2008 8th IEEE-RAS International Conference on Humanoid Robots
December 1 ~ 3, 2008 / Daejeon, Korea

Motion Imitation and Recognition using Parametric Hidden Markov Models

Dennis Herzog [1], Aleš Ude [2,3], Volker Krüger [1]

[1]*Aalborg University Copenhagen, Denmark* deh@cvmi.aau.dk, vok@cvmi.aau.dk

[2]*Jožef Stefan Institute, Slovenia* ales.ude@ijs.si

[3]*ATR Computational Neuroscience Laboratories, Japan* aude@atr.jp

***Abstract*—The recognition and synthesis of parametric movements play an important role in human-robot interaction. To understand the whole purpose of an arm movement of a human agent, both its recognition (e. g., pointing or reaching) as well as its parameterization (i. e., where the agent is pointing at) are important. Only together they convey the whole meaning of an action. Similarly, to imitate a movement, the robot needs to select the proper action and parameterize it, e. g., by the relative position of the object that needs to be grasped.**

We propose to utilize parametric hidden Markov models (PHMMs), which extend the classical HMMs by introducing a joint parameterization of the observation densities, to simultaneously solve the problems of action recognition, parameterization of the observed actions, and action synthesis. The proposed approach was fully implemented on a humanoid robot HOAP-3. To evaluate the approach, we focused on reaching and pointing actions. Even though the movements are very similar in appearance, our approach is able to distinguish the two movement types and discover the parameterization, and is thus enabling both, action recognition and action synthesis. Through parameterization we ensure that the synthesized movements can be applied to different configurations of the external world and are thus suitable for actions that involve the manipulation of objects.

I. Introduction

To effectively interact with people, a humanoid robot needs to be able, firstly, to recognize human movements in order to understand the intentions of an agent communicating with the robot, and secondly, to imitate human actions in a human way. These are essential tools that are needed to enable the robot to autonomously operate in a human environment.

We are interested in an action representation that is (a) easily and efficiently trained by demonstrating a set of exemplar movements and (b) able to recognize and synthesize the demonstrated actions for arbitrary parameterizations. The action representation should allow the robot to recognize and perform the learned movements also for not previously demonstrated parameterizations, e. g., reaching and pointing at objects at arbitrary locations.

To achieve this goal we utilize parametric hidden Markov models (PHMMs), which were originally proposed in [1]. We encode PHMMs by observations states that consist of Cartesian 3-D positions of shoulder, sternum, elbow, wrist, thumb, index-finger and its knuckle. The advantage of such an representation is that the goal of the considered actions (reaching and pointing) is explicitly encoded in the final state of the PHMMs, which makes it easier to find parameterization for action interpolation. To assure a proper interpolation that preserves the shape of the action trajectory, a proper alignment of the different exemplar actions, i. e. the alignment of corresponding hidden Markov states, is essential. We solved this problem by constraining the state transitions.

As an example, we studied the teaching of a humanoid robot through pointing and reaching gestures. The goal was to put differently shaped objects into a children's toy box with differently shaped holes corresponding to different shapes. The robot should learn both symbolic knowledge (which object belongs to which hole) and continuous action knowledge (how to move the objects and release them into the appropriate hole). For training and testing, the objects could be placed at any location on the table. An online demo is available via web page [2].

In addition, we present a systematic evaluation of the recognition and synthesis of our action representation. In the evaluation, we focus on the actions used in the demonstration on a humanoid robot HOAP-3, i.e. reaching out for an object to grasp it and pointing actions. Both actions have very similar trajectories (starting and ending in the same base pose) and are thus difficult to distinguish. It is interesting to note that simple diagnostic features like arm velocity, or the distance from hand to chest would fail in this context.

In the following sections, we first give a short overview of the related work. In Sect. III we introduce our exemplar-based parametric HMM movement representation. The action synthesis is described in Sect. IV. In Sect. IV-C we present the results of the robot experiments. In Sect. V we discuss our systematic evaluation of the precision of our action representation for recognition and synthesis. Conclusions in Sect. VI complete our paper.

II. Related Work

The discovery of mirror neurons motivated robotics researchers to look for action representations that can be used for both recognizing other agents' actions and generating the observer's own movements [3]. Among the representations that can be used both for synthesis and recognition: dynamic movement primitives [4], [5], recurrent neural network with parametric biases (RNNPB) [6], and hidden Markov models (HMMs) [7], [8], [9].

Hidden Markov models became popular in the context of speech recognition [10], [11]. One major advantage of HMMs is their ability to compensate for uncertainties in time. Inamura et al. [7] showed that by defining the observation states as postures on the human trajectories, HMMs can be used to represent specific movement trajectories, which they called protosymbols. However, neither discrete nor continuous HMMs are able to generalize over a class of movements which vary according to a specific set of parameters (like for example reaching and pointing). One possibility to recognize an entire class of movements is to use a set of hidden Markov models (HMMs) in a mixture-of-experts approach, as first proposed in [12]. In order to deal with a large parameter space, one ends up with a lot of experts, and training becomes unsustainable.

As mentioned above, the extension of the classical HMMs into parametric HMMs was first proposed by Wilson and Bobick [1]. They developed an approach that is able to learn a parametric HMM based on a set of demonstrations, where training and recognition is performed by using the EM algorithm with parameterization parameters as latent variables. Note that this is different from [8], and [9], where HMMs are trained from multiple examples of an essentially same movement. However, Wilson and Bobick considered recognition only, more specifically, e. g., the recovery of the pointing direction based on wrist trajectories.

Contrary to Wilson and Bobick, we aim at recognition *and* synthesis of full arm movements for the control of humanoid robots within the same framework. In our work, "recognition" means to recognize the action itself as well as its parameterization. The synthesis of parametric actions implies the use of data of high dimension (stacked 3-D trajectories, one for each joint), and a high number of states for an accurate movement representation. It has already been shown both in robotics [13] and in computer graphics [14] that parametric blending of movements can result in physically feasible actions that can attain the goal of an action. These works, however, do not consider the problem of recognition.

