Project no.:
IST-FP6-IP-027657

Project full title: **Perception, Action & Cognition through Learning of Object-Action Complexes**

Project Acronym: **PACO-PLUS**

Deliverable no.: D3.1.1

Title of the deliverable: Aspects of vision based upper body and grasp activity recognition

Contractual Date of Delivery to the CEC:	**31st January 2008**
Actual Date of Delivery to the CEC:	**31st January 2008**
Organisation name of lead contractor for this deliverable:	**Aalborg University (AAU)**
Author(s):	**Volker Krüger, Danica Kragic, Hedvig Kjällström, Ales Ude, Pedram Azad, Tamim Asfour**
Participants(s):	**AAU, KTH, UniKarl, JSI, UEDIN**
Work package contributing to the deliverable:	**WP3.1**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**28**
Start date of project:	1st Feb. 2006
Duration:	48 month

Projectco-funded by the European Commission within the Sixth Framework Programme (2002-2006)
Dissemination Level

PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This report contains several scientific publications that are relevant for deliverable D3.1.1. Deliverable D3.1.1 is concerned with the recognition and synthesise of goal-directed arm movements and hand grasps of objects. The included publications discuss different aspects of this area, including vision based tracking, learning and synthesizing of arm movements and grasps.

Keyword list: action recognition, action representation, computer vision, robotics, AI

Contents

1 Introduction

Deliverable D3.1.1, *Aspects of Vision Based Upper Body and Grasp Activity Recognition*, is concerned with two aims:

1. To become able to identify human upper body movements and actions which may include objects. Examples of such movements are humans pointing at objects or humans grasping objects. Here, two sub-issues are important:

   (a) Action recognition: Here, the aim is to recognize that a human, e.g., points or grasps an object. The problem of recognition is *relatively* simplified as in many situations, approximate information is sufficient. For example the mere fact that an object moves in 3D space in the same manner as the hand is sufficient to show that the object was grasped. How the grasping precisely was performed is often not important. See for a discussion [5] and the demo [3].

   (b) Learning from demonstration: Here, the aim is to learn movements and actions from observation. Contrary to the recognition task above, this task is more demanding as it matters how the action is precisely executed. For example, what kind of grasping is performed needs to be considered here so that later, the robot can perform such grasps itself.

2. o become able to synthesize actions based on those seen before. A typical example is a situation where a human teacher has explained to a robot to perform a certain action by having demonstrated it (learning from demonstration). A general problem in this case is that new, upcoming situations will not perfectly match the situation during training. Objects may be at different locations and the robot itself may be positioned differently from before. Thus, the robot has to learn not the *precise* trajectory of the movement but a more abstract representation of the action such as *grasp that type of object and insert it into a box*, where, e.g., *grasp* implies the appropriate and specific hand grasp at the position of the object.

In this reporting period, we have focused on the following aspects[1]:

1. Development of a new learning framework for synthesized *goal-directed* actions from example movements [8]. The approach is based on memorizing of training data and locally weighted regression to compute suitable movements for a large range of situations. This method is especially useful for performances of very *precise movements*, such as the employed test movement of throwing a ball into a basket.

2. Development of a representation that allows to a) recognize movements performed by other humans independent from the parameterizations of the movements and b) synthesize movements based on given parameters [5]. For example a human may point to an object on a table. The pointing movements is parameterized through the object position on the table. The movement as well as its parameterization can be recognized. Similarly, a movement can be synthesized with a given parameterization. For example, the robot may grasp the object at which a human has pointed earlier and the parameterization of the newly synthesized movement is done with the parameters previously recovered from the observed movement that was performed by the human.

3. Human motion capture is a requirement for human action learning and recognition to work online. In [1] we have developed further our marker-free human motion capture

[1] These are described in greater detail in the same order in Sect. 2-6

approach. A sub-aspect of marker-free human body motion capture was discussed in [2]. Marker-free human body tracking in conjuction with [5] allows online action recognition as demonstrated in demo (see [3]).

4. The work in [5] is concerned with the learning and recovery of arm movements in general but does not discuss the learning and recovery of the precise grasps. The grasping itself is a problem on its own and is discussed in [7]. To incorporate this into one joint demo is ongoing work.

5. Finally, we were able to show in [6] that the recognition of actions can be greatly eased when decoupling complex actions into motor primitives.

The following sections give an overview and summary of the appended publications in the context of this deliverable. The order in which the publications are appended follows the order of the subsequent sections.

2 New learning framework for synthesized *goal-directed* actions from example movements

Learning actions on objects without the availability of prior physical models is a difficult problem. While many tasks can be learned assuming a proper representation for the physics of the task, such an approach relies on a priori knowledge about the task and therefore does not solve the complete learning problem. If physical models are not available, the search space becomes very large. Learning from demonstration or imitation learning has been proposed to effectively reduce the search space. However, in tasks involving the manipulation of objects, it is not possible to directly replicate the demonstrated movements. Instead, they need to be adapted to the current state of the 3-D world. For any given situation, it is highly unlikely that an appropriate movement would be observed in advance and included in the library.

In [8] we have developed a new learning framework for synthesizing *goal-directed actions* from example movements. The approach is based on the memorization of training data and locally weighted regression to compute suitable movements for a large range of situations. The proposed method avoids making specific assumptions about an adequate representation of the task. Instead, we use a general representation based on fifth order splines. The data used for learning comes either from the observation of events in the Cartesian space or from the actual movement execution on the robot. Thus it informs us about the appropriate motion in the example situations. We show that by applying locally weighted regression to such data, we can generate actions having proper dynamics to solve the given task. To test the validity of the approach, we present simulation results under various conditions as well as experiments on a real robot. Our experiments demonstrate that we can achieve fairly accurate results without providing the system with models about the dynamics of the task and without needing to acquire an excessive amount of example movements.

3 Recognizing and Synthesizing Parametric Movements

The aim of the above work is to enable a robot to execute precise movements (such as throwing something into a basket).

In another work we focus also on learning action, however, with the focus of recognizing classes of actions. We developed an exemplar-based parametric hidden Markov model (PHMM) that allows to represent, e.g., movements of a particular type (e.g. pointing actions) and that compensates for the different appearances and parameterizations of that movement. The PHMM

is based on exemplar movements that have to be "demonstrated" to the system. Recognition and synthesis are carried out through locally linear interpolation of the exemplar movements.

In order to be able to interpolate several HMMs, we have developed a new training method that allows to train several differently actions exemplars at the same time. Dynamic Time warping, which is commonly used, does not allow to synchronize more than two signals at a time. Our new approach allows to synchronize any number of signals during the EM training for the HMMs.

In our experiments we combine our PHMM approach with our 3D body tracker. Experiments were performed with pointing and grasping movements. The synthesis for grasping was parameterized by the positions of the objects that were to be grasped. In the case of recognition, our approach is able to recover the position of an object at which a human volunteer is pointing. Our experiments show the flexibility of the PHMMs in terms of the amount of training data and its robustness in terms of noisy observation data. In addition, we have compared our PHMM approach to another kind of PHMM, which has been introduced earlier by Wilson and Bobick. The report [5] includes and extends the publication [4].

4 Aspects of Stereo-based Marker-free Human Motion Capture

We continued our work on a stereo-based marker-based human motion capture system that can run in real-time on a humanoid robot, making use of the cameras integrated in its head only [1]. In order to be able to capture movements that include arm poses which result in projections of only very short edges in the camera images, stereo vision is used explicitly to measure 3D positions of the hands and the head. These positions are used as an additional cue to the commonly used edge cue. The combination of these both cues enables tracking arm movements in situations in which a purely edge- and region-based system would fail, while preserving the advantages of contour-based tracking. With this approach, we could successfully track the movements of a person in front of the robot with a processing rate of 15 fps.

In [2] we discuss a sub-aspect of marker-free human body tracking. We combine two classical methods for parameter estimation in this context, namely gradient descent and particle filtering. The combination is done such that (a) the correspondence based estimation gains the advantage of the particle filter and becomes able to follow multiple hypotheses while (b) the particle filter becomes able to propagate the particles in a better manner and thus gets by with a smaller number of particles. We were able to show that pairing up the two methods allows to reduce the number of particles and to increase convergence speed.

5 Recognition and Synthesis of Human Grasps

Extraction and tracking of hand contours represents an important part of sign language and gesture recognition systems. In robotics, recent developments in imitation learning show the need for complete hand pose estimation such that recognition and evaluation of grasps applied to object can be performed.

In [7] We have developed a model based stereo approach. Once hand contours are identified in each of the stereo images, the points along the contours are matched in order to compute their 3D position. The contribution of our work is a method for matching and reconstruction of contour points using Dynamic Time Warping (DTW), giving a 3D reconstruction of the hand contour. This 3D contour can be tracked over time while the hand is performing various object manipulation activities. The estimated configuration of the hand gives information about a number of parameters, such as the type of grasp, the grasp approach vector and the grasping point for the type of object grasped.

As the study showed 3D estimation of hand pose to be an extremely challenging problem, the current work in this area in PACO+ is directed towards 3D pose (or grasp) recognition *without* explicit estimation of the hand pose. The results of this effort will be presented in the next PACO+ deliverable.

6 Modelling and Recognition of Actions through Motor Primitives

In earlier work at KTH, an HMM model was designed for recognition of manipulating actions. The model was based upon the assumption that any action can be described as a sequence of motor primitives. 3D positions and orientations of a number of points of the hand, measured with magnetic 3D sensors, were used as input to the model.

There are two problems with generative approaches like HMM. Firstly, assumptions (possibly erroneous) have to be made about the motion data, like the Markov assumption that the state at the current time step only depends on the previous state and not on any earlier states. Secondly, the motion data has to be segmented prior to classification.

To address these issues, a discriminative sequential classification method was employed [6], conditional random fields (CRF). The performance of the HMM and CRF models was evaluated on a body of magnetic 3D data, and the results showed the CRF to perform slightly better.

References

[1] P. Azad, A. Ude, T. Asfour, and R. Dillmann. Stereo-based markerless human motion capture for humanoid robot systems. In *International Conference on Robotics and Automation (ICRA)*, pages 3951–3956, Rome, Italy, 2007.

[2] D. Grest and V. Krüger. Gradient-enhanced particle filter for vision-based motion capture. In *Proc. ICCV Workshop Human Motion - Understanding, Modeling, Capture and Animation*, pages 28–36, Rio de Janeiro, Brazil, 2007.

[3] D. Herzog, V. Krüger, and D. Grest. www.cvmi.aau.dk → videos.

[4] D. Herzog, V. Krüger, and G. Grest. Exemplar-based parametric hidden markov models for recognition and synthesis of movements. In *Proceedings of Vision, Modeling, and Visualization 2007 (VMV)*, Saarbrücken, Germany, 2007.

[5] D. Herzog, V. Krüger, and G. Grest. Parametric hidden markov models for recognition and synthesis of movements. Technical Report 1, Computer Vision and Machine Intelligence Lab, Aalborg University Copenhagen, 2008. sumbitted to CVPR08.

[6] D. Martínez and D. Kragic. Modeling and recognition of actions through motor primitives. In *International Conference on Robotics and Automation (ICRA), to appear*, 2008.

[7] J. Romero, D. Kragic, V. Kyrki, and A. Argyros. Dynamic time warping for binocular hand tracking and reconstruction. In *International Conference on Robotics and Automation (ICRA), to appear*, 2008.

[8] A. Ude, M. Riley, B. Nemec, A. Kos, T. Asfour, and G. Cheng. Synthesizing goal-directed actions from a library of example movements. In *Proc. IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, Pittsburgh, Pennsylvania, December 2007.

Synthesizing Goal-Directed Actions from a Library of Example Movements

Aleš Ude*†, Marcia Riley¶, Bojan Nemec*, Andrej Kos*, Tamim Asfour‡ and Gordon Cheng†§

*Jožef Stefan Institute, Dept. of Automatics, Biocybernetics and Robotics
Jamova 39, 1000 Ljubljana, Slovenia

†ATR Computational Neuroscience Laboratories, Department of Humanoid Robotics and Computational Neuroscience
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

‡University of Karlsruhe, Institute of Computer Science and Engineering
c/o Technologiefabrik, Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany

§Japan Science and Technology Agency, ICORP Computational Brain Project
4-1-8 Honcho, Kawaguchi, Saitama, Japan

¶Georgia Institute of Technology, College of Computing
Atlanta, Georgia 30332–0250, USA

***Abstract*— We present a new learning framework for synthesizing *goal-directed actions* from example movements. The approach is based on the memorization of training data and locally weighted regression to compute suitable movements for a large range of situations. The proposed method avoids making specific assumptions about an adequate representation of the task. Instead, we use a general representation based on fifth order splines. The data used for learning comes either from the observation of events in the Cartesian space or from the actual movement execution on the robot. Thus it informs us about the appropriate motion in the example situations. We show that by applying locally weighted regression to such data, we can generate actions having proper dynamics to solve the given task. To test the validity of the approach, we present simulation results under various conditions as well as experiments on a real robot.**

I. Introduction

Humanoid robotics has dealt with the problem of learning complex humanoid behaviors for a long time. It was soon realized that to overcome problems arising from high dimensional and continuous perception-action spaces, it is necessary to guide the search process, thus effectively reducing the search space, and also to develop higher-level representations suitable for faster learning. To achieve these goals, researchers in sensorimotor learning have explored various solutions. Some of the most notable among those are learning from demonstration (or imitation learning) and motor primitives.

Building on the large body of work by the computer graphics community, it has been shown that motion capture technology can be used to generate complex humanoid robot motions that may require a great deal of skills and practicing to be realized, e. g. dancing [11], [18], [19]. Techniques to adapt the generated movements with respect to various robot constraints have also been proposed [10], including more complex constraints such as self-collision avoidance and balancing of a free standing dancing robot [8], [14]. Dynamics filter that can create a physically consistent motion from motion capture data has also been proposed [22]. While these works can overcome the problem of different embodiments of the robot and the demonstrator, they do not deal with the effects of motion acting on the external world. Different methods are needed to adapt the captured motions to the changes in the external world and synthesize *goal-directed actions*, such as in the case of object manipulation tasks.

In tasks involving the manipulation of objects, it is necessary to adapt the observed movements to the current state of the 3-D world. For any given situation, it is highly unlikely that an appropriate movement would be observed in advance and included in the library. While many tasks can be learned assuming a proper representation for the physics of the task, such an approach relies on a priori knowledge about the action and therefore does not solve the complete learning problem. To avoid specifying the physical model of the task, Miyamoto et al. [9] based their methodology on programming by demonstration and derived a representation for optimal trajectories, which they referred to as via-points. They were able to teach a robotic arm a fairly difficult game of Kendama and tennis serves. Schaal et al. [5], [17] proposed a more general nonparametric approach based on nonlinear dynamic systems as policy primitives. They developed canonical equations for rhythmic and discrete movements and demonstrated that these systems can be used to learn tasks such as tennis strokes and drumming. Hidden Markov models (HMMs) are another popular methodology to encode and generalize the observed movements [1], [3], [6]. While techniques that enable the reproduction of generalized movements from multiple demonstrations have been proposed, generalization across movements to attain an external goal of the task is not central to these works. HMMs, however, can be used effectively for motion and situation recognition [6] and to determine which control variables should be imitated and how [3].

