the CUSUM test and the dotted red lines are the time the test triggered.

training the number of required training samples is expected to be higher.

We tested the LHMM on the task shown in Fig. 15 (left) with the sequence of gestemes $\{l_1, l_2, c_1, l_2, c_1, l_1\}$. Even tough the high accuracy of the underlying gesteme classifiers are very high the use of the LHMM is still motivated by two facts. First, it can encode the sequence of the gestemes and thus tell them apart even tough the same gesteme appears more than ones. Second the discriminate power is greater so we can have a more confident classification. We also tested the same LHMM on the sequence $\{l_1, l_2, c_1, l_1, c_1, l_2\}$ which is not seen during training. We see that the LHMM can still recognize the correct state sequence. However, there is a significant delay before the evidences (observations) are strong enough warrant a state change. This can be seen around sample 70 and 90 of Fig. 17 (right).

For the 3D trajectory tracking task we have manually segmented the data into 6 different mental stages. In order to fit the LHMM to this mental model we compute the $B$ matrix as

$$b_i(j) = \frac{S_{i,j}}{\sum_{k=1}^{M} S_{i,k}} \tag{5}$$

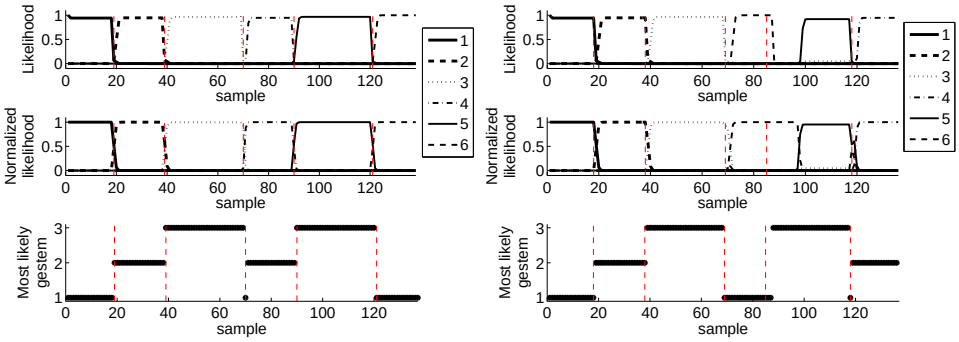


Fig. 17. Classification of the LHMM for the two dimensional trajectory tracking task. The top plot shows the likelihood of each state and the plot in the middle shows the normalized likelihood. The bottom plot shows the (online) classification of the motion by the gesteme classifiers and is the input to the task-level HMM. **Left:** the trained sequence $\{l_1, l_2, c_1, l_2, c_1, l_1\}$. **Right:** A sequence not seen during training, $\{l_1, l_2, c_1, l_1, c_1, l_2\}$.

where $M$ is the number of possible observation symbols, $S_{i,j}$ is the number of times the $j$th observation symbols is observed in state $i$ during training. In order to make the model more robust with limited training data, we do not allow a zero probability in the $B$ matrix. We set all zero elements of the $B$ matrix to a small non-zero value. The reason for computing the $B$ matrix in this way is that the states of the mental model may not correspond with the states obtained from the Baum-Welch algorithm which optimizes the model parameters to (locally) maximize $P(O|\lambda)$. It can be seen from Fig. 16 that even tough the gesteme classifiers are sometimes detecting the wrong gesteme the LHMM can still clearly recognize the correct state. It can also be seen from the unnormalized likelihood that the total probability drops rapidly when unexpected gestemes are identified. This can be used to assign a measure of the certainty of the system and can be useful determining how much trust to put into the classification during, for example, a fixturing of the motion, as was done in [6].

## 6 Summary and Conclusions

Task segmentation and modeling is one of the core research areas in the field of teleoperation, human-machine collaborative and programming-by-demonstration systems. In this paper, we have investigated which parameters determine the success of the HMM approach to motion intention recognition as well as presented a layered hidden Markov model approach for complex task modeling. The proposed methodology uses three different HMM models at the gesteme level: one-dimensional HMM, multi-dimensional HMM and multi-dimensional HMM with Fourier transform. These three models were evaluated with respect to the number of gestemes, the influence of the number of training samples, the effect of noise and the effect of the number of observation symbols in both 2D and 3D tasks.

Experimental evaluation shows that LHMMs have a good potential for modeling and real-time recognition of teleoperative, HMCS and PbD tasks. The evaluation has also shown that both one and multi dimensional HMMs are suitable for modeling gestemes and they are even able to handle gestemes that are quite similar in nature as long as the SNR is low. The HMMs are able to suppress relatively large amounts of noise as long as the noise is white. However, initial results indicate that the HMMs are more sensitive to other types of noise. One observation was that the OD HMM shows better performance with respect to noise sensitivity and we conclude that this is due to the low dimensionality and k-means clustering of the pre-processing step. The FT-based recognition performs somewhat worse than the analysis of spatial data. we believe that this is because the sampling rate and the trajectory are not harmonic.

It is clear from the experimental evaluation that the LHMM has a strong potential to model complex tasks since it is able to perform well even with miss-classifications in the underlying layers. Thus as long as the gesteme classifiers produce consistent misclassification during training and testing the layered structure of the LHMM is able to handle this. The LHMM also has a much greater discriminating power than the standard HMM approach.