Human motion capture [15] as such is not the topics of this paper. We note, however, that motion capture is an essential tool for the acquisition of training data for imitation. We experimented with magnetic systems, optical tracking devices, and general vision-based methods. General vision data was tested for recognition, but we used the data from a marker-based optical system as an initial input when working with the robot (see Fig. 1).

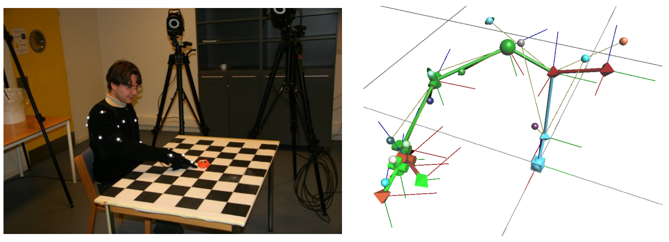


Fig. 1. *Left: Capturing Session* for our dataset. — *Right: Capture Model.* For motion capturing, the markers of the depicted model (tiny balls) are aligned to the captured markers (see left figure).

III. Movement Representation by PHMMs

In this section we first give a short introduction to HMMs and then discuss our extension to parametric HMMs.

A. *Preliminaries of HMMs*

A hidden Markov model is a finite state machine extended in a probabilistic manner. It is defined as a triple $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. Let q_t be the hidden states of the model at time t and $\boldsymbol{x}$ the observations associated with each hidden state. Then, $\boldsymbol{B}$ defines the output distributions $b_i(\boldsymbol{x}) = \mathrm{P}(\boldsymbol{x}|q_t = i)$ of the hidden states. The transition matrix $\boldsymbol{A} = (a_{ij})$ defines the transition probability between the hidden states $i, j = 1, \ldots, N$, and thus encodes the temporal behavior of the modeled sequences. The initial state distribution is defined by the vector $\boldsymbol{\pi}$.

Our approach is based on continuous left-right HMMs [16]. The output probability distribution of each state i is modeled by a single Gaussian distribution $b_i(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. State transitions are either self-transitions or transitions to the successor. All other transition probabilities are set to zero. Given a sequence of postures $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_t \ldots \boldsymbol{x}_T$ on an example trajectory, each Gaussian $\mathcal{N}_i(\boldsymbol{x}) := b_i(\boldsymbol{x})$ "covers" a section of the trajectory, where the state i increases over time. In the case of multiple trajectories, the Gaussians capture the variance of the training input. In addition, an HMM compensates for different progression rates of the training trajectories by varying the transition from one hidden state to the next. Obviously, the synthesis of movements is straightforward for this type of HMMs. For a comprehensive introduction to HMMs, we refer to [10], [11].

For recognition or classification, HMMs are generally used as follows: For each sequence class k, an HMM $\boldsymbol{\lambda}^k$ is trained by maximizing the likelihood function $\mathrm{P}(\mathcal{X}|\boldsymbol{\lambda}^k)$ with the Baum-Welch expectation maximization (EM) algorithm [11] over a given training data set $\mathcal{X}^k$. The classification of a specific output sequence $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_T$ is done by identifying the class k for which the likelihood $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^k)$ is maximal. The forward-backward algorithm [11] is used to efficiently calculate these likelihoods.

One obvious approach to handling classes of parameterized actions for the purpose of parameter recognition is a mixture-of-experts approach [12] by sampling the parameter space. However, this approach suffers from the great number of HMMs needed to be trained and stored for all possible trajectories within one class. Therefore, we introduce the parameterization of an action as an additional model parameter. This is also the basic idea of [1].

B. *Parametric HMM Framework*

The basic idea of our approach to handling classes of parameterized actions is to generate a new HMM for novel action parameters by a locally linear interpolation of exemplar HMMs that were previously trained on exemplar movements with known parameters. The generation of an HMM $\boldsymbol{\lambda}^{\phi}$ for a specific parameter $\boldsymbol{\phi}$ is carried out by component-wise linear

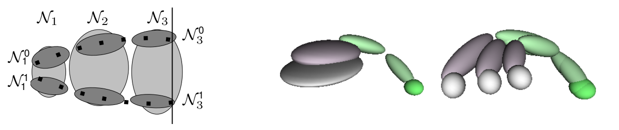


Fig. 2. *Left:* The upper three dark ellipsoids are depicting Gaussians $\mathcal{N}_1^0, \ldots, \mathcal{N}_3^0$ of states $i = 1, 2, 3$ of an local HMM $\boldsymbol{\lambda}^0$, whereas the lower three dark ellipsoids belong to a model $\boldsymbol{\lambda}^1$. The dots within the dark ellipsoids are sketching training sequences with parameterization $u = 0$, and $u = 1$ depending on the target at the vertical line. In addition, the Gaussians $\mathcal{N}_i$ of a global model $\boldsymbol{\lambda}$ are indicated in light gray. This global $\boldsymbol{\lambda}$ is a model for all training sequences with $u \in [0, 1]$. — *Right:* Similar to the left figure, some states of a global HMM (middle) and local HMMs (right) are shown. In contrast to the figure left, the HMMs are trained based on recorded 3D trajectories of a finger tip of a person pointing at three different positions at table-top (compare Fig. 1). The index finger starts always at the same point as modeled by the Gaussian which is sketched by the green ball, and approaches specific positions which are modeled by the light gray balls of the local HMMs. The global HMM used to setup these local HMMs has a disc like Gaussian, which models all approached positions at table-top.

interpolation of the nearby exemplar models. In the case of a single parameter u and two exemplar models $\boldsymbol{\lambda}^{u=0}$ and $\boldsymbol{\lambda}^{u=0}$, a new Gaussian $\mathcal{N}_i^u(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^u, \boldsymbol{\Sigma}_i^u)$ for a model $\boldsymbol{\lambda}^u$ with $u \in [0, 1]$ is generated by interpolation

$$\boldsymbol{\mu}_i^u = (1-u)\boldsymbol{\mu}_i^0 + u\boldsymbol{\mu}_i^1, \quad \boldsymbol{\Sigma}_i^u = (1-u)\boldsymbol{\Sigma}_i^0 + u\boldsymbol{\Sigma}_i^1. \tag{1}$$

This situation is described in Fig. 2. Such an approach, however, works *only if* two corresponding states of the two exemplar HMMs model the same semantical part of the trajectory (as in Fig. 2, left). Hence the state-wise alignment is vital and is described in Sec. III-B1. The expansion to the multi-variate case of parameterization $\boldsymbol{\phi}$ is then straightforward, e. g., by using bilinear ($\boldsymbol{\phi} = (u, v)$) or trilinear interpolation.