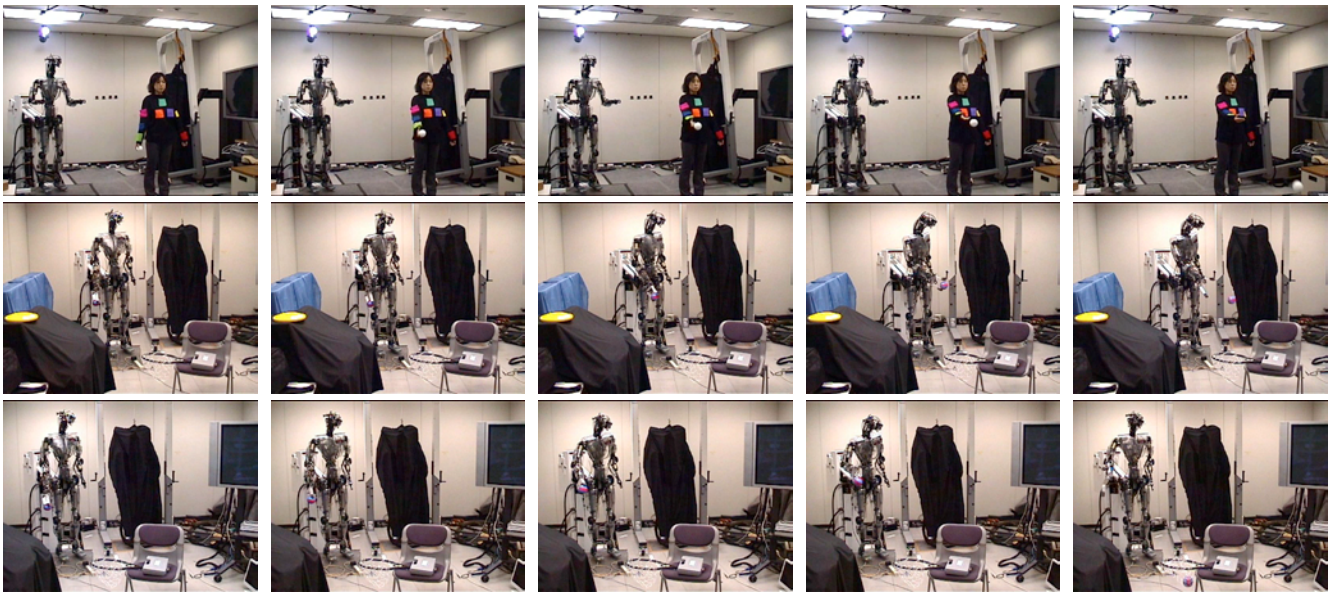

Fig. 1. Human demonstration of the ball throw, unsuccessful direct reproduction on a humanoid robot, and a successful action execution after coaching

The computer graphics community has also studied human motion synthesis from example movements. The most common approach is to generalize across a number of movements by linear interpolation, like e. g. [21]. If done correctly, such an approach results in physically correct movements under many circumstances [15]. Rose et al. [13] represent the motions by B-splines and use radial basis functions to interpolate between the control points of B-Splines. Automatic re-timing of the captured movements based on registration curves has also been considered [7]. Most of the early works dealt with the intepolation of relatively short movements, but interpolation of longer action sequences is also possible as shown in [16]. While these works address many problems relevant to the robotics community, their main aim is to generate realistic computer animations. Our focus, however, is to show that movement interpolation can generate actions that can change the external world in such a way that the goal of an action is attained. In order words, we focus on the synthesis of *goal-directed actions* and how to make action synthesis from example movements applicable for the implementation on a real robotic system.

In the following we propose a new movement generalization methodology based on locally weighted regression [2]. The goal of an action is used to index into the library of stored movements. We also briefly deal with different approaches that can be applied to generate a suitable movement library. We show both in simulation and on a real robot that the proposed approach can be used to synthesize goal-directed actions. As a test example we use the task of throwing a ball into a basket, which has the advantage that its physics is well understood and we can thus compare our results with an ideal system.

II. Collecting the Example Movement Library

As mentioned in the introduction, motion capture has been used successfully to generate fairly complex movements on a humanoid robot. However, direct reproduction of movements, even if it includes the physical constraints of a robot, rarely results in a successful execution of the task that involves external goals. In the throwing example of Fig. 1, the direct reproduction ended up in a throw that missed the basket (middle row of figures). Moreover, the execution of the throwing movement was suboptimal in many other ways such as for example timing of the ball release and smoothness. It was therefore necessary to develop a methodology to adapt the initial robot motion. In our previous work, we explored the coaching paradigm to solve these problems. Coaching provides a familiar setting to most people for interacting with and directing the behavior of a complex humanoid robot where human-robot communication takes the form of coach's demonstrations and high-level qualitative instructions. In this way it is possible to generate throwing movements that result in successful throws with good dynamical properties, which are suitable for generalization. See [12] for more details.

There are other ways than coaching to adapt captured movements to attain the goal of the task in a given situation. The via-point representation based on the forward-inverse relaxation neural network model (FIRM) [20] is one of them. Via points are extracted sequentially by taking the first two via-points to be the end-points of the movement and interpolating the movement using the minimum principle for the approximated dynamics model (point mass), which results in a minimum jerk trajectory (fifth order polynomial). New via-points are determined by calculating the distance between the observed and interpolated trajectory and adding the via-points at the point of the maximum squared error until the error is small enough. Hovewer, the movement generated by the final set of via-points still cannot ensure the successful execution of the task. It was therefore proposed to adapt the trajectory by moving the via-points until the robot is successful [9]. This is

accomplished by constructing a function from via-points to the task goal and by moving the via-points using a Newton-like optimization method.

In certain situations, it is well possible that a skilled engineer would be able to design optimal trajectories for some situations. The coaching paradigm described above just provides the technology that enables non-skilled people to design "good" movements for learning. Thus, all these methods for trajectories generation can be utilized for the construction of a library of movements. The method we propose in the following is independent of the data collection method[1].

III. Generalization across Movements

The data collection mechanisms described in the previous section provide us with a set of movements $\boldsymbol{M}_i,\ i = 1,\ldots,NumEx$, that were executed by the robot and succeeded to accomplish the goal of the task in the observed situations. We denote the goals by $\boldsymbol{q}_i \in \mathbb{R}^m,\ i = 1,\ldots,NumEx$. In the case of throwing a ball into a basket, the goals $\{\boldsymbol{q}_i\}$ are specified by the positions of the basket. Every movement $\boldsymbol{M}_i$ is encoded by a sequence of trajectory points $\boldsymbol{p}_{ij}$ at times $t_{ij},\ j = 1,\ldots,n_i$. We have experimented both with end-effector trajectories (in this case $\boldsymbol{p}_{ij}$ are points in the Cartesian space) and with robot joint trajectories (in this case $\boldsymbol{p}_{ij}$ are the joint angles stemming from the active degrees of freedom). Our aim is to develop a method that can compute motions that attain the goal of the task for any given query point (goal) $\boldsymbol{q}$.

To find a representation for the desired movements, we follow [9], [20] and represent the trajectories by fifth order splines. Due to their local support property, we chose B-splines [4] to implement the spline functions, which results in the following representation

$$\boldsymbol{M}(t) = \sum_k \boldsymbol{b}_k \mathrm{B}_k(t), \tag{1}$$

where B_k are the basis functions from the selected B-spline basis.

A. Determination of Basis B-Spline Functions

We adapted the via-point approach of [9] to find a good spline basis. Unlike [9], which deals with only one example movement, we need to consider multiple examples. We therefore introduce what we call *common knot points*. Common knot points are extracted sequentially as follows:

1) First all trajectories are time-scaled to interval $[0, 1]$. The duration of every movement T_i is also stored with each example. Without re-timing it is not possible to interpolate between the examples. See Section III-C for more details on this issue. The initial knot sequence for the fifth order spline is taken to be $\mathrm{K}_1 = \{0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1\}$, which results in a so called clamped spline. Clamped splines can be used to calculate minimum jerk splines interpolating the desired position, velocity, and acceleration at the end points (0 and 1). The initial spline basis consists of six basis functions.
2) For every movement $\boldsymbol{M}_i$ we determine the best approximating fifth order spline $\boldsymbol{S}_{li}$ with basis functions defined by the current knot sequence K_l.
3) For all configurations $(\boldsymbol{p}_{ij}, s_{ij}),\ s_{ij} = t_{ij}/T_i$, we calculate the distance to the generated spline trajectories
$$e_{lij} = \|\boldsymbol{p}_{ij} - \boldsymbol{S}_{li}(s_{ij})\|. \tag{2}$$
We select the knot point to be added to the existing knot sequence at the point of the maximum squared error e_{lij} between the example movements and the generated spline trajectories. The new knot sequence is given by
$$K_{l+1} = \{0, \ldots, s_{ij}, \ldots, 1\}. \tag{3}$$
4) The procedure continues at step 2 (with $l \leftarrow l+1$) until the difference between the example movements and the generated spline trajectories becomes sufficiently small.

The above process is similar to the way Miyamoto et al. [9] determine via-points. Its final result, the knot sequence K_L, is applied to define a spline that we use to synthesize goal-directed actions.

B. Synthesizing New Actions

Given a goal $\mathbf{q}$, we would like to find movement $\mathbf{M}(\mathbf{q})$ that can attain this goal. Using the above representation we can write

$$\boldsymbol{M}(\mathbf{q}) = \sum_{k=1}^{N} \mathbf{b}_k(\mathbf{q})\mathrm{B}_k, \tag{4}$$

where N is the number of B-spline basis functions defined by the knot sequence K_L. In computer graphics, new movements are often synthesized by simply interpolating the splines approximating the example movements [15]

$$\boldsymbol{M} = \sum_i w_i \boldsymbol{M}_i. \tag{5}$$

However, if the approximation by splines is not accurate, such an approach can introduce undesired deformations in the example movements, which can affect the synthesized actions. We therefore studied other techniques such as *locally weighted regression* [2] to generate movements for any given goal. Our main motivation is that it is difficult to find global models that are valid everywhere and that it is therefore better to look for local models that are correct only in one particular situation, but are easier to compute. In locally weighted regression, local models are fit to nearby data. Its application results in the following optimization problem

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \arg\min_{\boldsymbol{b}}\{\mathrm{C}(\mathbf{q})\}, \\ \mathrm{C}(\mathbf{q}) &= \sum_{i=1}^{NumEx} \sum_{j=1}^{n_i} \left\| \sum_{k=1}^{N} \mathbf{b}_k \mathrm{B}_k(s_{ij}) - \boldsymbol{p}_{ij} \right\|^2 \mathrm{W}\left(\mathrm{d}_i(\mathbf{q}, \mathbf{q}_i)\right). \end{aligned} \tag{6}$$

Here W is the weighting kernel function and d_i are the distance functions between the query point and the data points

[1] However, a via-point like method is used to obtain a suitable representation for goal-directed actions.

$\mathbf{q}_i$. The unknown parameters we minimize over are $\mathbf{b} = [\boldsymbol{b}_1^T, \ldots, \boldsymbol{b}_N^T]^T$.

Since $\mathrm{W}\left(\mathrm{d}_i(\mathbf{q}, \mathbf{q}_i)\right)$ does not depend on the B-spline coefficients $\boldsymbol{b}_k$, the optimization problem (6) is a classic linear least squares problem. It is, however, very large because it contains all data points $\boldsymbol{p}_{ij}$ describing the example movements. Before describing how to solve it, we define distance functions d_i and the kernel function W. We take the weighted Euclidean distance for d_i, i. e.

$$\mathrm{d}_i(\mathbf{q}, \mathbf{q}_i) = \frac{1}{a_i}\|\mathbf{q} - \mathbf{q}_i\|, \ a_i > 0. \tag{7}$$

It is best to select a_i so that there is some overlap between the neighboring query points. One possibility is

$$a_i = 2 \min_j \|\mathbf{q}_i - \mathbf{q}_j\| \tag{8}$$

By selecting a_i in this way we ensure that as query points transition from one data point to another, the generated movements also transition between example movements associated with the data points.

There are many possibilities to define the weighting function W [2]. We chose the tricube kernel

$$\mathrm{W}(d) = \begin{cases} (1 - |d|^3)^3 & \text{if} |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

This kernel has finite extent and continuous first and second derivative. Combined with distance (7), these two functions determine how much influence each of the example movements $\boldsymbol{M}_i$ has. It is easy to see that the influence of each $\boldsymbol{M}_i$ diminishes with the distance of the query point $\boldsymbol{q}$ from the data point $\boldsymbol{q}_i$. If the data points $\boldsymbol{q}_i$ are distributed uniformly along the coordinate axes, then every new goal directed movement $\boldsymbol{M}(\boldsymbol{q}), \boldsymbol{q} \neq \boldsymbol{q}_i$, will be influenced by 4^m example movements[2], where m is the dimension of the query point.

Optimization of criterion (6) is a linear least-squares problem. Locally weighted regression combined with the local support of B-spline basis functions make the resulting linear system that needs to be resolved sparse. Additionally, since only the weights and not the basis functions depend on the query point $\boldsymbol{q}$, the sparse system matrix can be precomputed in its entirety. We applied the Matlab implementation of sparse matrix algebra to solve the resulting linear problems, which enabled us to generate new actions quickly despite the large number of trajectory points $\boldsymbol{p}_{ij}$. Another advantage of the proposed method is that there is no need to search for nearby movments in the database; locally weighted regression and sparse matrix algebra do this job.

C. Re-Timing of the Generated Actions

To interpolate between example movements, we needed to first scale the timing of all trajectories to a common interval, which we chose to be $[0, 1]$. This scaling, however, causes the velocities and accelerations of both the example movements and the synthesized actions to be scaled. To synthesize movements with proper velocities and accelerations – which is essential to solve dynamic tasks – we need to rescale the resulting actions back to the original time interval. As described in Section III-A, the timing of each example motion $\boldsymbol{M}_i$ is scaled by $1/T_i$, where T_i is the duration of the example movement. Hence to re-time the synthesized action, we need to compute an estimate for the expected time duration T.

For this purpose, we approximate the expected timing by a multivariate B-spline function $\mathrm{f}_t : \mathbb{R}^m \rightarrow \mathbb{R}$, which is estimated by minimizing the following criterion

$$\sum_{i=1}^{NumEx} \left(\mathrm{f}_t(\boldsymbol{q}_i) - T_i\right)^2. \tag{10}$$

In our experiments we defined a B-spline basis by uniformly subdiving the domain of the goal points $\boldsymbol{q}_i$. A suitable timing for the synthesized action is then given by

$$T = \mathrm{f}_t(\boldsymbol{q}) = \sum_{i=1}^{M} a_i B_i(\boldsymbol{q}). \tag{11}$$

Finally, the correctly timed trajectories for the synthesized actions obtained by minimizing criterion (6) can be calculated by mapping the knot points $K_L = s_i$ to the new knot sequence K'_L

$$K'_L = \{0, \ldots, T * s_i, \ldots, T\}. \tag{12}$$

The optimal coefficients $\boldsymbol{b}_k(\boldsymbol{q})$ remain unchanged and the spline with these coefficients defined on the knot sequence K'_L specifies an action with appropriate velocities and accelerations.

It should be noted here that uniform scaling might not be suitable for every task. In some situations it might be more appropriate to segment the example movements and apply different scaling factors to different time intervals. Here matching of key events is crucial for good results [15]. Computer graphics community has proposed some approaches to automatically resolve this problem [7], [13]. Since the task considered in this paper does not require nonuniform scaling, we did not attempt to develop more complex re-timing methods here.