Furthermore we have also pointed out three main issues that must be addressed in order to successfully build a layered HMM for online motion intention recognition. The first issue is that it is necessary to have a robust gesteme classifier. Secondly the gesteme classifier must work online, with only partial observations of the gesteme. Finally, it is necessary to detect when one gesteme ends in order to restart the gesteme classifiers. In the future, we will implement a system similar to that in [6] using a LHMM to solve a larger set of more complicated tasks.

## References


[1] C. S. Hundtofte, G. D. Hager, A. M. Okamura, Building a Task Language for Segmentation and Recognition of User Input to Cooperative Manipulation Systems, in: Proc. of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002, pp. 225–230.

[2] M. A. Peshkin, J. E. Colgate, W. Wannasuphoprasit, C. Moore, R. B. Gillespie, P. Akella, Cobot Architecture, IEEE Trans. on Robotics and Automation 17 (2001) 377–390.

[3] M. Li, A. Okamura, Recognition of Operator Motions for Real-Time Assistance Using Virtual Fixtures, in: Proc. of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003, pp. 125– 131.

[4] D. Kragić, P. Marayong, M. Li, A. M. Okamura, G. D. Hager, Human-Machine Collaborative Systems for Microsurgical Applications, Int. Journal of Robotics Research 24 (2005) 731 – 741.

[5] A. Castellani, D. Botturi, M. Bicego, P. Fiorini, Hybrid HMM/SVM Model for the Analysis and Segmentation of Teleoperation Tasks, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2004, pp. 2918–2923.

[6] D. Aarno, S. Ekvall, D. Kragić, Adaptive Virtual Fixtures for Machine Assisted Teleoperation Tasks, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2005, pp. 897–903.

[7] W. Yu, R. Alqasemi, R. Dubey, N.Pernalete, Telemanipulation Assistance Based on Motion Intention Recognition, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2005, pp. 1121– 1126.

[8] R. Zöllner, O. Rogalla, R. Dillmann, M. Zöllner, Understanding Users Intention: Programming Fine Manipulation Tasks by Demonstration, in: Proc. of the Int. Conf. on Intelligent Robots and Systems, 2002, pp. 1114– 1119.

[9] M. Kaiser, R. Dillmann, Building Elementary Robot Skills from Human Demonstration, in: Proc. of the Int. Conf. on Robotics and Automation, 1996, pp. 2700–2705.

[10] R. H. Taylor, D. Stoianovici, Medical Robotics in Computer-Integrated Surgery, IEEE Trans. on Robotics and Automation 19 (2003) 765– 781.

[11] R.-H. Liang, M. Ouhyoung, A Real-Time Continuous Gesture Recognition System for Sign Language, in: Proc. of the Int. Conf. on Automatic Face and Gesture Recognition, 1998, pp. 558–567.

[12] N. Oliver, A. Garg, E. Horvitz, Layered Representations for Learning and Inferring Office Activity from Multiple Sensory Channels, Computer Vision and Image Understanding 96 (2004) 163–180.

[13] L. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proc. of the IEEE 77 (1989) 257–286.

[14] A. Brakensiek, A. Kosmala, D. Willett, W. Wang, G. Rigoll, Performance Evaluation of a New Hybrid Modeling Technique for Handwriting Recognition Using Identical On-Line and Off-Line Data, in: Int. Conf. on Document Analysis and Recognition, 1999.

[15] J. T. Nolin, P. M. Stemniski, A. M. Okamura, Activation Cues and Force Scaling Methods for Virtual Fixtures, in: Proc. of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003, pp. 404– 409.

[16] P. Marayong, A. Bettini, A. Okamura, Effect of Virtual Fixture Compliance on Human-Machine Cooperative Manipulation, in: Proc. of the IEEE/RSJ Int. Conf. on Inteligent Robots and Systems, 2002, pp. 1089– 1095.

[17] S. Fine, Y. Singer, N. Tishby, The Hierarchical Hidden Markov Model: Analysis and Applications, Mach. Learn. 32 (1) (1998) 41–62.

[18] D. Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, G. Lathoud, Modeling Individual and Group Actions in Meetings: a Two-Layer HMM Framework, in: 2nd IEEE Workshop on Event Mining: Detection and Recognition of Events in Video, In Association with CVPR, 2004, iDIAP-RR 04-09.

[19] L. Xie, S. Chang, A. Divakaran, H. Sun, Unsupervised Discovery of Multilevel Statistical Video Structures Using Hierarchical Hidden Markov Models, in: Proc. of the IEEE Int. Conf. on Multimedia and Expo (ICME), 2003, pp. III – 29 – 32.

[20] A. Dielmann, S. Renals, Dynamic Bayesian Networks for Meeting Structuring, in: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2004.

[21] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
URL http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=18626

[22] B. Hannaford, P. Lee, Multi-Dimensional Hidden Markov Model of Telemanipulation Tasks With Varying Outcomes, in: Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics, 1990, pp. 127–133.

[23] F. J. Harris, On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform 66 (1978) 51–83.

[24] F. Gustafsson, Adaptive Filtering and Change Detection, John Wiley & Sons, Ltd, 2000.