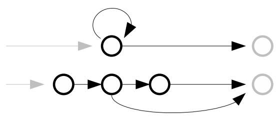


Fig. 3. *Time duration by State Replacement.* Each state is replaced by several (here, three) pseudo states which share the same output distribution.

1) Synchronization of HMMs: In this section we show how to assure that corresponding states of exemplar HMMs model the same semantical parts of movements. The required alignment of the states to the sequences is somehow similar to the alignment of two sequences by dynamic time warping (DTW) [17]. Here, however, we need to align the hidden states in the presence of many training sequences.

The underlying idea of the synchronization technique is to set up local exemplar HMMs $\boldsymbol{\lambda}^{\boldsymbol{\phi}}$ by using the invariance of HMMs to temporal variations. We proceed in two steps: firstly, a global HMM $\boldsymbol{\lambda}$ is trained based on the whole training set $\mathcal{X}$ of different parameterizations $\boldsymbol{\phi}$ of the same action. This is done using the EM algorithm mentioned in Sect. III-A. Such a global HMM is sketched in Fig. 2, left, by the light gray Gaussians. The situation that actions of different parameterizations are covered in such a symmetrical way as in Fig. 2 can be enforced by ensuring that the hidden state sequences pass the trajectory always in the sequential order from state 1 to N. This is done (a) by choosing left-right HMMs to model the movement, and (b) by allowing only state sequences that start in the first and end in the last state. In addition, (c) the invariance of the HMM to temporal variations, even though necessary, needs to be somehow constrained. We accomplish this by adding explicit time durations to the states of the HMM [11]. This is similar to constraining the warping in standard DTW approaches [17]. Here, by using explicit state durations, one can prevent that one state generates only one output for one sequence and a lot of outputs for another sequence. Otherwise, one would end up with an improper alignment of the sequences (see [17], Fig. 2 C, for an improper, and B for a proper alignment). To avoid the numerical problems associated with scaling [11], we replaced each state of the left-right HMM with pseudo states that share one Gaussian (see Fig. 3). This forces the hidden states sequences to stay in a state, e. g., as in Fig. 3, for at least two and for maximal three time steps.

In the second step, we consider the partial training set $\mathcal{X}^{\boldsymbol{\phi}}$ associated with a specific parameterization $\boldsymbol{\phi}$. Using this data set, we estimate the local HMM $\boldsymbol{\lambda}^{\boldsymbol{\phi}}$ again by using the EM algorithm, but now, we use the parameters of the global HMM $\boldsymbol{\lambda}$ as an initial value for the EM algorithm. In addition, to preserve the state alignment of the local HMM as it is given by the global HMM, we fix the means after the first EM step. In the following, we will exemplify, that this adapted EM procedure gives a proper state alignment of the local HMMs: In the first E step of the EM algorithm, the posterior probabilities $\gamma_t^k(i) = \mathrm{P}(q_t = i|\boldsymbol{X}^k, \boldsymbol{\lambda})$ of being in state i at time t are computed for each sequence $\boldsymbol{X}^k = \boldsymbol{x}_1^k \ldots \boldsymbol{x}_T^k$ of the training set $\mathcal{X}^{\boldsymbol{\phi}}$. This is done based on the current parameter values, which are at this point the values of the global HMM. Thus, $\gamma_t^k(i)$ is the "responsibility" of state i for generating $\boldsymbol{x}_t^k$, as given by the global HMM. In the M step of the EM algorithm, each $\boldsymbol{\mu}_i$ of a Gaussian of state i is re-estimated as $\gamma_t^k(i)$-weighted mean:

$$\boldsymbol{\mu}_i = \frac{\sum_{t,k} \gamma_t^k(i)\boldsymbol{x}_t^k}{\sum_{t,k} \gamma_t^k(i)} \tag{2}$$

Now, consider Fig. 2. The responsibilities $\gamma_t^0(i)$ of the upper sequence $\boldsymbol{x}_1^0\boldsymbol{x}_2^0 \ldots \boldsymbol{x}_7^0$ for $t = 1$, 2 are large for $i = 1$ but small for $i > 1$ under consideration of the position of the Gaussian $\mathcal{N}_1$ of the global HMM $\boldsymbol{\lambda}$. Thus, $\boldsymbol{\mu}_1^0$ of $\mathcal{N}_1^0$ of the local HMM $\boldsymbol{\lambda}^0$, as calculated by Eq. (2), lies as desired between $\boldsymbol{x}_1^0$ and $\boldsymbol{x}_2^0$. Similarly, $\boldsymbol{\mu}_1^1$ of the local HMM $\boldsymbol{\lambda}^1$ computed based on the lower sequence $\boldsymbol{x}_1^1\boldsymbol{x}_2^1 \ldots$ would lie between $\boldsymbol{x}_1^1$ and $\boldsymbol{x}_2^1$. Hence, the alignment of $\boldsymbol{\mu}_1^0$ and $\boldsymbol{\mu}_1^1$ of the local HMMs $\boldsymbol{\lambda}^0$ and $\boldsymbol{\lambda}^1$ as given by the global HMM $\boldsymbol{\lambda}$ is ensured.