IV. Experimental Results

We validated our approach both in simulation and on the real robot. As a test example we considered the task of throwing a ball into a basket, which has the advantage that it is a dynamic task, dependent not only on the positional part of the movement, and that its physics is well understood. This allows us to compare our results with an ideal system. It can easily be shown that the trajectory of the ball after the release is fully specified by the position and velocity at the release point

$$x = x_0 + v_0 t \cos(\alpha), \ y = y_0 + v_0 t \sin(\alpha) - \frac{gt^2}{2}, \tag{13}$$

where (x_0, y_0) is the release point, v_0 is the linear velocity of the ball at release time and α is the initial angle of the throw. We considered the problem where the target basket is placed

[2]Exception are the movements at the edge of the training space.

in xy-plane. Note that a humanoid robot could normally turn towards the basket, thus solving this problem allows the robot to throw the ball to any position in space.

A. Simulation Results

For the interpolation to work, the style of example movements must be similar. Interpolation between movements that have nothing in common would not results in sensible actions. To generate examples that can be used for action synthesis, we used Eq. (13) to design Cartesian space trajectories that theoretically result in successful throws for a given basket position. The base of the robot, which was taken to be a seven degrees of freedom arm, was fixed in space. The designed trajectories consisted of circular and linear parts. From a given basket position, we determined a suitable release point and by specifying the desired angle under which the ball should fall into a basket (taken to be 60 degrees), a good trajectory for each situation could be calculated. We distributed the goal basket positions within a rectangular area of size 4×2 meter squares, with the lower left corner positioned at $(1.2, 0.1)$ meters. The base of the robot was placed at $(-0.5, 0.1)$ meters. Fig. 2 shows the velocity profiles of the movements generated by specifying a grid in thin rectangular area with baskets placed every 0.5 meters (altogether 45 basket positions). We used inverse kinematics to generate example trajectories in joint space.

By specifying different grid sizes for training (we took grid side lenghts of 0.25, 0.5, and 1 meter, which resulted in 15, 45, and 153 example movements within the training area), we tested how many example movements are necessary to throw a ball anywhere within the training area with good precision. Tables I and II show the errors in the synthesized throws. They were calculated by using Eq. (13) to determine the ball flight trajectory after release. All values in the tables are given in centimeters. The density of the training data is specified by the grid size (rightmost column). Since the error was smaller away from the edges of the library (see Fig. 3), we estimated

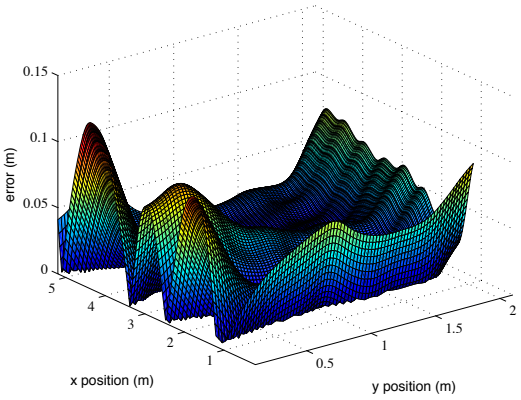

Fig. 3. The throw error. The graph corresponds to the condition of Tab. I with grid size 50×50, joint space synthesis. The error is larger at the edges of the training area where less data is available for synthesis.

the error in the complete training area and in the area reduced by the side length of the grid along the edges. In Tab. I the data points $\boldsymbol{p}_{ij}$ used in (6) consisted of both positions and velocities[3], which were approximated by the spline functions. In Tab. II the data points $\boldsymbol{p}_{ij}$ consisted of positional information only. To test the method we evaluated the throws by applying a grid of 2.5×2.5 centimeter squares, which resulted in 13041 test throws for every training condition.

Both tables show that the accuracy of the ball throw is significantly improved when more data is available. We achieved average precision between 1 and 2.5 cm for the two finer grids. Hence, 45 training examples were enough for an average precision of below 2 cm within the reduced area. The comparison of Tab. I and II also shows that the explicit addition of velocity information did not improve the throwing precision. We believe that the main reason for this is that our data was simulated at a typical robot servo rate of 500 Hz,

[3]Formula (6) is valid for positional information only, but extension to velocities and accelerations is straightforward and does not significantly change the linear system that needs to be resolved.

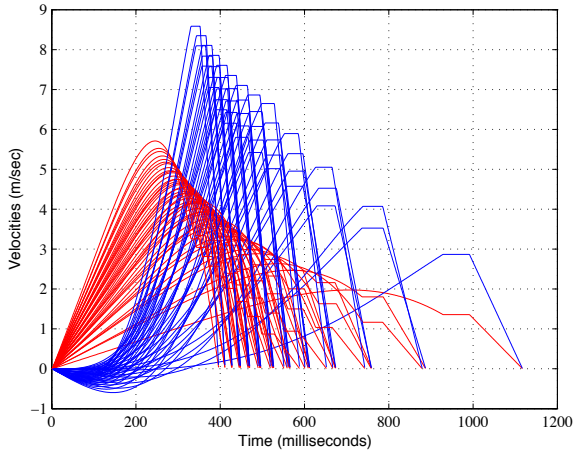

Fig. 2. Cartesian velocities of example movements

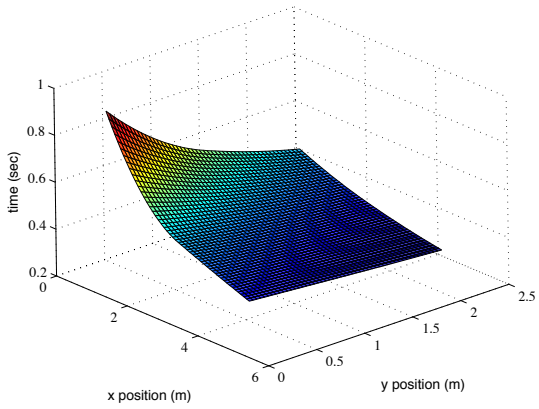

Fig. 4. Spline function approximating the release times of the movements with respect to the basket position

TABLE I

ERRORS IN THE SYNTHESIZED THROWS (IN CENTIMETERS). SEE TEXT FOR THE EXPLANATION.

	Joint space		Cartesian space		Grid size
Training area	Full	Reduced	Full	Reduced	
Average error	2.18	1.52	1.70	1.28	25×25
Max. error	10.39	5.79	9.63	4.67	25×25
Average error	2.72	1.75	2.25	1.40	50×50
Max. error	12.57	7.08	13.79	6.01	50×50
Average error	10.15	7.03	9.85	6.37	100×100
Max. error	38.97	15.27	37.71	13.23	100×100

TABLE II

ERRORS IN THE SYNTHESIZED THROWS WITHOUT INCLUDING VELOCITIES IN THE DATA (IN CENTIMETERS). SEE TEXT FOR THE EXPLANATION.

	Joint space		Cartesian space		Grid size
Training area	Full	Reduced	Full	Reduced	
Average error	2.25	1.50	2.25	1.60	25×25
Max. error	10.03	4.89	9.69	4.58	25×25
Average error	2.41	1.54	2.43	1.61	50×50
Max. error	13.35	6.17	13.77	5.91	50×50
Average error	10.39	6.55	10.23	6.40	100×100
Max. error	38.31	13.34	37.78	12.94	100×100

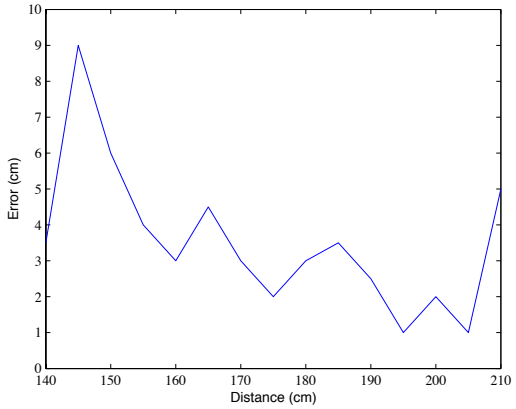

Fig. 5. Accuracy of the learned throwing action executed by the robot

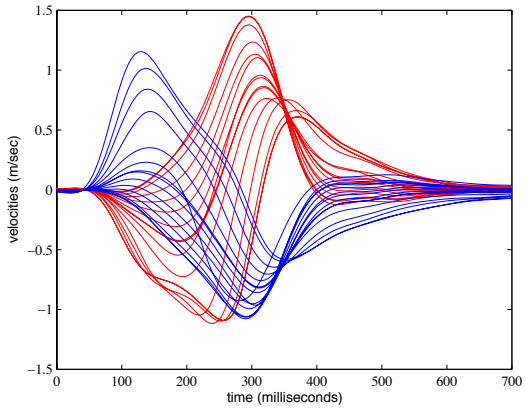

Fig. 6. Cartesian velocities of generated robot movements for throws into a basket positioned at 1.4, 1.45, 1.5, ... 2.1 meters

hence enough data was available to estimate the velocities from positional information. For sparser data the addition of velocity and acceleration will become more important.

We applied the proposed approach to the data collected in both the Cartesian and the joint space. Tab. I and II show that in most but not all cases the precision was slightly better when using the Cartesian space data. However, the differences were so small that we consider both types of data equally suitable.

The improvement with denser training data was much more significant when moving from the grid size of 1×1 to 0.5×0.5 meter squares than when moving to the grid size of 0.25×0.25. The main reason was that the estimation of the timing function f_t of Section III-C (see Fig. 4) used the same set of basis functions to form the approximating spline in all cases. Thus when the grid size was reduced, the inaccuracies in the timing function started to dominate and the throwing precision did not improve any further. This shows the importance of the proper estimation of timing.

Our results demonstrate that albeit the system was not provided with the model of the task, it managed to learn how to throw the ball with high precision using no other information but the example movements and the associated basket positions.

B. Robot Experiments

We used a humanoid robot arm with seven degrees of freedom for our first real-world action synthesis experiments. We used five training examples (taken at 1.37, 1.63, 1.77, 1.98, and 2.18 meters) to train the throwing behavior along the line from 1.4 to 2.1 meters. Fig. 5 shows the accuracy of the synthesized throws. The average error was 3.36 centimeters. The training had to be done in the joint space because the robot can not follow Cartesian space trajectories with sufficient accuracy. Also, it is important to use the desired joint trajectories and not the actual joint trajectories for training, so that the synthesized actions directly relate to the actual robot commands. Our results show that locally weighted regression provides us with the ability to synthesize goal-directed actions directly from the data instead of first approximating the example movements by spline functions and then interpolating the coefficients of the approximating splines,

Fig. 6 depicts the velocities of robot hand movements in $xy-$plane of the Cartesian space. These velocities are different from the velocities of example simulated movements in Fig. 2 because we used different types of throws in these two

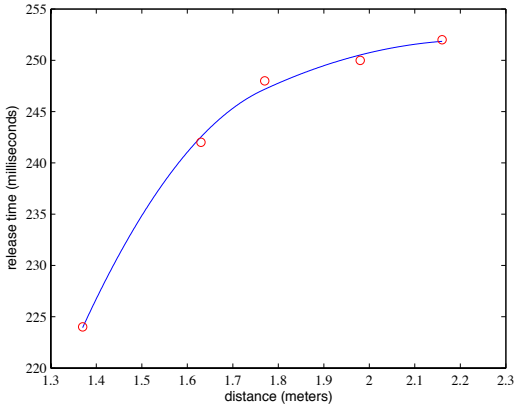

Fig. 7. Spline function approximating the release times (blue) and release times of the example movements (red)

examples. Nevertheless, both figures show a typical smooth transition between movements as the target position moves in space. Finally, Fig. 7 shows the spline approximating the release point timings. Again, the form of the spline is somewhat different from the simulated spline of Fig. 4, but both splines exhibits smooth transition of release times as the basket position changes.

V. Conclusion

The most important result of this paper is that dynamic goal-directed actions can be synthesized by applying locally weighted regression to the library of example movements, where each of the example movements is known to fulfil the task in one particular situation. We showed how to connect action synthesis with techniques such as coaching and programming by demonstration, which enables us to acquire the example library. Our experiments demonstrate that we can achieve fairly accurate results without providing the system with models about the dynamics of the task and without needing to acquire an excessive amount of example movements. Finally, we demonstrated that locally weighted regression is suitable for synthesizing goal-directed actions directly from the training data instead of first approximating the example movements by spline functions and then interpolating the approximating splines.

Our approach is by no means limited to ball throwing. It is pretty straightforward to apply it to other discrete movements such as reaching, catching, tennis strokes, etc. More work is necessary to generalize the approach to rhythmic movements. We believe, however, that such a generalization is possible by utilizing closed splines instead of the clamped splines, which we used to synthesize discrete movements in this paper.

Acknowledgment: The work described in this paper was partially conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

[1] 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.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, 1997.

[3] 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.

[4] C. de Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[5] 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.

[6] 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.

[7] L. Kovar and M. Gleicher, "Flexible automatic motion blending with registration curves," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2003, pp. 214–224.

[8] S. Kudoh, T. Komura, and K. Ikeuchi, "Stepping motion for a human-like character to maintain balance against large perturbations," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, Florida, 2006, pp. 2561–2567.

[9] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakanao, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[10] N. S. Pollard, J. K. Hodgins, M. Riley, and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, Washington, DC, May 2002, pp. 1390–1397.

[11] M. Riley, A. Ude, and C. G. Atkeson, "Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching," in *Proc. 2000 Workshop on Interactive Robotics and Entertainment*, Pittsburgh, Pennsylvania, April/May 2000, pp. 35–42.

[12] M. Riley, A. Ude, C. G. Atkeson, and G. Cheng, "Coaching: An approach to efficiently and intuitively create humanoid robot behaviors," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Genoa, Italy, December 2006, pp. 567–574.

[13] C. Rose, B. Bodenheimer, and M. F. Cohen, "Verbs and adverbs: Multidimensional motion interpolation using radial basis functions," *Computer Graphics, Proc. SIGGRAPH '96*, pp. 147–154, August 1998.

[14] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Humanoid robot motion generation with sequential physical constraints," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, Florida, 2006, pp. 2649–2654.

[15] 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.

[16] ——, "Construction and optimal search of interpolated motion graphs," in *ACM Transactions on Graphics*, 2007.

[17] 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.

[18] A. Ude, C. G. Atkeson, and M. Riley, "Planning of joint trajectories for humanoid robots using B-spline wavelets," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, California, April 2000, pp. 2223–2228.

[19] ——, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, 2004.

[20] Y. Wada and M. Kawato, "A neural network model for arm trajectory formation using forward and inverse dynamics models," *Neural Networks*, vol. 6, no. 7, pp. 919–932, 1993.

[21] D. J. Wiley and J. K. Hahn, "Interpolation synthesis of articulated figure motion," *IEEE Computer Graphics and Applications*, vol. 17, no. 6, pp. 39–45, 1997.

[22] K. Yamane and Y. Nakamura, "Dynamic filter – Concept and implementation of online motion generator for human figures," *IEEE Trans. Robotics Automat.*, vol. 19, no. 3, pp. 421–432, 2003.