2) Synthesis, Recognition, and Parameters: At first we consider the synthesis in the general case of a parametric movement parameterized by $\boldsymbol{\phi} \in \mathbb{R}^d$. Let $\boldsymbol{\phi}$ be the parameterization of the movement that needs to be generated. For synthesis we need properly synchronized HMMs $\boldsymbol{\lambda}^{\boldsymbol{\phi}_n}$ trained for movements with parameterization $\boldsymbol{\phi}_n$ for the 2^d corners $\boldsymbol{\phi}_n$ of a d-dimensional cuboid (or at least a warped version of such an cuboid) that contains the required parameterization $\boldsymbol{\phi}$. Since $\boldsymbol{\phi}$ is inside the cuboid, there exist interpolation parameters $\boldsymbol{\omega}(\boldsymbol{\phi})$ such that $\boldsymbol{\phi}$ can be expressed as a d-linear combination of $\boldsymbol{\phi}_n$ with parameters between 0 and 1. Lets denote by $\boldsymbol{\lambda}^{\boldsymbol{\phi}}$ the component wise d-linear interpolation of the HMMs $\boldsymbol{\lambda}^{\boldsymbol{\phi}_n}$ interpolated with interpolation parameters $\boldsymbol{\omega}(\boldsymbol{\phi})$. This $\boldsymbol{\lambda}^{\boldsymbol{\phi}}$, based on the local HMMs $\boldsymbol{\lambda}^{\boldsymbol{\phi}_n}$, is what we call a parametric HMM (PHMM). The calculation of the parameters $\boldsymbol{\omega} = (u, v, w)$ in the case of 3-D, or (u, v) in the case of 2-D parameterization interpolation, which we used in the experiments, is straight-forward, see Sect. IV. The bilinear interpolation formula is given by (5), the trilinear by (4), the extension to the d-linear case can be constructed easily. The movement trajectory $\boldsymbol{f}(t)$ for a specific parameterization $\boldsymbol{\phi}$ can be synthesized using, e. g., linear spline interpolation of the sequence of means $\boldsymbol{\mu}_1^{\boldsymbol{\phi}}\boldsymbol{\mu}_2^{\boldsymbol{\phi}}\boldsymbol{\mu}_3^{\boldsymbol{\phi}}\ldots$ of $\boldsymbol{\lambda}^{\boldsymbol{\phi}}$ with respect to the expected time durations encoded in the transition matrix of $\boldsymbol{A}^{\boldsymbol{\phi}}$.

The recognition of the type of movement and its parameterization can be accomplished as follows. For each class k we have a PHMM $\boldsymbol{\lambda}_k^{\boldsymbol{\phi}}$ that represents a parametric movement. Now, lets assume that we need to classify a sequence $\boldsymbol{X}$. We start by estimating the most likely parameter $\boldsymbol{\phi}_k$ for each possible parameterized action class k. This involves maximizing the likelihood functions:

$$\boldsymbol{\phi}_k = \underset{\boldsymbol{\phi}\in[0-\varepsilon,1+\varepsilon]^d}{\arg\max}\ \mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\boldsymbol{\phi}}), \tag{3}$$

The class identity is given by the class for which the likelihood $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\boldsymbol{\phi}})$ is the highest. In addition, the associated parameter $\boldsymbol{\phi}_k$ gives the most likely parameterization. In the experiments we maximized the log likelihood functions $l_k(\boldsymbol{\phi}) = \log \mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\boldsymbol{\phi}})$ (see Fig. 9 (c)). This is done using the gradient descent method and numerical derivatives of $l_k(\boldsymbol{\phi})$.

In our evaluation of the movement representation, the parameterization is given by 2-D coordinates in the plane of a table-top. In that experiment we have up to 3×3 local HMMs for each movement type which form a regular raster. In this case the first guess is always based on the HMMs defined at the outermost positions, then the estimate is refined based on the HMMs with parameter positions that define the smallest rectangle which includes the first guess.

IV. Transfer of Movements to the Robot

In this section we consider how to generate arbitrary robot reaching and pointing movements using PHMMs. Our approach can be easily used for other types of parametric movements. For training we use example reaching and pointing movements that stretch out or point to a number of different grasping positions distributed in the robot workspace. An example of the workspace is shown in Fig. 4, where the eight target $\boldsymbol{t}_{ijk}$ positions form a cuboid. This way, the location of the gripper on the table as well as up to the certain height above the table can be controlled. The training movements are used as explained in III to train the PHMM. The PHMM is based on eight local HMMs, one for each of the eight example positions. Note that at each training position, several movements are used to train the HMM. The training step assures that the HMMs are all synchronized for interpolation.

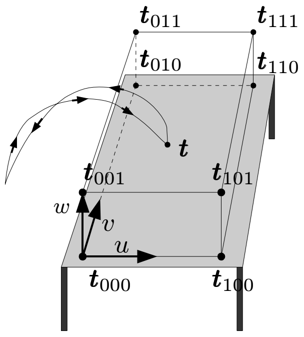


Fig. 4. *Target Points of Movements* at a table-top. In addition, a finger trajectory of a pointing movement with target point $\boldsymbol{t}$ is shown. The coordinates u, v, w are used for interpolation.

In our experiment, the recorded reaching and pointing movements start and stop at the same base position (hand beside hip), i. e. we recorded both the reaching for the object or pointing movement and the withdrawing movement. In order to be able to synthesize realistically looking sequences of several pointing and reaching movements without discontinuities if played back in sequence, the base pose at the beginning and at the end of the motions is normalized. This is done by blending the trajectories with respect to the mean starting position.

A. Synthesis of Robot Movements for Specific Positions

For each movement type we have a PHMM of eight local HMMs trained for specific movements to the targets $\boldsymbol{t}_{ijk}$, like shown in Fig. 4. The recorded movements are a stacked vector $\boldsymbol{x} = (\boldsymbol{p}^\top, \boldsymbol{q}^\top, \ldots)^\top$ of the trajectories of the right arm shoulder $\boldsymbol{p}$, elbow $\boldsymbol{q}$, thumb, finger, and its knuckle.

Since we consider the trajectories relative to the shoulder position we calculate the mean shoulder position over all example trajectories and use this as reference. Now, consider an arbitrary target $\boldsymbol{t} = \boldsymbol{t}^{uvw}$ in the workspace that is given by interpolating the corners $\boldsymbol{t}_{ijk}$ by trilinear interpolation with the parameters (u, v, w):

$$\boldsymbol{t}^{uvw} = \bar{w}\boldsymbol{t}^{uv0} + w\boldsymbol{t}^{uv1} \tag{4}$$
$$\boldsymbol{t}^{uv0} = \bar{v}(\bar{u}\boldsymbol{t}_{000} + u\boldsymbol{t}_{100}) + v(\bar{u}\boldsymbol{t}_{010} + u\boldsymbol{t}_{110}) \tag{5}$$
$$\boldsymbol{t}^{uv1} = \bar{v}(\bar{u}\boldsymbol{t}_{001} + u\boldsymbol{t}_{101}) + v(\bar{u}t_{011} + t\boldsymbol{t}_{111}), \tag{6}$$

where

$$\bar{w} = 1 - w, \quad \bar{v} = 1 - v, \quad \bar{u} = 1 - u.$$

Then a movement $\boldsymbol{f} = (\boldsymbol{p}^\top, \boldsymbol{q}^\top, \ldots)^\top$ for the target position $\boldsymbol{t}$ can be synthesized as described in Sect. III-B2. Essentially, that is interpolating each component (shoulder, elbow, ...) of the means of the corresponding states of the local HMMs as given by the interpolation formula (4). The interpolation parameters (u, v, w) for a specific target $\boldsymbol{t}$ are easily calculated by

$$u = \frac{\boldsymbol{t} - \boldsymbol{t}_{000}}{|\boldsymbol{t} - \boldsymbol{t}_{000}|} \cdot \frac{\boldsymbol{t}_{100} - \boldsymbol{t}_{000}}{|\boldsymbol{t}_{100} - \boldsymbol{t}_{000}|}, \tag{7}$$

$$v = \frac{\boldsymbol{t} - \boldsymbol{t}_{000}}{|\boldsymbol{t} - \boldsymbol{t}_{000}|} \cdot \frac{\boldsymbol{t}_{010} - \boldsymbol{t}_{000}}{|\boldsymbol{t}_{010} - \boldsymbol{t}_{000}|}, \tag{8}$$

$$w = \frac{\boldsymbol{t} - \boldsymbol{t}_{000}}{|\boldsymbol{t} - \boldsymbol{t}_{000}|} \cdot \frac{\boldsymbol{t}_{001} - \boldsymbol{t}_{000}}{|\boldsymbol{t}_{001} - \boldsymbol{t}_{000}|}. \tag{9}$$

Since the robot is smaller than the demonstrator, these trajectories are scaled to fit to the overall arm length of the robot. Of course the distances, e. g. between the wrist and elbow, are not preserved by interpolation, but become slightly shorter or longer than the true limb lengths. However, this does not matter if the movements are transferred to the robot as described in Sect. IV-B.

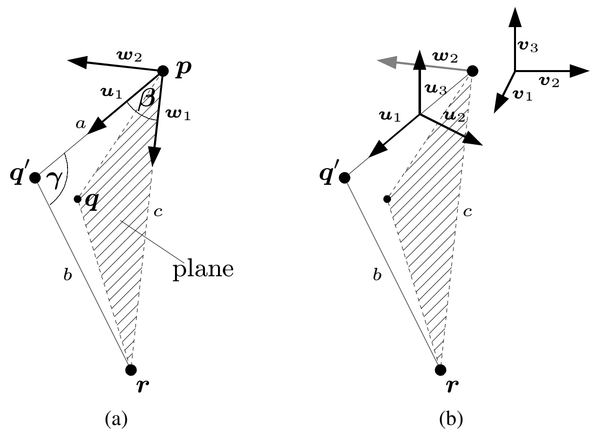


Fig. 5. In figure (a), the points $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$, which represent the shoulder, elbow, and hand, define a plane, on which the elbow $\boldsymbol{q}'$ of the robot should lie. Figure (b) shows the two reference coordinate systems $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)$ and $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$. The coordinate system $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)$ is the reference of the upper arm, whereas $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$ is fixed and does not move with the upper arm.

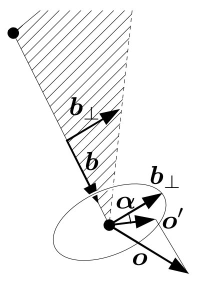


Fig. 6. *Rotation of the Robot's Wrist.* The twist angle $\alpha = \measuredangle(\boldsymbol{b}_\perp, \boldsymbol{o}')$ of the wrist is calculated in the twist plane.

B. Calculation of Joint Angles

The interpolation of Sect. IV-A results in an arm trajectory which is described by the Cartesian trajectories of the shoulder, elbow, wrist, thumb, index-finger and its knuckle. In this section, we will describe how the robot joint angles of the upper arm are computed from this data. The joints under consideration are the shoulder joint with three degrees of freedom, and the elbow joint with one degree of freedom. These values are determined only on the basis of the human elbow and hand position relative to the shoulder. The hand position $\boldsymbol{r}$ (Fig. 5) is given as the mean of the human thumb, index finger, and knuckle of that finger. Furthermore, the orientation of the hand is given by the thumb and the finger.

To account for a difference in size between the human demonstrator and the robot, human data is scaled to the size of the robot as described in Sect. IV-A. For the calculation of joint angles, only the relative positions of the elbow $\boldsymbol{q}$ and hand $\boldsymbol{r}$ with respect to the shoulder $\boldsymbol{p}$ are of interest. The shoulder can therefore be assumed to be at the origin ($\boldsymbol{p} = \boldsymbol{0}$); and the human hand and elbow positions, $\boldsymbol{r}$ and $\boldsymbol{q}$, of Fig. 5 (b) are appropriately scaled for the robot.

The four joint angles of the robot arm are calculated so that the robot hand is at the same position as the scaled human hand, and that the elbow $\boldsymbol{q}'$ is in the plane defined by three points $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$ of the human demonstrator. The elbow position $\boldsymbol{q}'$ of the robot differs from the (scaled) elbow position of the demonstrator if the proportions of the robot, i. e. the upper arm compared to the forearm, differ from those of the demonstrator.