AALBORG UNIVERSITY

Copenhagen

Computer Vision and Machine Intelligence Lab (CVMI)

Parametric Hidden Markov Models for Recognition and Synthesis of Movements

Dennis Herzog, Volker Krüger, Daniel Grest

Advances in Computer Vision and Machine Intelligence CVMI 2008:1

ISSN 1902-2034

Dennis Herzog, Volker Krüger, Daniel Grest
Parametric Hidden Markov Models for Recognition and Synthesis of Movements

Report number: CVMI 2008:1
ISSN 1902-2034
Publication date: January 2008
E-mail of author: vok@cvmi.aau.dk

Reports can be ordered from:

Computer Vision and Machine Intelligence Lab
Aalborg University Copenhagen (AAU)
DK-2970 Ballerup
DENMARK

telefax: +45 96 35 24 80
http://www.cvmi.aau.dk/

Parametric Hidden Markov Models for Recognition and Synthesis of Movements

Dennis Herzog, Volker Krger, Daniel Grest
Computer Vision and Machine Intelligence Lab
Aalborg University Copenhagen

January 7, 2008

Abstract

A common problem in human movement recognition is the recognition of movements of a particular type (semantic). E.g., grasping movements have a particular semantic (grasping) but the actual movements usually have very different appearances due to, e.g., different grasping directions. In this paper, we develop an exemplar-based parametric hidden Markov model (PHMM) that allows to represent, e.g., movements of a particular type and that compensates for the different appearances and parameterizations of that movement. The PHMM is based on exemplar movements that have to be ”demonstrated” to the system. Recognition and synthesis are carried out through locally linear interpolation of the exemplar movements. For a meaningful interpolation, the exemplars have to be in sync, what exhibits certain problems that are resolved in this paper. In our experiments we combine our PHMM approach with our 3D body tracker. Experiments are performed with pointing and grasping movements. Synthesis for grasping is parameterized by the positions of the objects to be grasped. In case of recognition, our approach is able to recover the position of an object at which a human volunteer is pointing. Our experiments show the flexibility of the PHMMs in terms of the amount of training data and its robustness in terms of noisy observation data. In addition, we compare our PHMM to an other kind of PHMM, which has been introduced by Wilson and Bobick.

Keywords: action recognition, action representation, computer vision, robotics, AI

1 Introduction

One of the major problems in action and movement[1] recognition is to recognize actions that are of the same type but can have very different appearances depending on the situation they appear in. In addition, for some actions these differences are of major importance in order to convey their meaning. Consider for example the movement of a human pointing at an object, “This object there...”, with the finger pointing at a particular object (like in Fig. 1). Clearly, for such an action, the action itself needs to be recognized but also the spot in 3D space at which the human is pointing. Only together do these two pieces of information convey the full semantics of the movement. Another common problem is the synthesis of action: This concerns two major problem areas: In robotics, one is interested in teaching robots through simple demonstrations (imitation learning) [1, 2, 10]. In 3D human body tracking, one is interested in using motion models in order to constrain the parameter space (e.g. [8] for simple cyclic motions). In both cases, one is interested in teaching the system in an easy and efficient manner a particular movement so that afterwards, the system is able to synthesize movements of the same type, however, with a different parameterization. Here, we consider grasping movements as an example where a human is reaching out for an object to grasp it[2]. One may perform as demonstration a set of grasping movements. All grasping

[1] We use the terms *action* and *movement* interchangingly. Actions usually denote movements that involve objects.

[2] The precise choice of a hand grasp depends on the type of object, from where it is being grasped, etc. In our discussion, we omit the issue of the different hand grasps and focus only on the arm movements.

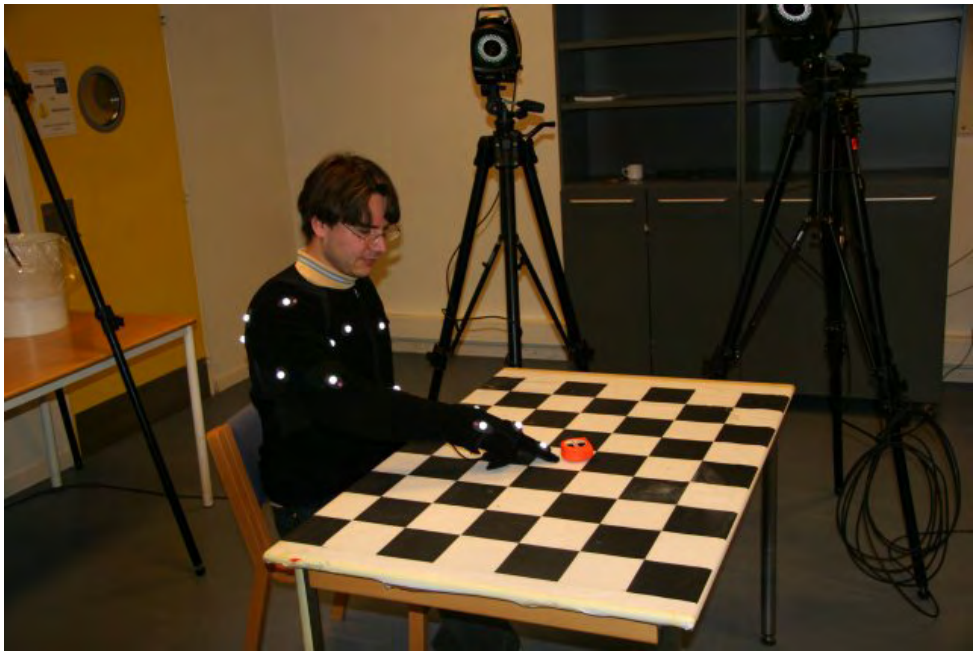

Figure 1: The image shows the setup of the capturing session for our dataset. The person is currently pointing at a raster position at the table-top.

movements depend on the location of the object to be grasped. In case of a humanoid robot, the synthesis should then allow the robot to perform the learned grasping movements with new parameterizations, e.g., grasping objects at different positions. In case of the 3D body tracking, synthesis would allow a better prediction of the next pose and even allows an estimate of parametric actions instead of the full joint configuration which would result into a considerable reduction in search space.

Most current approaches model movements with a set of movement *prototypes*, and identify a movement by identifying the prototype which explains the observed movement best. This approach, however, has its limits concerning efficiency when the space of possible parameterizations is large.

A pioneering work in this context was done by Wilson and Bobick [12]. Wilson and Bobick presented a parametric HMM approach that is able to learn an HMM based on a set of demonstrations. Their training and recognition approach is based on the EM algorithm, where the parameters of the movements are taken as latent variables. For recognition, they recover the parameters that explains best the observation.

In this paper, we develop a different parametric hidden Markov model approach. Contrary to Wilson and Bobick, our aim is recognition as well as synthesis. Also, we would like to provide a simpler and more efficient training strategy by being able to simply provide exemplars based on which the generation of novel HMMs can be done.

A further contribution is a novel method for time warping of HMM training data that is not limited to pairwise warps like the classical time warping approaches.

In the following section, we give a short overview of the related work. In Sect. 3 we provide some basics to introduce our exemplar-based parametric HMM in Sec. 4. Extensive experimental results including a comparison with [12] are presented in Sect. 5. Conclusions in Sect. 6 complete our paper.

2 Related Work

Most approaches for movement representation that are of interest in our problem context are trajectory based: Training trajectories, e.g., sequences of human body poses, are encoded in a suitable manner. Newly incoming trajectories are then compared with the previously trained ones. A recent review can be

found in [7].

Some of the most common approaches to represent movement trajectories use hidden Markov models (HMMs) [4, 9]. HMMs offer a statistical framework for representing and recognition of movements. One major advantage of HMMs is their ability to compensate for some uncertainty in time. However, due to their nature, HMMs are only able to model specific movement trajectories, but they are not able to generalize over a class of movements that vary accordingly to a specific set of parameters.

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 [5]. In order to deal with a large parameter space one ends up, however, with a lot of experts and a large amount of training becomes necessary.

Another extension of the classical HMMs into parametric HMMs was presented in [12], as mentioned above. A more recent approach was presented by [1]. In this work, the interpolation is carried out in spline space where the trajectory of the end-effector is modeled. Apart from the fact that the authors have not yet performed an evaluation of their system, their approach does not seem suitable for controlling the entire arm movements for movement synthesis and recognition.

In addition to HMMs, there are also other movement representations that are interesting in our context, e.g., [6, 11]. However, these approaches share the same problems as the HMM based approaches.

3 Preliminaries of HMMs

A hidden Markov model is a probabilistic finite state machine extended in a probabilistic manner, that is defined as a triple $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$, where the transition matrix $\boldsymbol{A} = (a_{ij})$ defines the transition probability between the hidden states $i, j = 1, \ldots, N$, and $\boldsymbol{B}$ defines the output distributions $b_i(\boldsymbol{x}) = \mathrm{P}(\boldsymbol{x}|q_t = i)$ of the states. The vector $\boldsymbol{\pi}$ defines the probabilities of each state of being the initial state of a hidden state sequence.

In our approach a restrictive type of continuous left-right HMMs (as in [3]) is used, whose output probability distributions of each state i are modeled by single Gaussian distribution $b_i(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, and whose state transitions are self-transitions or are transitions leading to the successor state, i.e., other transition probabilities are set to zero.

If such an HMM is used to model, e.g., a simple trajectory or sequence $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_t \ldots \boldsymbol{x}_T$, then each Gaussian $\mathcal{N}_i(\boldsymbol{x}) := b_i(\boldsymbol{x})$ would "cover" some part of the trajectory, where the state i increases as the time of the trajectory evolves. In addition, the temporal behavior of the trajectory is coded in the transition probabilities. In the case of multiple trajectories of the same kind, the Gaussian capture the variance of these trajectories, but in addition, such a model can compensate for different progression rates between the trajectories. As we want to facilitate the synthesis of movements it is obviously necessary to use left-right HMMs. However, it is worth to be mentioned that, even in the case of such a restrictive left-right model, there is no strict assignment between states and observations $\boldsymbol{x}_t$.

For a comprehensive introduction to HMMs, we refer to [4, 9]. The most important algorithms of the HMM framework are mentioned in the following example section.

3.1 Recognition using HMMs

For recognition or classification HMMs are generally used as follows: For each specific class k of sequences an HMM $\boldsymbol{\lambda}^k$ is trained by a representative training set $\mathcal{X}^k$ for that class. The training of an HMM $\boldsymbol{\lambda}$ is done by adjusting the model parameters to values, which are maximizing the likelihood function $\mathrm{P}(\mathcal{X}|\boldsymbol{\lambda})$. For this maximization, we apply the Baum/Welch expectation maximization (EM) algorithm [9].

The classification of a specific output sequence $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_T$ is done by selecting that class k, for which the likelihood $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^k)$ is maximal. The probability of a sequence $\boldsymbol{X}$ given the model is efficiently computed by the forward/backward algorithm [9].

One obvious approach for handling whole classes of parameterized actions for the purpose of action recognition and parameter estimation is a mixture-of-experts approach [5] and to sample the parameter space by training for each sample a prototype HMM. The HMM that maximizes the likelihood—given an action sequence—identifies class membership and the parameterization of the action. However, this approach is not appropriate, because too many repetitions of the action are needed to train the prototype HMMs of all samples. Therefore, we introduce the parameterization of the movements as a new parameter of the model, which also is the basic idea of the approach in [12].

4 Parametric HMM Framework

The main idea of our approach for handling whole classes of parameterized actions is a supervised learning approach where we deduce an HMM for novel action parameters by locally linear interpolation of exemplar HMMs that were previously trained on exemplar movements with known parameters. The generation of newly parameterized HMMs can be done online or offline.

The deduction of an HMMs $\boldsymbol{\lambda}^{\phi}$ for a specific parameter is carried out by component-wise linear interpolation of the nearby exemplar models. That results, e.g., in case of a single scalar parameter u and two given exemplar models $\boldsymbol{\lambda}^0$ and $\boldsymbol{\lambda}^1$ for $u = 0, 1$, in a state-wise or Gaussian-wise deduction of the Gaussian $\mathcal{N}_i^u(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^u, \boldsymbol{\Sigma}_i^u)$ of the state i of the model $\boldsymbol{\lambda}^u$ with means and covariances, as given by

$$\begin{aligned} \boldsymbol{\mu}_i^u &= (1-u)\boldsymbol{\mu}_i^0 + u\boldsymbol{\mu}_i^0 \\ \boldsymbol{\Sigma}_i^u &= (1-u)\boldsymbol{\Sigma}_i^1 + u\boldsymbol{\Sigma}_i^1. \end{aligned} \tag{1}$$

This situation of two exemplar models $\boldsymbol{\lambda}^0$ and $\boldsymbol{\lambda}^1$ for $u = 0, 1$ is scetched in Fig. 2 for sequences of parameterization $u = 0$, and $u = 1$. Obviously, in the case of such an arrangement, the state-wise interpolation results in a good model $\boldsymbol{\lambda}^u$ for sequences, e.g., in the middle (where $u = 0.5$). But this is the case *only if* the same two states of the exemplar HMMs do model the same semantical part of the motion. Consider, e.g., the n-th state of each of the two HMMs, where one of the two states state possibly models a part of a forward motion of a hand while the other might model a part of a backward motion. Clearly, interpolation of two such states does not make sense. Therefore, we develop an alignment of the states as described in Sec. 4.2 below. The expansion to the multi-variate case of parameterization $\boldsymbol{\phi}$ is straightforward, e.g., by using bilinear ($\boldsymbol{\phi} = (u, v)$) or trilinear interpolation.

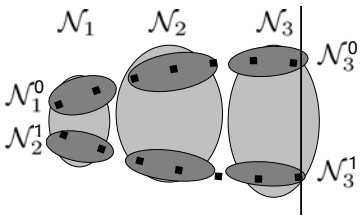

Figure 2: The upper three dark ellipsoids are depicting the Gaussians $\mathcal{N}_1^0, \ldots, \mathcal{N}_3^0$ of the states $i = 1, 2, 3$ of an HMM $\boldsymbol{\lambda}^0$ that is trained by sequences, that begin on the left and are leading to the upper part of the vertical line on the right hand side. In this case the parameterization of the sequences is $u = 0$. The dots sketch one of these training sequences. Similarly, the lower three ellipsoids of an HMM $\boldsymbol{\lambda}^1$ model sequences with a parameter $u = 1$. Additionally, the Gaussians $\mathcal{N}_i$ of a global model $\boldsymbol{\lambda}$ are indicated in light gray. In this case λ is trained with all training sequences.

4.1 Synthesis

Suppose a given grasp position $\boldsymbol{p}$ on a table. Then, synthesis can be done as follows: At first, four HMMs $\boldsymbol{\lambda}^i{}_{,i=1,\ldots,4}$ with closest associated grasp positions $\boldsymbol{p}^i$ are chosen under the constraint that at least three of the $\boldsymbol{p}^i$ are strongly not collinear and that $\boldsymbol{p}$ lies accurately in the convex hull of $\{\boldsymbol{p}^i\}$. Then, the bilinear interpolation parameters u, v are estimated such that the interpolated point $\boldsymbol{p}^{uv}$ approximates $\boldsymbol{p}$ best. Then, the model $\boldsymbol{\lambda}^{uv}$, i.e., the sequence $\boldsymbol{\mu}_1^{uv} \ldots \boldsymbol{\mu}_N^{uv}$ of the Gaussians, is calculated. Afterwards, this sequence can be expanded to a function $\boldsymbol{f}(t)$ by using spline interpolation (we use linear spline interpolation). If needed, this can be done with respect to the time durations coded in the transition probabilities.

4.2 Synchronized Setup of HMM States

As mentioned above, it is necessary to setup corresponding states of local exemplar HMMs in such a way, that the corresponding states model the same semantical parts of the movements. This task is somehow similar to dynamic time warping. The time warping algorithms synchronize sequences to compensate for different dynamics. But these algorithms are not suitable for our task. On the one hand, these algorithms synchronize sequences only pairwise. On the other hand, the alignment of sequences do not overcome the task of setting up the exemplar HMMs. Here, it is worth to mention, that we have successfully used HMMs for time warping—in a not pairwise way—of several sequences, which do vary, considerably.