To calculate the shoulder angles (see Fig. 5 (a)), the direction $\boldsymbol{u}_1$ of the elbow $\boldsymbol{q}'$ is required. The angle γ of the elbow is defined by the length a of the upper arm, the length b of the lower arm (elbow – hand) of the robot, and the distance $c = \overline{\boldsymbol{pr}}$ from the shoulder $\boldsymbol{p}$ to the target point $\boldsymbol{r}$. We calculate γ as

$$\gamma = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right), \tag{10}$$

and

$$\beta = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right). \tag{11}$$

Now, $\boldsymbol{u}_1$ is given by

$$\boldsymbol{u}_1 = \cos\beta \cdot \boldsymbol{w}_1 \; + \; \sin\beta \cdot \boldsymbol{w}_2, \tag{12}$$

where

$$\boldsymbol{w}_1 = \overrightarrow{\boldsymbol{pr}} \,/\, \overline{\boldsymbol{pr}} \tag{13}$$

$$\boldsymbol{w}_2 = \boldsymbol{w}_2' \,/\, |\boldsymbol{w}_2'| \tag{14}$$

$$\boldsymbol{w}_2' = \overrightarrow{\boldsymbol{pq}} - \langle \overrightarrow{\boldsymbol{pq}}, \boldsymbol{w}_1 \rangle \, \boldsymbol{w}_1. \tag{15}$$

For HOAP-3, the three shoulder joint angles can be calculated as Cardan angles for the rotation between the coordinate systems $\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{u}_i\}$ (see Fig. 5 (b)). The Cardan angles[1] ϕ, θ, ψ are calculated using the rotational matrix between the

[1]Cardan angles ϕ, θ, ψ define a rotation $\boldsymbol{R}$ as $\boldsymbol{R} = \boldsymbol{R}^{\boldsymbol{z}}_{\psi} \boldsymbol{R}^{\boldsymbol{y}}_{\theta} \boldsymbol{R}^{\boldsymbol{x}}_{\phi}$.

coordinate systems[2]:

$$\phi = \operatorname{arctan2}(r_{32}, r_{33}) \tag{16}$$
$$\theta = -\arcsin(r_{31}) \tag{17}$$
$$\psi = \operatorname{arctan2}(r_{21}, r_{11}) \tag{18}$$

where

$$\boldsymbol{R} = (r_{ij}) \;=\; [\boldsymbol{u}_1|\boldsymbol{u}_2|\boldsymbol{u}_3]^\top \, [\boldsymbol{v}_1|\boldsymbol{v}_2|\boldsymbol{v}_3]\,. \tag{19}$$

The still unknown vectors $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$ are given by the equations:

$$\boldsymbol{u}_2 = \boldsymbol{u}'_2 \,/\, |\boldsymbol{u}'_2| \tag{20}$$
$$\boldsymbol{u}'_2 = -\boldsymbol{w}_2 - \langle -\boldsymbol{w}_2, \boldsymbol{u}_1 \rangle\, \boldsymbol{u}_1 \tag{21}$$
$$\boldsymbol{u}_3 = \boldsymbol{u}_1 \times \boldsymbol{u}_2. \tag{22}$$

The orientation of the hand was initially extracted from the motion data based on the direction given by the vector from the finger to thumb. However, it turns out to be better to align the robot gripper to the plane of the table, at least for that part of the motion which is used to interact with the objects. The degree of opening or closing the robot gripper was initially set to the distance between the finger and the thumb of the human. However, it turns out to be better to use maximal gripper opening while interacting with an object, as we used objects of different shapes in the recording session.

Since the wrist of the HOAP-3 has only one degree of freedom (twist), only the projection of the orientational direction vector $\boldsymbol{o}$ (finger$\longrightarrow$thumb) onto the rotation plane of the wrist is used to set its twist angle. This is illustrated in Fig. 6. The joint angle is given by (see Fig. 6):

$$\alpha = \arccos\left(\frac{\langle \boldsymbol{b}_\perp, \boldsymbol{o}' \rangle}{|\boldsymbol{b}_\perp|\,|\boldsymbol{o}'|}\right) \cdot \operatorname{sign}(\langle \boldsymbol{o}', \boldsymbol{b} \times \boldsymbol{b}_\perp \rangle), \tag{23}$$

where

$$\boldsymbol{o}' = \boldsymbol{o} - \frac{\langle \boldsymbol{o}, \boldsymbol{b}_\perp \rangle}{|\boldsymbol{b}_\perp|^2} \boldsymbol{b}_\perp. \tag{24}$$

C. Experiments with a Humanoid Robot HOAP-3

We tested the effectiveness of our approach by implementing a task involving a number of objects that are first associated with different openings on the table and need to be placed into the correct opening (see also Fig. 7 (left)). An online demo is available via web page [2]; the recording of demonstrated movements is not included. The experiment proceeds in three phases:

- The PHMMs for reaching and pointing movements are learnt.
- Human demonstrator shows to the robot which object belongs to which hole. The robot associates each object with the appropriate hole.
- Afterwards, the objects are placed again at arbitrary positions. In this phase, the human points at one of the object, which is then identified and placed into the associated hole using the previously learnt PHMMs. This is repeated until all objects are removed.

To technically implement the placement of an object into the appropriate hole, the robot first estimates the transformation between the robot and the camera coordinate system by moving the hand of the robot to at least four different configurations, which is enough to calculate the transformation between two camera systems. After the object is identified, the robot reaches for the object using the learnt PHMM. We use here only a part of the interpolated movement corresponding to the motion before the grasp. To be able to grasp the object, the robot reaches towards the position in front of the object (see Fig. 8). Relocating the object from the current position on the table to the hole is accomplished by generating a sequence of reaching positions on the table and using the PHMM to generate the corresponding reaching movements for these positions. The configurations where the robot reaches for these positions are extracted and interpolated to generate the relocation trajectory. In this way we ensure that the robot uses only natural arm postures. Grasping and releasing of the object is implemented using standard robotics methods. The withdrawing movement is realized in the same way as the reaching movement, the only difference being that in this case we use the part of the trajectory after the grasp.

V. Evaluation of Movement Representation

In the evaluation we focus our considerations on pointing (see Fig. 1) and reaching actions, which are not only the most important movements in our behavioral experiment but are also important in other interaction scenarios. Both are performed in a very similar way, starting and ending in the same base position (arm along the body). The motion data of our systematic evaluation is acquired using an eight camera Vicon system with cameras running at 60 Hz (see Fig. 1). The recognition and synthesis experiments are based on seven 3-D points located at different segments of a human body. The seven data points are: sternum; shoulder and elbow of the right arm; knuckles, index finger, and thumb of the right hand.

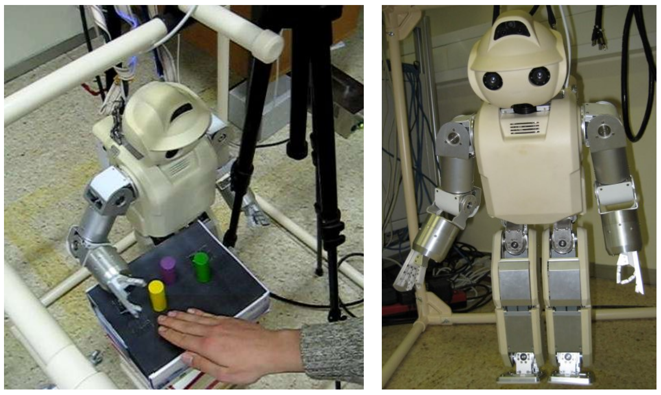


Fig. 7. *Our Experimental Setup.* A person advises the robot HOAP-3 how to clean up objects. *Online Demo*, available via web page [2].

[2]Contrary to $\arctan(a/b)$, $\operatorname{arctan2}(a,b) \in (-\pi, \pi]$ respects the signs of a and b.

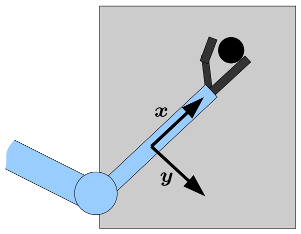


Fig. 8. *Approach Motion* along the x-axis to grasp an object

The exemplar positions at table-top form a regular raster, which covers a region of 80cm $\times$ 30cm (width $\times$ deeps). For training, a 3×3 raster is used, where 10 repetitions have been recorded for each exemplar position and each action type (pointing, reaching). For evaluation, a 5×7 test-raster is used with 4 repetitions for each raster position to allow a good evaluation statistic. All in all, for testing we used several hundreds of movements.

A. Training: Setup of PHMMs

Training and setup of the exemplar PHMMs for grasping and pointing is done as described in Sec. III-B1. We used PHMMs of 20 states, where the hidden state sequences are forced to stay between 4 and 6 time steps in each state. The training sequences are normalized to 100 samples. We train the PHMMs based on data of the full 3×3 raster (9 exemplar HMMs) or based on a 2×2 raster, which consists of the four outermost exemplar positions of the 3×3 raster at table-top. These PHMMs will be refered in the following as 3×3 or 2×2 PHMM of reaching or pointing.

B. Synthesis

Synthesis is done as described in Sec. III-B2 with the setup described in the section above. The performance of synthesis is systematically evaluated by plotting the synthesis error for each of the positions of the 5×7 test-raster. Therefore, the error is based on the distance between each synthesized movement and an average of the four test exemplars of a test-raster positions. The averaging is done by training and re-synthesizing from an 80 state HMM. Again, the re-synthesis is a function $\bar{\boldsymbol{f}}(t) = (\bar{\boldsymbol{f}}_i(i))_{i=1}^{7}$ of "stacked" 3-D trajectories $\boldsymbol{f}_i(t)$ (elbow, wrist, …). The final error ε is calculated as the root-mean-square error between the time warped synthesis, $\boldsymbol{f}(t)$, and re-synthesized average, $\bar{\boldsymbol{f}}(t)$:

$$\varepsilon = \sqrt{\int \sum_{i=1}^{7} \frac{(\boldsymbol{f}_i(\alpha(t)) - \bar{\boldsymbol{f}}_i(\bar{\alpha}(t)))^2}{7} \mathrm{d}t \Big/ \int \alpha(t) dt}, \quad (25)$$

where $\alpha(t)$ and $\bar{\alpha}(t)$ are warping functions. As the starting and ending points of the reference $\bar{\boldsymbol{f}}(t)$ sometimes vary slightly, the first and last 10% of the sequences are not considered. Obviously, the error ε is normalized w. r. t. the length of the sequence.

The Fig. 9 (a), and (b) compare the synthesis errors of 2×2 and 3×3 exemplar PHMMs. Clearly, the performance in the middle of the covered region increases, if the 3×3 PHMM is used. Fig. 9 (c), and (d) show the fact that the results for the reaching action are very similar to the pointing actions. If the outer regions are neglected the synthesis errors are approximately 1.8cm both in the case of reaching and pointing.

C. Recognition

For recognition, we evaluate (a) the performance of estimating the parameter of a movement, (b) the rate of the correct classifications of movement types, and (c) the robustness to noise.