Here, the underlying idea is to set up local exemplar HMMs $\boldsymbol{\lambda}^\phi$ by using the ability of HMMs to compensate to some extend temporal variations. We precede in two steps: In the first step a global HMM $\boldsymbol{\lambda}$ is trained based on the whole training set $\mathcal{X}$ that contains movements of different parameterizations ϕ, but of the same type. Such a global HMM is sketched in Fig. 2 with the light gray ellipsoids/Gaussians. The situation that movements of different parameterizations are covered in such a symmetrical way as in Fig. 2 can be enforced, in some way, by enforcing the hidden state sequences to pass the states always in the same sequential order from state 1 to state N. This is caused by the choice of the type of left-right model, and by allowing only sequences that start in the first and end in the last state.

In the second step, consider the reduced training set $\boldsymbol{\lambda}^\phi$ of a specific parameterization ϕ. On this training set we train an exemplar HMM $\boldsymbol{\lambda}^\phi$ while using the parameters of the global HMM $\boldsymbol{\lambda}$ as initial values. In the terminology of the EM algorithm, the exemplar model $\boldsymbol{\lambda}^\phi$ for $\mathcal{X}^\phi$ is computed using $\boldsymbol{\lambda}$ as an initial configuration and by fixing the means of the Gaussians after the first EM iteration. It is worth to note, that this gives the wanted result: In the first E step of EM 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 given the global model are computed for each sequence $\boldsymbol{X}^k = \boldsymbol{x}_1 \ldots x_T$ of the training set $\mathcal{X}^\phi$. Thus, $\gamma_t^k(i)$ defines the somehow the "responsibility" of state i for generating $\boldsymbol{x}_t^k$. In the M step the mean $\boldsymbol{\mu}_i$ of the Gaussian of state i is re-estimated as an $\gamma_t^k(i)$-weighted mean:

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

If one considers the case of Fig. 2 and the depicted upper sequence $\boldsymbol{x}_1 \boldsymbol{x}_2 \ldots \boldsymbol{x}_7$ the responsibilities $\gamma_t(i)$ would be large for $t = 1, 2$ and $i = 1$ but small for $i > 1$ (and $t = 1, 2$) caused by the position of the Gaussian of state $i = 1$. This way $\boldsymbol{\mu}_1$, as calculated by Eq. (2), lies between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, as required. One issue gives raise to problems concerning the setup of our PHMM. The Gaussians of the global HMM that is used for the alignment of the exemplar HMMs should cover the movements of exemplar movements of different parameterization in a symmetrical way as shown in Fig. 3, and described in Sec. 4.2. However, sometimes the global HMM takes a form, where some Gaussians model *only* movements of certain parameterizations—similar to the Gaussians on the right of Fig. 3. This is not surprising if one consider the ability of HMM to compensate for temporal variations, even in our restrictive left-right model. Such an HMM can be a good model for a sequence, even though one state does not fit for the sequence,

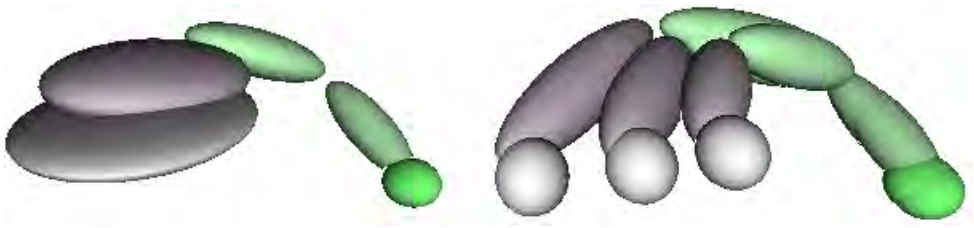

Figure 3: Synchronized setup of HMMs. Left: Some Gaussians of a global HMM are depicted on the left, which model index finger trajectories leading from the right (green ball) to the left, where the disc like ellipsoid of a Gaussian models finger positions for all pointed at positions on a table. This global HMM is used to setup the local exemplar HMMs for specific positions in a synchronized way (right).

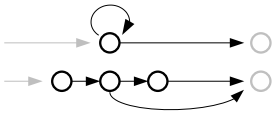

Figure 4: Time Durations of States. The upper state of a left-right HMM is replaced by the lower three pseudo states, so that the state duration lays between 2 and 3.

because, a hidden state sequence can pass a state that doesn't fit in one step and can stay for several time steps in suitable states. We addressed that problem by adding explicit time durations to the states of the HMM. For simplicity we replaced each state of the left-right HMM by some pseudo states which share one Gaussian (compare Fig. 4). This forces the hidden states sequences to stay in a state, e.g., as in Fig. 4, for at least two and for maximal three time steps.

4.3 Recognition and Parameters

In this section, we describe the recognition of the type and the parameterization of the recognized type of a parameterized movement. This is straight forward compared to the nonparametric case of classification. Consider a given sequence $\boldsymbol{X}$. We precede in two steps: First, for each possible movement type k the most likely parameter $\boldsymbol{\phi}_k$ of the corresponding parameterized HMM $\boldsymbol{\lambda}_k^{\boldsymbol{\phi}}$ is estimated. We maximize $f_k(\boldsymbol{\phi}) = P(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\boldsymbol{\phi}})$ under the constraint of senseful values (e.g., $\boldsymbol{\phi} \in [0-\varepsilon, 1+\varepsilon]^d$) by using gradient descent. The next step is the recognition of the action type. Now, the classification is reduced to the classical way by choosing the most likely model $\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_k^{\boldsymbol{\phi}_k}$. Furthermore, the parameter $\boldsymbol{\phi}^k$ of the most likely action gives us the parameterization of the recognized movement, e.g., the pointed at position $\boldsymbol{p}^{uv}$ in the table-top scenario, which is given by the bilinear interpolation parameters $(u, v) = \boldsymbol{\phi}^k$.

In our table-top experiments there are up to nine exemplar HMMs in the PHMM. Therefore, the estimate of the parameter $\boldsymbol{\phi}$ is done in a hierarchical way. In a first step $\boldsymbol{\phi}$ is estimated based on the PHMM given by bilinear interpolation of the outermost four exemplar positions or HMMs. In a second step $\boldsymbol{\phi}$ is refined as an estimate using four exemplar HMMs, which are nearest to the previous estimate.

5 Experiments

In our experiments we focus our considerations on pointing and grasping actions, which are in common action scenarios probably two of the most important movements. The pointing movements are movements such as "This object there..." (Fig. 1). Our grasping movements are reaching towards a particular object in order to grasp it.

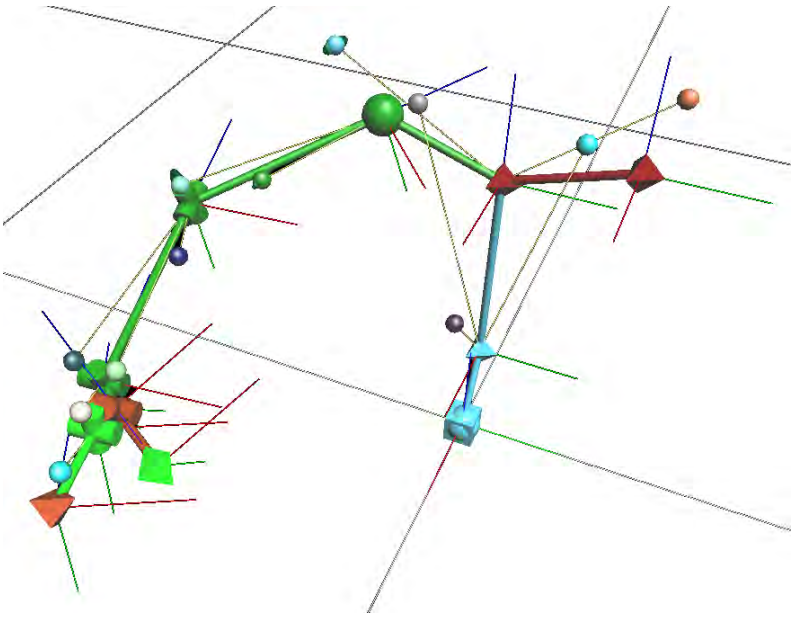

Figure 5: Capture Model of Right Arm. This model is used for motion capturing, for what the model's markers (tiny balls in picture) are aligned to captured marker positions (compare to Fig. 1).

In our systematic experiments we limit our considerations on extensive data that is acquired using a motion capture system. This way, we exclude the vision problem and are able to focus only on the representational issues for movement representation. Based on this data, we evaluate the synthesis and recognition performance of our PHMM approach. In addition, we compare the results to the results that are yielded by that type of PHMMs, which has been proposed by Wilson and Bobick [12]. However, we consider only the linear case of their model. — Concerning online recognition and synthesis, we have first results in a form of an online video, but our experiments based on visual stereo tracking data are still ongoing.

The motion capture data of our systematic experiments is acquired with an eight camera visual marker motion capture system of *Vicon*. For capturing, a model of the right arm (see Fig. 5) is aligned to visual-captured marker positions. The recognition and synthesis experiments are based on seven 3D points located at different segments of the model's body. Capturing speed is 60Hz. The seven data points are located at: sternum; shoulder, and elbow of the right arm; knuckles, index finger, and thumb of the right hand.

The setup of the capture session for acquiring takes place, as follows: The person or actor sits in front of a table (see Fig. 1). The actions are performed at a specific table-top position in such a way, that it is starting and ending in a base position (arm hanging down).

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

5.1 Training: Setup of PHMMs

The setup of the exemplar HMMs of the PHMMs for the grasping and pointing movements are done as described in Sect. 4.2. Training is done as described in Sect. 4.2. 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 corner exemplar positions of the 3×3 raster. These PHMMs, will be refered in the following as 3×3 or 2×2 PHMM of grasping or pointing. The linear PHMM developed by Wilson and Bobick is also trained

by using the training data of the 3×3 or 2×2 raster, which is refered by us as "Wilson's $n \times n$-trained PHMM".

The PHMMs are setup as follows: The training sequences are rescaled to 100 samples. The PHMMs have 20 states, where the hidden state sequences are forced to stay between 4 and 6 steps in each state.

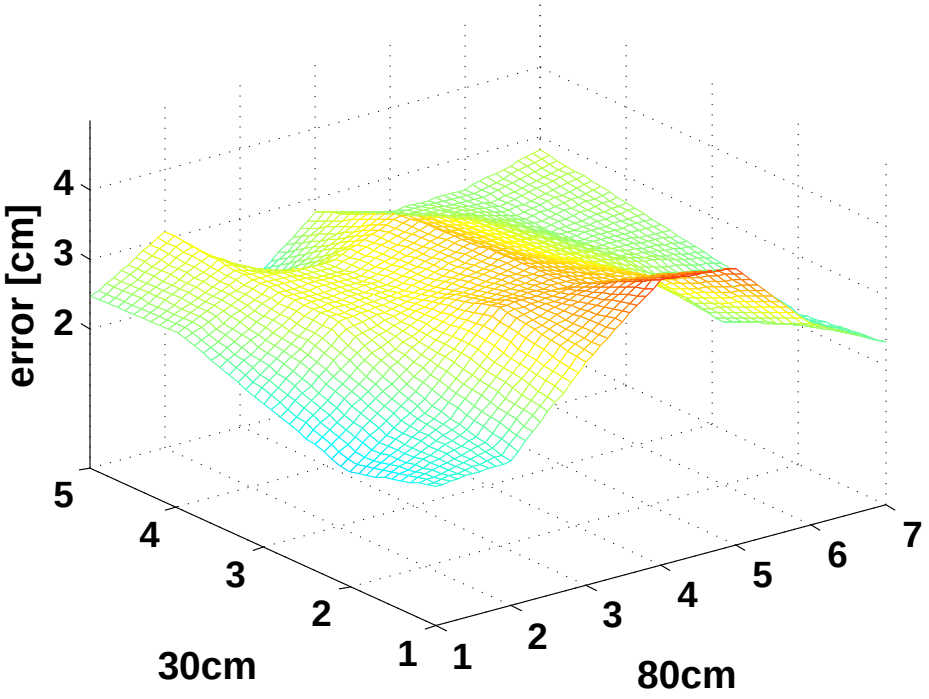

Figure 6: Synthesis Error of Pointing for 2×2 PHMM.

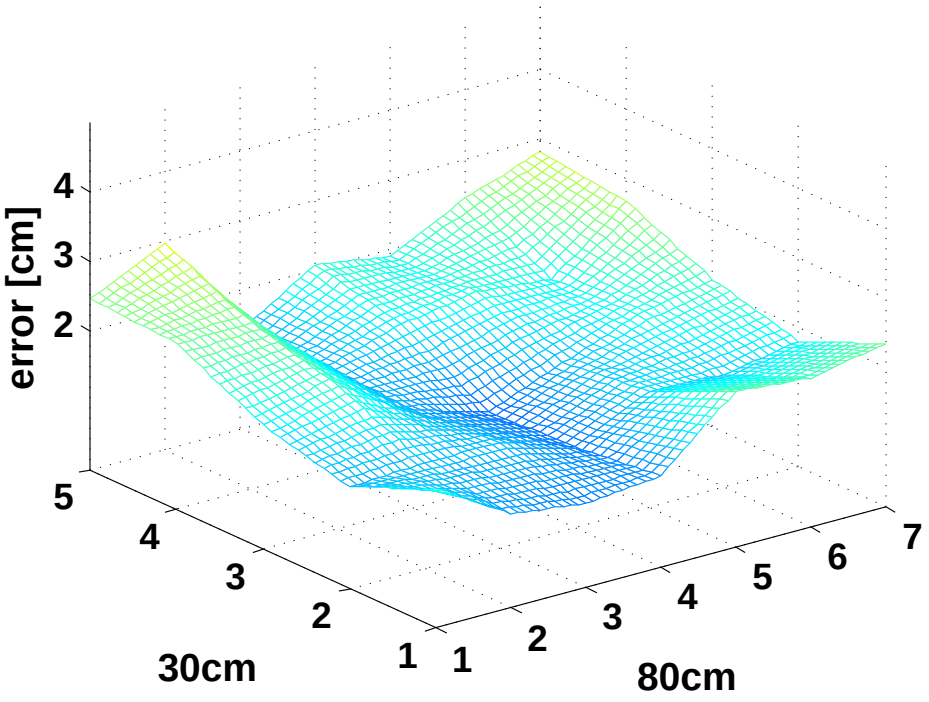

Figure 7: Synthesis Error of Pointing for 3×3 PHMM.

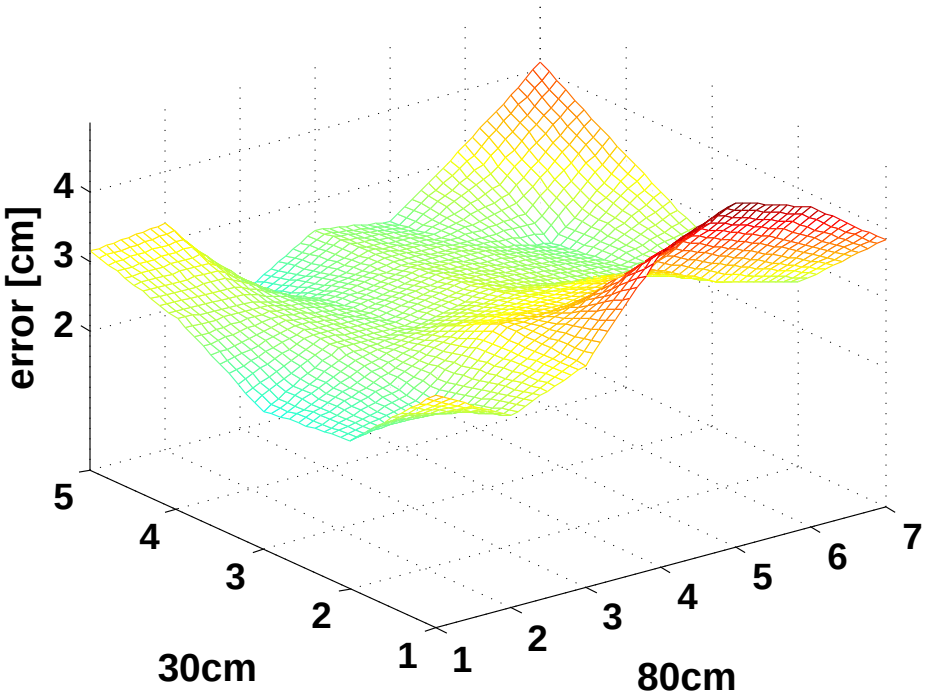

Figure 8: Synthesis Error of Pointing for Wilson's 2×2-trained PHMM.