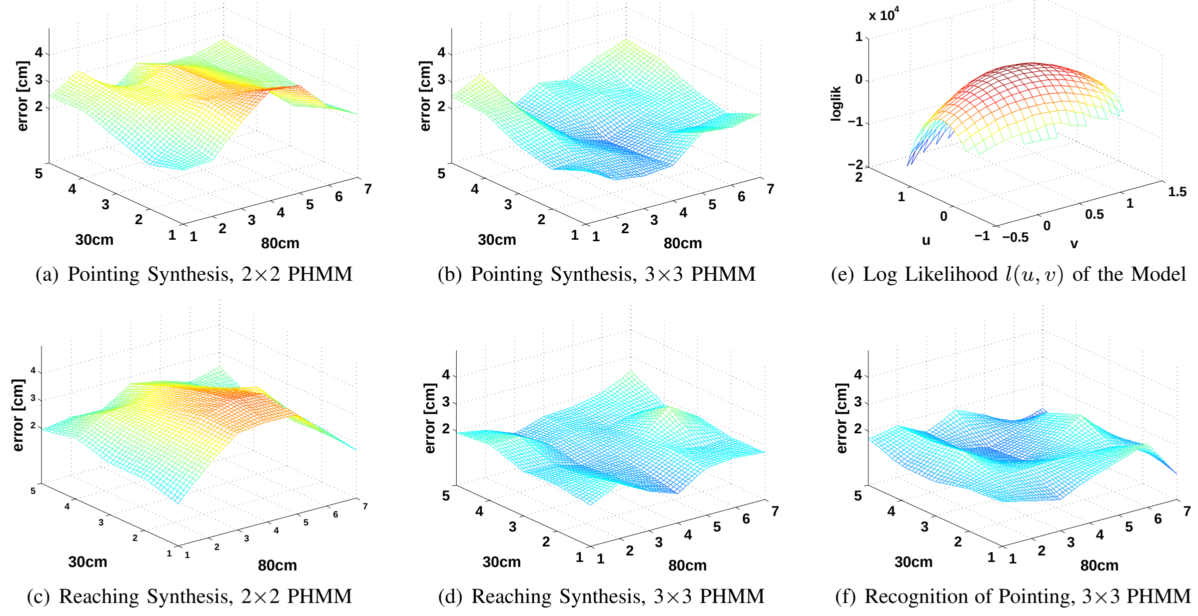
First it is worth to take a look at Fig. 9 (e), which shows that the optimization problem of maximizing the log likelihood function $l(u, v) = \log P(\boldsymbol{X}|\boldsymbol{\lambda}^{uv})$ given a movement is tractable by standard optimization techniques (smoothness and strict concavity). Thus, the most likely parameterization (u, v) can easily be estimated. The errors for each position of the 5×7 test-raster are calculated as the average deviation of the estimated position and the ground truth position for the test example movements. The estimation accuracy of the table-top positions behaves very similar to the results of the synthesis Fig. 9 (f). The performance increases similar to the synthesis in the inner region for our 3×3 PHMM compared to the 2×2 PHMM (not depicted). The rate of correctly classified types of the 280 grasping and pointing test movements is 94% for the 2×2 PHMMs, and changes just insignificantly for the 3×3 PHMMs.

We tested the robustness of estimating the parameters of movements by adding Gaussian noise to each component of the samples of the movements. Here, we realized no significant influence for independently distributed noise with $\sigma < 15$cm. Obviously, that is due to a large number of samples in the sequence.

VI. Conclusions

We have presented a novel approach to represent, classify, and imitate parametric movements using parametric hidden Markov models. Our approach contains several contributions: (a) how to learn and represent parametric human movements, (b) how to use this representation for action recognition, (c) how to transform and project the actions onto the embodiment of the robot, and (d), how to generate the actions on a new embodiment. Furthermore, we have solved several subproblems such as multi-dimensional time warping of the multiple training sequences so that HMMs can be properly interpolated.

We systematically evaluated the synthesis and recognition performance of the proposed PHMM framework. The experiments show the accuracy of our approach for the generation of new movements and for the estimation of the associated movement parameters (errors of ≈2cm). This shows that the newly generated movements are similar to the observed movements. This also was confirmed in the behavioral experiment of Sec. IV-C, where the generated reaching movements were similar to the training examples, and, like the examples, avoided collisions with the table. This could not be guaranteed if the movement was generated by standard robotics approaches. It is



(a) Pointing Synthesis, 2×2 PHMM (b) Pointing Synthesis, 3×3 PHMM (e) Log Likelihood $l(u, v)$ of the Model

(c) Reaching Synthesis, 2×2 PHMM (d) Reaching Synthesis, 3×3 PHMM (f) Recognition of Pointing, 3×3 PHMM

Fig. 9. *Error of Synthesis* (a-d) *and Recognition* (f): evaluated at 7×5 raster positions for different PHMMs resolutions, a plot of the log likelihood for the PHMM parameters (u, v) is given in (e).

worth to note that the synthesis error does not affect the robot-object interaction since we interpolate based on the gripper position which is encoded in the movements.

The classification rate of the type movement is $\approx 94\%$. It is worth pointing out that this recognition rate is achieved without any kind of diagnostic features. Furthermore, it should be noticed that the movements of pointing and reaching are very similar. In earlier experiments we had trained classical HMMs on pointing and reaching actions where the training movements were directed similarly (up to the natural variance of human performances). During the tests, our classical HMMs reached a recognition rate of $\approx 85\%$. The fact that the PHMMs lead to considerably better recognition rates shows that they are much better in describing the actions and in compensating for natural variability of the performances.

We conclude that PHHMs are suitable for imitation because they are both generative and accurate for recognition.

Acknowledgment

This work was partially supported by EU through grant PACO-PLUS, FP6-2004-IST-4-27657.

References

[1] A. D. Wilson and A. F. Bobick, "Parametric hidden Markov models for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 884–900, 1999.

[2] D. Herzog, A. Ude, and V. Krueger, "Hoap-3 online demo." [Online]. Available: http://www.cns.atr.jp/~aude/rules.mpg

[3] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: A review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[4] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, Mass.: MIT Press, 2003, pp. 1547–1554.

[5] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research: The Eleventh International Symposium*, P. Dario and R. Chatila, Eds. Berlin, Heidelberg: Springer, 2005, pp. 561–572.

[6] J. Tani, M. Ito, and Y. Sugita, "Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using RNNPB," *Neural Networks*, vol. 17, no. 8-9, pp. 1273–1289, 2004.

[7] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *Int. J. Robotics Research*, vol. 23, no. 4-5, pp. 363–377, 2004.

[8] A. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, pp. 370–384, 2006.

[9] T. Asfour, F. Gyarfas, P. Azad, and R. Dilmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Genoa, Italy, December 2006, pp. 40–47.

[10] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[11] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4–15, January 1986.

[12] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[13] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng, "Goal-directed action synthesis from a library of example movements," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Pittsburgh, Pennsylvania, December 2007.

[14] A. Safonova and J. Hodgins, "Analyzing the physical correctness of interpolated human motion," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2005, pp. 171–180.

[15] T. Moeslund, A. Hilton, and V. Krueger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–127, 2006.

[16] S. Gunter and H. Bunke, "Optimizing the number of states, training iterations and Gaussians in an HMM-based handwritten word recognizer," *icdar*, vol. 01, p. 472, 2003.

[17] E. Keogh and M. Pazzani, "Derivative dynamic time warping," in *Proceedings of the 1st SIAM International Conference on Data Mining (SDM'2001), Chicago, USA*, 2001.