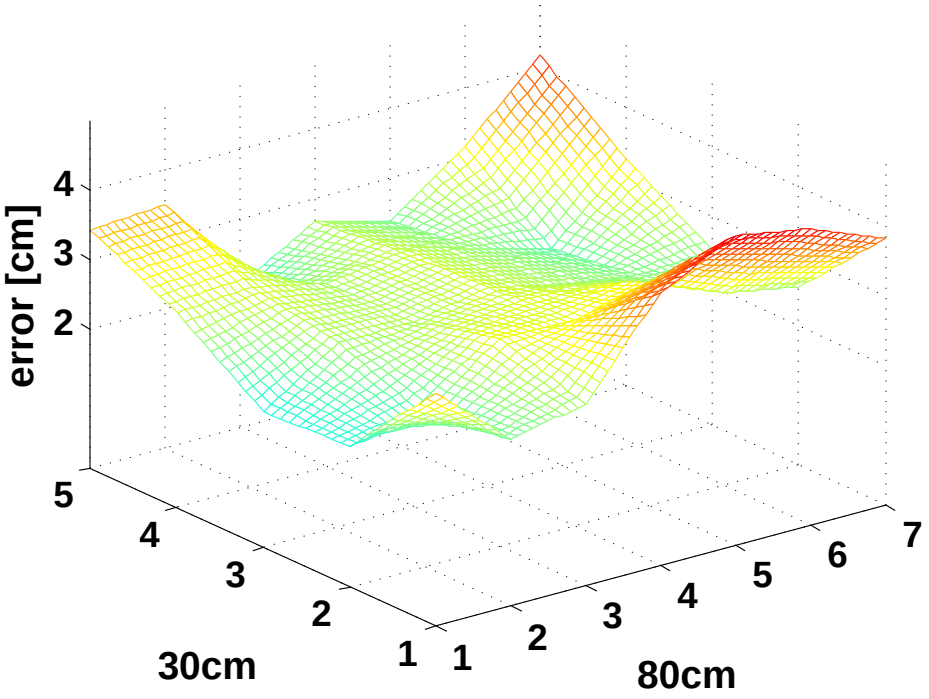

Figure 9: Synthesis Error of Pointing for Wilson's 3×3-trained PHMM.

5.2 Synthesis

Synthesis is done as described above in Sec. 4.1. The performance of synthesis is systematically evaluated by plotting the synthesis error for each of the 5×7 positions, for which test exemplars have been recorded.

The error calculation for each of the 35 synthesized movements for the raster is done as follows: The

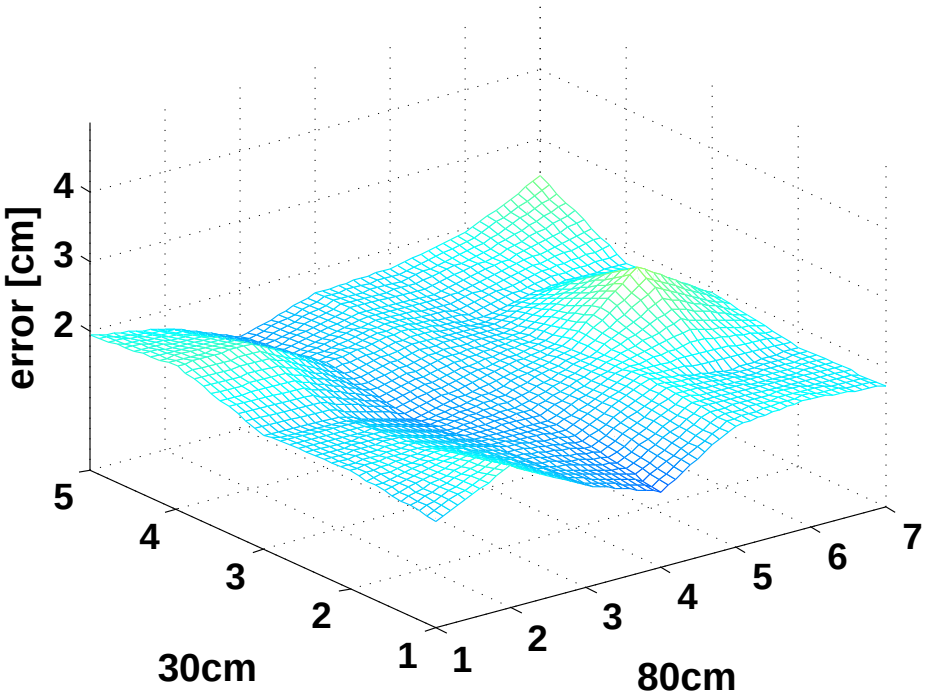

Figure 10: Synthesis Error of Grasping for 3×3 PHMM.

error is calculated as a distance measure between the synthesized movement and a statistical ground truth estimate which is based on the four test exemplars. Therefore, the four test movements of a specific position are averaged by first training an 80 state HMM with the test movements and by then re-synthesizing the average movement $\bar{\boldsymbol{f}}(t) = (\bar{\boldsymbol{f}}_i(i))_{i=1}^{7}$ from the HMM. where the $\boldsymbol{f}_i(t)_{i=1,\ldots,7}$ are 3D trajectories (e.g., of the wrist, elbow...).

The error ε of the synthesized movement $\boldsymbol{f}(t) = (\boldsymbol{f}_i(t))_{i=1}^{7}$, which is synthesized based on the PHMM, is calculated as the route-mean-square error between the time warped synthesis, $\boldsymbol{f}(t)$, and reference, $\bar{\boldsymbol{f}}(t)$:

$$\varepsilon = \sqrt{\int \sum_{i=1}^{7} \frac{(\boldsymbol{f}_i(\alpha(t)) - \bar{\boldsymbol{f}}_i(\bar{\alpha}(t)))^2}{7} \mathrm{d}t \Big/ \int \alpha(t) dt}, \tag{3}$$

where $\alpha(t)$ and $\bar{\alpha}(t)$ are warping functions. The calculation of ε is based on the super-sampled sequences using linear interpolation. As the starting and ending points of the reference $\bar{\boldsymbol{f}}(t)$ do vary slightly, the first and last 10% of the sequences are not considered in the error measure.

The Figs. 6, and 7 compare the synthesis errors for the pointing movement over the 5×7 raster (covering a table-top range of 80cm×30cm) for our PHMM approach based on 2×2 and 3×3 exemplar HMMs. Clearly, the performance in the middle of the covered region increases, if the 3×3 PHMM is used. The Figs. 8, 9 show the performance of Wilson's PHMM. Here, the performance does not change dramatically for a training raster of higher resolution. This is, however, not surprisingly as we use only the linear type of Wilson's PHMM. Fig. 10 shows that the results for the grasping action are very similar to the pointing actions.

The synthesis errors are approximately 1.8cm for our PHMM for grasping and pointing, if the outer regions are neglected, where the pose of the person is extremely stretched. For the linear type of Wilson's PHMM, the errors are slightly higher (≈2.5).

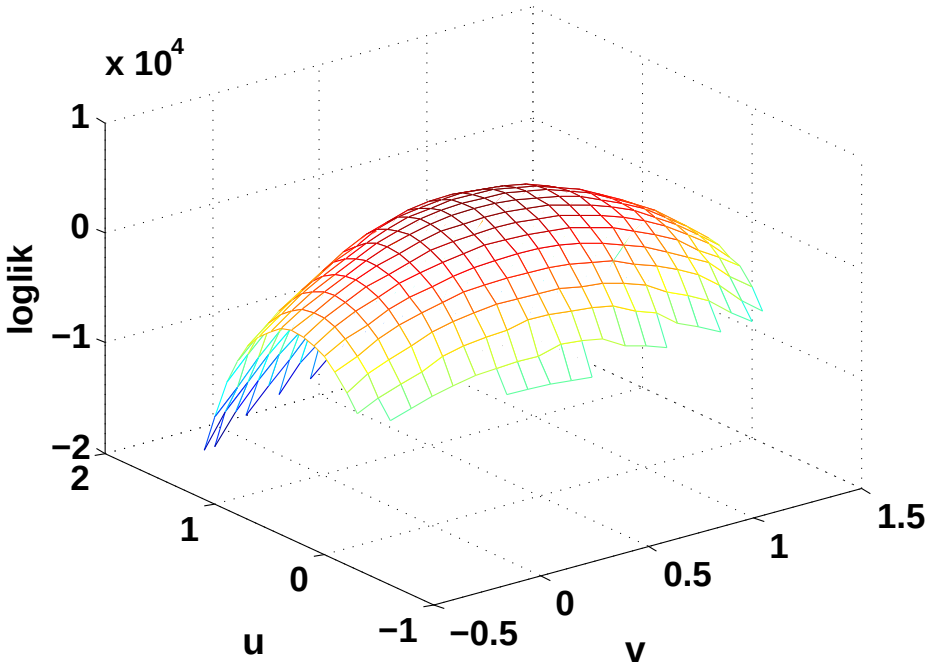

Figure 11: Loglik of the Model Parameters (u, v) given a sequence of Parameterization $(0.5, 0.5)$. The interval $[0, 1]^2 \ni (u, v)$ is mapped to the table-top region of 80cm$\times$30cm.

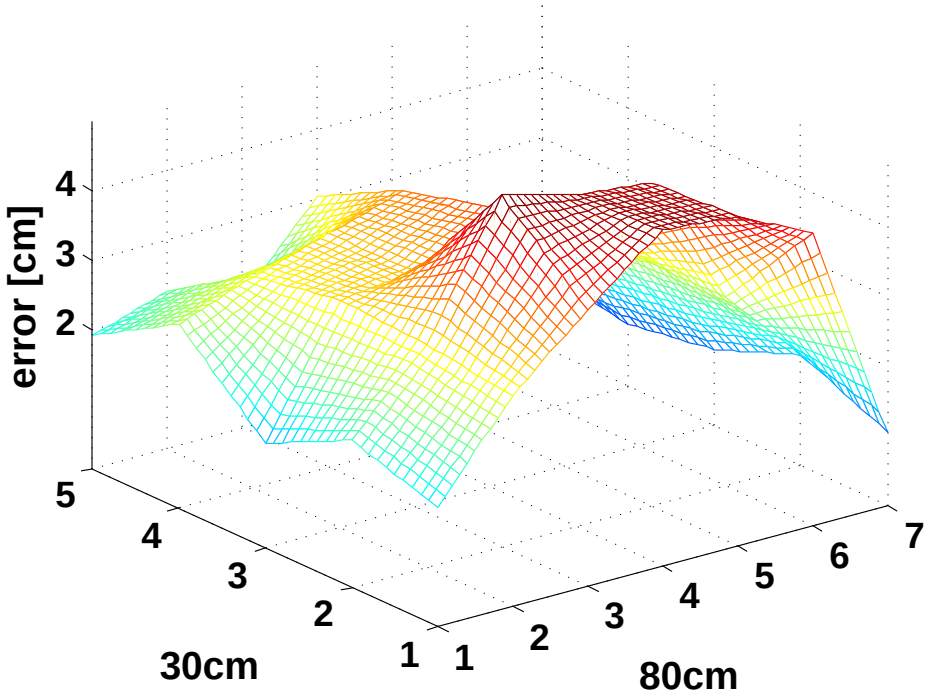

Figure 12: Recognition Error of Pointing for 2$\times$2 PHMM.

5.3 Recognition

Here, two things are to be considered: the recognition performance in terms of the recognized associated position of an action, and rate of correct classifications of the types of the test actions.

In advance, it is worth to take a look at Fig. 11, which gives a hint that the optimization problem of maximizing the log likelihood function $f(u, v) = \log P(\boldsymbol{X}|\boldsymbol{\lambda}^{uv})$ given a movement $\boldsymbol{X}$ is tractable

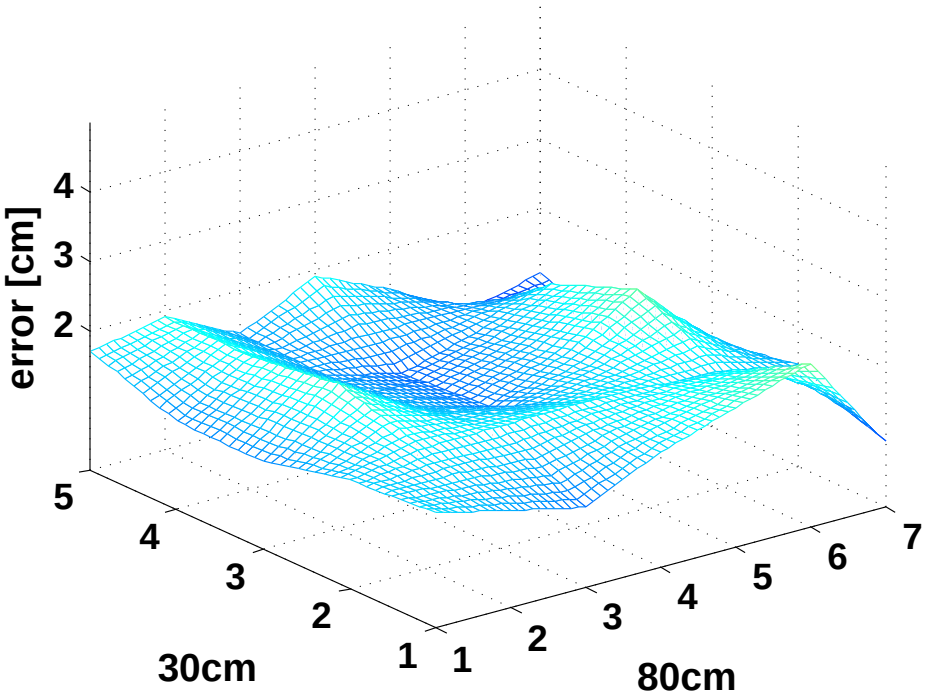

Figure 13: Recognition Error of Pointing for 3×3 PHMM.

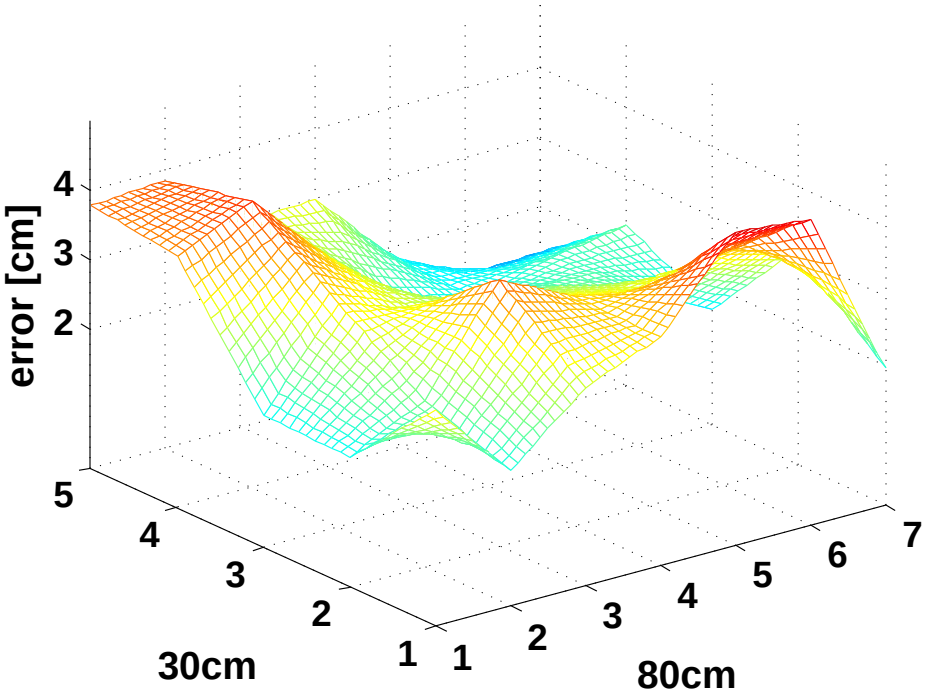

Figure 14: Recognition Error of Pointing for Wilson & Bobick's 3×3-trained PHMM.

by standard optimization techniques (smoothness and strict convexity). In this case, the most likely parameterization (u, v), or associated table-top position of $\boldsymbol{X}$ can be easily estimated. However, in our experiments it has turned out that the maximum of $f(u, v)$ is sometimes a very sharp peak. To address this problem, the function can be smoothed for the first iterations of the optimization by increasing the covariances of the model's Gaussians.

The recognition performance of the associated table-top positions behave very similar to the results

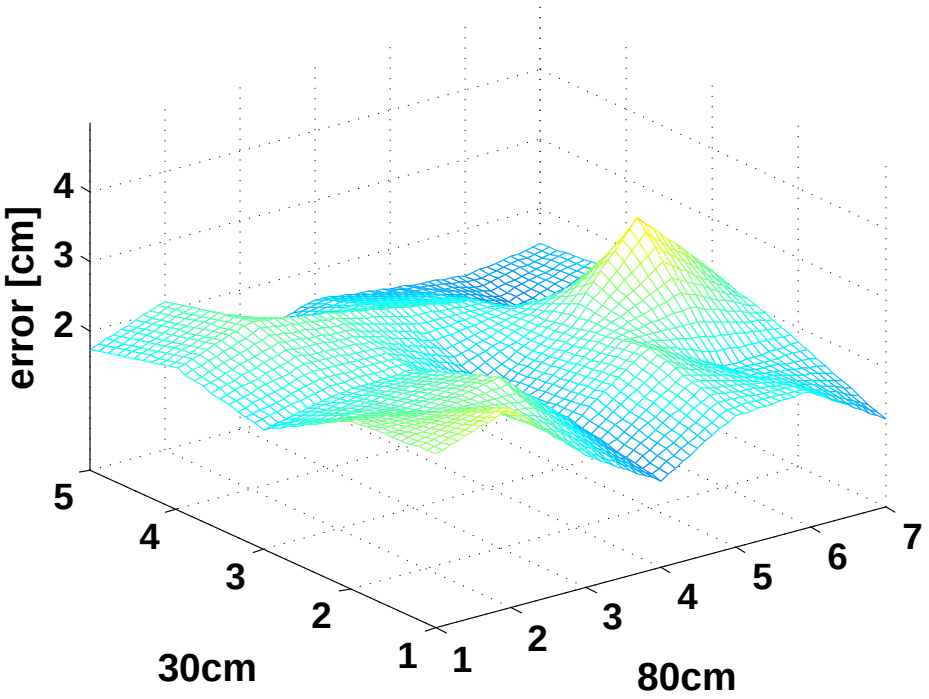

Figure 15: Recognition Error of Grasping for Wilson's 3×3 PHMM.

of synthesis. The error for each position of the 5×7 raster are calculated as the average deviation of the estimated position and the ground truth position for all four test example movements. The recognition performance of our PHMM for pointing and grasping and the performance of Wilson and Bobick's PHMM are presented (Fig. 12–15). Again, clearly, the performance increases in the inner region for our 3×3 PHMM (Fig. 12) compared to the 2×2 PHMM (Fig. 13).

The recognition performance of our 3×3 HMM are similar to Wilson and Bobick's linear type of PHMM (averaged errors of ≈2cm, and slightly smaller for our PHMM).

The rate of right-classified types of the 280 grasping and pointing test movements decreases from 94% to 93% by using the 3×3 PHMMs instead of the 2×2 PHMMs. It is 95% for Wilson and Bobick's PHMM, independently from the used training data (data of the 3×3 or 2×2 raster).

5.4 Online Recognition

Our online demo [REF] shows the applicability of our approach for online recognition of pointing, the position pointed at, and also for motion synthesis. For the synthesis of the robot's arm movement, a PHMM is used, that is trained by the data used in the experiments above. The recognition is based on the position of the elbow and wrist, that are estimated by our online body tracker based on 3D data of a stereo head camera. For the recognition, a PHMM is trained for the last part of pointing movements. The recognition of the position is estimated as the most likely parameter of the PHMM over a recent time window of the elbow and wrist positions, which is recognized as pointing by simple thresholding.

6 Conclusion

We have presented and evaluated a novel approach to handle recognition and synthesis of movements of particular type (sematic), which vary in a parametric way. The basic idea is to incorporate the parameterization of the movements into the HMM (PHMM). Contrary to Wilson and Bobick [12], where the model learns the variation of the Gaussian means, we align some exemplar HMMs for specific pa-

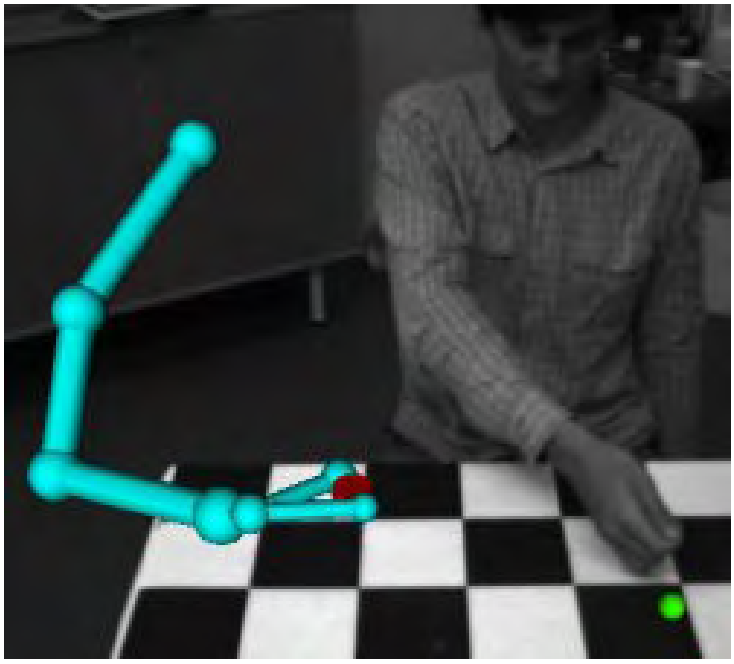

Figure 16: Online Demo. A person is advising a virtual robot arm to relocate a red object (currently, in hand of the robot) at table-top. A person is pointing at the new position. The ball nearby the person's hand indicates the recognized position. The current color (green) of the ball indicates a high likelihood of a pointing movement.

rameterizations, so that the interpolation between the Gaussians is senseful. In our approach all PHMM parameters are allowed to vary depending on the parameterization, unlike in [12].

The experiments show the applicability of our approach for synthesis and recognition of movements (errors ≈2cm), where the performance is similar compared to that of Wilson and Bobick's approach. The classification rate is for both approaches similar $\approx 94\%$.

Finally, it's worth mentioning—even though, we did not compare to the nonlinear case of Wilson and Bobick's approach—that our approach should perform better in such cases, where the movements do vary strongly (as all PHMM parameters can change), with the draw back that several exemplar HMMs have to be setup.

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

References

[1] T. Asfour, K. Welke, A. Ude, P. Azad, J. Hoeft, and R. Dillmann. Perceiving objects and movemetns to generate actions on a humanoid robot. In *Proc. Workshop: From features to actions – Unifying perspectives in compnational and robot vision, ICRA*, Rome, Italy, April 2007. 1, 3

[2] B. Dariush. Human Motion Analysis for Biomechanics and Biomedicine. *Machine Vision and Applications*, 14:202–205, 2003. 1

[3] S. Gunter and H. Bunke. Optimizing the number of states, training iterations and gaussians in an hmm-based handwritten word recognizer. *icdar*, 01:472, 2003. 3

[4] X. Huang, Y. Ariki, and M. Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990. 3

[5] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. 3, 4

[6] C. Lu and N. Ferrier. Repetitive Motion Analysis: Segmentation and Event Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):258–263, 2004. 3

[7] T. Moeslund, A. Hilton, and V. Krueger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–127, 2006. 3

[8] D. Ormoneit, H. Sidenbladh, M. Black, and T. Hastie. Learning and Tracking Cyclic Human Motion. In *Workshop on Human Modeling, Analysis and Synthesis at CVPR*, Hilton Head Island, South Carolina, June 13-15 2000. 1

[9] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986. 3

[10] S. Schaal. Is Imitation Learning the Route to Humanoid Robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999. 1

[11] D. Vecchio, R. Murray, and P. Perona. Decomposition of Human Motion into Dynamics-based Primitives with Application to Drawing Tasks. *Automatica*, 39(12):2085–2098, 2003. 3

[12] A. D. Wilson and A. F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999. 2, 3, 4, 7, 13, 14

Exemplar-based Parametric Hidden Markov Models for Recognition and Synthesis of Movements

Dennis Herzog, Volker Krüger, Daniel Grest

Computer Vision and Machine Intelligence Lab
CIT, Aalborg University Copenhagen
Lautrupvang 15
2750 Ballerup, Denmark
Email: {deh,vok,dag}@cvmi.aau.dk

Abstract

A common problem in movement recognition is the recognition of movements of a particular type. E.g. pointing movements are of a particular type but differ in terms of the pointing direction. Arm movements with the goal of reaching out and grasping an object are of a particular type but differ with the location of the involved object. In this paper, we present an exemplar-based parametric hidden Markov model (PHMM) that is able to recognize and synthesize movements of a particular type. The PHMM is based on exemplar movements that have to be "demonstrated" to the system. Recognition and synthesis are carried out through locally linear interpolation of the exemplar movements. Experiments are performed with pointing and grasping movements. Synthesis is done based on the object position as parameterization. In case of the recognition, the coordinates of the grasped or pointed at object are recovered. Our experiments show the flexibility of our exemplar-based PHMMs in terms of the amount of training data and its robustness in terms of noisy observation data.

1 Introduction

One of the major problems in action and movement[1] recognition is to recognize actions that are of the same type but can have very different appearances depending on the situation they appear in. In addition, for some actions these differences are of major importance in order to convey their meaning. Consider for example the movement of a human pointing at an object, "This object there...", with the finger pointing at a particular object. Clearly, for such an action, the action itself needs to be recognized but also the spot in 3D space at which the human is pointing. Only together do these two pieces of information convey the full semantics of the movement. Another common problem is the synthesis of action: This concerns two major problem areas: In robotics, one is interested in teaching robots through simple demonstrations (imitation learning) [3, 13, 2]. In 3D human body tracking, one is interested in using motion models in order to constrain the parameter space (e.g. [11] for simple cyclic motions). In both cases, one is interested in teaching the system in an easy and efficient manner a particular movement so that afterwards, the system is able to synthesize movements of the same type, however, with a different parameterization. Here, we consider grasping movements as an example where a human is reaching out for an object to grasp it[2]. One may perform as demonstration a set of grasping movements. All grasping movements depend on the location of the object to be grasped. In case of a humanoid robot, the synthesis should then allow the robot to perform the learned grasping movements with new parameterizations, e.g., grasping objects at different positions. In case of the 3D body tracking, the synthesis would allow a better prediction of the next pose.

Most current approaches model movements with a set of movement *prototypes*, and identify a movement by identifying the prototype which explains the observed movement best. This approach, how-

[1] we use the terms *action* and *movement* interchangingly. Actions usually denote movements that involve objects.

[2] The precise choice of a hand grasp depends on the type of object, from where it is being grasped, etc. In our discussion, we omit the issue of the different hand grasps and focus only on the arm movements.

ever, has its limits concerning efficiency when the space of possible parameterizations is large.

A pioneering work in this context was done by Wilson and Bobick [15]. Wilson and Bobick presented a parametric HMM approach that is able to learn an HMM based on a set of demonstrations. Their training and recognition approach is based on the EM algorithm, where the parameters of the movements are taken as latent variables. For recognition, they recover the parameter set that explains best the observation.

In this paper, we develop a different parametric hidden Markov model approach. Contrary to Wilson and Bobick, our aim is recognition as well as synthesis. Also, we would like to provide a simpler and more efficient training strategy by being able to simply provide exemplars based on which the generation of novel HMMs can be done.

In the following section, we give a short overview of the related work. In Sect. 3 and 4 we introduce our exemplar-based parametric HMMs. Extensive experimental results are presented in Sect. 5. Conclusions in Sect. 6 complete our paper.

2 Related Work

Most approaches for movement representation that are of interest in our problem context are trajectory based: Training trajectories, e.g., sequences of human body poses, are encoded in a suitable manner. Newly incoming trajectories are then compared with the previously trained ones. A recent review can be found in [9].

Some of the most common approaches to represent movement trajectories use hidden Markov models (HMMs) [12, 5]. HMMs offer a statistical framework for representing and recognition of movements. One major advantage of HMMs is their ability to compensate for some uncertainty in time. However, due to their nature, HMMs are only able to model specific movement trajectories, but they are not able to generalize over a class of movements that vary accordingly to a specific set of parameters.

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 [7]. In order to deal with a large parameter space one ends up, however, with a lot of experts and a large amount of training becomes necessary.

Another extension of the classical HMMs into parametric HMMs was presented in [15], as mentioned above. A more recent approach was presented by [2]. In this work, the interpolation is carried out in spline space where the trajectory of the end-effector is modeled. Apart from the fact that the authors have not yet performed an evaluation of their system, their approach does not seem suitable for controlling the entire arm movements for movement synthesis and recognition.

In addition to HMMs, there are also other movement representations that are interesting in our context, e.g., [14, 8]. However, these approaches share the same problems as the HMM based approaches.

3 Preliminaries

A hidden Markov model is a probabilistic finite state machine, which is generally defined as a triple $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$, where the transition matrix $\boldsymbol{A}$ defines the transition probability between the hidden states $q = 1, \ldots, N$, $\boldsymbol{B}$ defines the output probabilities of each state, and $\boldsymbol{\pi}$ defines the probabilities of each state of being the initial state of a hidden state sequence.

In this approach continues HMMs are used, whose output probabilities are modeled by single Gaussian distribution. For each state i the output distribution $b_i(\boldsymbol{x}) = \mathrm{P}(\boldsymbol{x}|q = i) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is just defined by the parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$. In this simple case of single Gaussian distributions it is more intuitive to use the notation $\mathcal{N}_i(\boldsymbol{x})$ instead of $b_i(\boldsymbol{x})$.

To facilitate synthesis of actions we chose to use left-right HMMs (as used in [4]) to model the actions. Here, a state sequence always has to start at the same state, therefore $\boldsymbol{\pi}$ is set $\boldsymbol{\pi} = (1, 0, \ldots, 0)$. In addition, state transition are restricted to the state itself or to the next state (other transition probabilities are set to zero). Such an HMM is depicted in Fig. 1.

3.1 Recognition using HMMs

For recognition or classification HMMs are generally used as follows: For each specific class k of sequences an HMM $\boldsymbol{\lambda}^k$ is trained by a representative training set $\mathcal{X}^k$ for that class. The training of an HMM $\boldsymbol{\lambda}$ is done by adjusting the model parameters to values, which are maximizing the likelihood

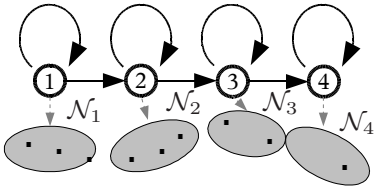

Figure 1: Left-to-Right HMM. For each state i the output normal distribution $\mathcal{N}_i$ is implied by an ellipsoid. An output sequence $\boldsymbol{X} = \boldsymbol{x}_1 \dots \boldsymbol{x}_9$ going from left to right is implied by small black dots. This sequence is likely to be generated by that HMM, where $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}) \gg 0$.

function $\mathrm{P}(\mathcal{X}|\boldsymbol{\lambda})$. For this maximization, we apply the Baum/Welch algorithm [6].

The classification of an specific output sequence $\boldsymbol{X} = \boldsymbol{x}_1 \dots \boldsymbol{x}_T$ is done by selecting that class k, for which the likelihood $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^k) = \max_i \mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^i)$ is maximal.

One obvious approach for handling whole classes of parameterized actions for the purpose of action recognition and parameter estimation is a mixture-of-experts approach [7] and to sample the parameter space by training for each sample a prototype HMM. The HMM maximizing the likelihood of an given action sequence identifies class membership and the parameterization of the action. However, this approach is not appropriate, because too many repetitions of the action are needed to train the prototype HMMs of all samples.

4 Parametric HMMs

The main idea of our approach for handling whole classes of parameterized actions is a supervised learning approach where we generate an HMM for novel action parameters by locally linear interpolation of exemplar HMMs that were previously trained on exemplar movements with known parameters. The generation of a newly parameterized HMM can be done online or offline.

The interpolation of HMMs is carried out statewise, as we will explain in Sect. 4.1. As we will see, an interpolation is only possible, if the states of the exemplar HMMs are aligned in time. This alignment is discussed in Sect. 4.2. In Sect. 4.3 and 4.4, we explain the approaches for recognition and synthesis of the movements.

4.1 Interpolation of HMM

For simplicity the approach is explained in the case of a class of actions parameterized by a single parameter u, e. g., sequences of trajectories of a person's wrist leading to different positions on a table, where the positions are on a straight line leading from left to right on the table. This idea is depicted in Fig. 2.

It is assumed that two HMMs $\boldsymbol{\lambda}^{\mathrm{l}}$ and $\boldsymbol{\lambda}^{\mathrm{r}}$ belonging to motions of different parameterization u are given, where $u = 0$ and $u = 1$ respectively. Additionally, it is assumed that the Gaussians $\mathcal{N}_i^{\mathrm{l}}$ and $\mathcal{N}_i^{\mathrm{r}}$ of $\boldsymbol{\lambda}^{\mathrm{l}}$ and $\boldsymbol{\lambda}^{\mathrm{r}}$ are arranged as depicted in Fig. 2. Under this assumptions the Gaussians $\mathcal{N}_i^u$ of an HMM, that shall model motions leading to some parameterized location u, can be approximated by statewise interpolation of corresponding Gaussians. The interpolation of the Gaussians is directly applied to the means and sigmas:

$$\mathcal{N}_i^u(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^u, \boldsymbol{\Sigma}_i^u), \tag{1}$$

where

$$\begin{aligned} \boldsymbol{\mu}_i^u &= (1-u)\boldsymbol{\mu}_i^{\mathrm{l}} + u\boldsymbol{\mu}_i^{\mathrm{r}} \\ \boldsymbol{\Sigma}_i^u &= (1-u)\boldsymbol{\Sigma}_i^{\mathrm{l}} + u\boldsymbol{\Sigma}_i^{\mathrm{r}} \end{aligned} \tag{2}$$

This is very intuitive in the constellation of Fig. 2, e. g., for $u = 0.5$, the $\mathcal{N}_i^u$ would define horizontal-oriented ellipsoids laying between the $\mathcal{N}_i^{\mathrm{l}}$ and $\mathcal{N}_i^{\mathrm{r}}$. Clearly, the interpolation is not meaningful if the two Gaussians of the same state of different HMMs do not belong to the same part of different motions, e. g., if the third state of one HMM belongs to a part of a forward motion of a forward-and-backward motion and the third state of the other belongs to the backward part of that motion.

Therefore, it is crucial for a meaningful interpolation that learned HMMs are synchronized or statewise aligned.

4.2 Aligned or Synchronized Setup of several HMMs

Starting point are movements that are performed into different directions. Even if they are aligned through linear warping the alignment with in the training sequences is not sufficiently good. Dynamic time warping algorithms [10] are existing just for pairwise alignment of sequences. And time

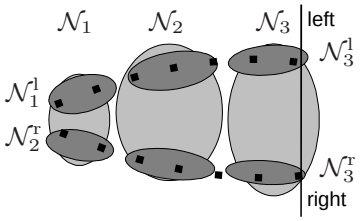

Figure 2: The upper three dark ellipsoids are depicting the output normal densities $\mathcal{N}_1^{\mathrm{l}}, \ldots, \mathcal{N}_3^{\mathrm{l}}$ of the states $1, \ldots, 3$ of an HMM $\boldsymbol{\lambda}^{\mathrm{l}}$ that is trained by sequences beginning on the left side of the picture and are leading onto the left of the vertical line, where the parameter of the parameterization of these sequences is supposed to be $u = 0$. The dots passing this ellipsoids imply one training sequence. Accordingly, the lower three ellipsoids due to an HMM $\boldsymbol{\lambda}^{\mathrm{r}}$ trained by sequences leading to the right of the line, where the parameter u is supposed to be $u = 1$. Additionally, Gaussians $\mathcal{N}_i$ of an $\boldsymbol{\lambda}$ trained by sequences of different parameterizations $u \in [0, 1]$ are indicated in light gray.

warping does not solve the problem of setting up the aligned HMMs. Our aim now is to train HMMs for non-aligned training data such that the HMM states correspond, so that Eq. (2) becomes senseful.

For easier explanation, we will use again the example setup from Sect. 4.1: For both HMMs $\boldsymbol{\lambda}^{\mathrm{l}}$, $\boldsymbol{\lambda}^{\mathrm{r}}$ training sets $\mathcal{X}^{\mathrm{l}}$ and $\mathcal{X}^{\mathrm{r}}$ are given. In order to find the two aligned HMMs $\boldsymbol{\lambda}^{\mathrm{r}}$ and $\boldsymbol{\lambda}^{\mathrm{l}}$, we first train a more general HMM $\boldsymbol{\lambda}$ for the whole training data $\mathcal{X}^{\mathrm{l}} \cup \mathcal{X}^{\mathrm{r}}$. In Fig. 2, we have depicted the general HMM $\boldsymbol{\lambda}$ with light gray ellipses. Now, for each sample $\boldsymbol{x}$ of each sequence $\boldsymbol{X}$ in $\mathcal{X}^{\mathrm{l}}$ and $\mathcal{X}^{\mathrm{r}}$, the closest Gaussian (or State) of the HMM $\boldsymbol{\lambda}$ can be easily assigned. HMM $\boldsymbol{\lambda}$ is then used as the starting point for the training of the specific HMMs $\boldsymbol{\lambda}^{\mathrm{l}}$ and $\boldsymbol{\lambda}^{\mathrm{r}}$ using this assignment. In that sense, HMM $\boldsymbol{\lambda}$ can be interpreted as a pre-alignment of the training data.

In the precise notion of the Baum/Welch EM-algorithm—an iterative algorithm, that repeatedly executes EM steps—that aligned setup is performed, e. g., by starting with the HMM $\boldsymbol{\lambda}^{\mathrm{l}} := \boldsymbol{\lambda}$ trained by the whole training set (likewise: $\boldsymbol{\lambda}^{\mathrm{r}} := \boldsymbol{\lambda}$). Then, one EM step is performed for $\boldsymbol{\lambda}^{\mathrm{l}}$ just using the data $\mathcal{X}^{\mathrm{l}}$ for training. The same procedure is applied to setup all HMMs for the needed key points in parameter space (like the HMM $\boldsymbol{\lambda}^{\mathrm{r}}$ using $\mathcal{X}^{\mathrm{r}}$ in our example). The single final EM step adapts the Gaussians $\mathcal{N}_i$ to its specific exemplar movements, but preserves the alignment.

Now, let's look at the precise EM steps, and let's consider $\boldsymbol{\lambda}^{\mathrm{l}}$. Let $\mathcal{X}^{\mathrm{l}} = \{\boldsymbol{X}^1, \ldots, \boldsymbol{X}^M\}$ be the training set of $\boldsymbol{\lambda}^{\mathrm{l}}$, which repetitions are denoted by $\boldsymbol{X}^{\mathrm{k}} = \boldsymbol{x}_1^k \ldots \boldsymbol{x}_T^k$. In the E step 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 (where $\boldsymbol{x}_t^k$ is emitted) are computed for each $\boldsymbol{X}^k$. Thus, $\gamma_t^k(i)$ defines the responsibility of state i for generating $\boldsymbol{x}_t^k$—or vice versa the membership of $\boldsymbol{x}_t^k$ to state i. In the M step the output probability density functions are re-estimated. Therefore, the means and covariance matrices are re-estimated for each Gaussian $\mathcal{N}_i$ of state i based on the responsibilities $\gamma_t^k(i)$. In the case of a single output sequence $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_T$ the mean re-estimation

$$\boldsymbol{\mu}_i = \frac{\sum_t \gamma_t(i) \boldsymbol{x}_t}{\sum_t \gamma_t(i)} \tag{3}$$

can be regarded as the weighted mean of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ by using the responsibilities as weights. The covariance re-estimation

$$\boldsymbol{\Sigma}_i = \frac{\sum_t \gamma_t(i) (\boldsymbol{x}_t - \boldsymbol{\mu}_t)^\top (\boldsymbol{x}_t - \boldsymbol{\mu}_t)}{\sum_t \gamma_t(i)} \tag{4}$$

can also be regarded as $\gamma_t(i)$-weighted covariance, where $\boldsymbol{\mu}_i$ denotes the previous re-estimation. (In the case of multiple observation $\boldsymbol{X}^k$ the nominators and denominators of (3,4) have to be extended to marginalize $\gamma_t^k(\cdot)$ and $\boldsymbol{x}_t^k$ over k.) As the responsibilities $\gamma_t(i)$ are chosen due to the general HMM $\boldsymbol{\lambda}$ each state of the HMM trained with a subset of the training data remains aligned to those of the other HMMs.

4.3 Recognition of Parameterized Sequences

The recognition of parameterized sequences is straightforward compared to the simple class classification. Given a sequence $\boldsymbol{X}$. For each class k and corresponding parameterized HMM $\boldsymbol{\lambda}_k^{\boldsymbol{\phi}}$ that class k and parameter $\boldsymbol{\phi} \in \mathbb{R}^n$ are taken, which yields the maximal likelihood $\max_{k,\boldsymbol{\phi}} \mathrm{P}(\boldsymbol{X} | \boldsymbol{\lambda}_k^{\boldsymbol{\phi}})$.

In the more general case $n > 1$ with $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$ this is done by using gradient descent methods for the maximization of $P(\boldsymbol{X} | \boldsymbol{\lambda}_k^{\boldsymbol{\phi}})$

for each k separately. Therefore, for a fixed k the parameters of ϕ are iteratively adapted in the direction of the gradient $\frac{\partial}{\partial \phi} P(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\phi})$, into which the likelihood function $f(\phi) = P(\boldsymbol{X}|\boldsymbol{\lambda}_k^{\phi})$ increases. Therefore, the likelihood function f has to be evaluated several times in the iteration process. The computation is done by the Forward/Backward Algorithm [6], which is the standard algorithm for this task. It is worth to be mentioned that the computationally cost of the evaluation of f is very small $\mathcal{O}(NT)$ in case of left-right HMMs. Here, N is the number of states, and T is the length of the output sequence.

In the case of several available exemplar HMMs, one needs to find those four closest exemplar HMMs that interpolate a newly observed movement most accurately. We do this by recursively comparing pairwise sets of HMMs. Consider the case of 2×3 aligned HMMs $\boldsymbol{\lambda}^{ij}$ with associated grasping positions $\boldsymbol{p}^{ij}$, that are forming a grid on the table such as

$$\begin{bmatrix} \boldsymbol{p}^{01} & \boldsymbol{p}^{11} & \boldsymbol{p}^{21} \\ \boldsymbol{p}^{00} & \boldsymbol{p}^{10} & \boldsymbol{p}^{20} \end{bmatrix} .$$

First, the bilinear interpolation parameters u, v of

$$\boldsymbol{\lambda}^{uv} = \begin{pmatrix} 1-u, u \end{pmatrix} \begin{bmatrix} \boldsymbol{\lambda}^{01} & \boldsymbol{\lambda}^{21} \\ \boldsymbol{\lambda}^{00} & \boldsymbol{\lambda}^{20} \end{bmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

maximizing $\mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^{uv})$ are determined. Now, if the point

$$\boldsymbol{p}^{uv} = \begin{pmatrix} 1-u, u \end{pmatrix} \begin{bmatrix} \boldsymbol{p}^{01} & \boldsymbol{p}^{21} \\ \boldsymbol{p}^{00} & \boldsymbol{p}^{20} \end{bmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

is lying on the left side of the line defined by $\boldsymbol{p}^{11}$ and $\boldsymbol{p}^{10}$ the procedure is repeated using the four left most HMMs $\boldsymbol{\lambda}^{01}, \boldsymbol{\lambda}^{00}, \boldsymbol{\lambda}^{11}, \boldsymbol{\lambda}^{10}$, otherwise using the four right most.

4.4 Action Synthesis

Suppose a grasp position $\boldsymbol{p}$ on the table-top is given. Then synthesis can be done as following:

1. The three HMMs $\boldsymbol{\lambda}^{i}{}_{,i=1,2,3}$ with closest associated grasp positions $\boldsymbol{p}^{i}$ are chosen under the constraint that the $\boldsymbol{p}^{i}$ are not collinear.
2. Then the interpolation parameters u, v are estimated as given by the point equation

$$\boldsymbol{p} - \boldsymbol{p}^{1} = \begin{bmatrix} \boldsymbol{p}^{2} - \boldsymbol{p}^{1} \Big| \boldsymbol{p}^{3} - \boldsymbol{p}^{1} \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} .$$

   Here, least-squares approximation has to be used in that case that $\boldsymbol{p}$ is not exactly in the plane of $\boldsymbol{p}^{i}$.
3. As basis for synthesis the interpolated HMM $\boldsymbol{\lambda}^{uv} = \boldsymbol{\lambda}^{1} + u(\boldsymbol{\lambda}^{2} - \boldsymbol{\lambda}^{1}) + v(\boldsymbol{\lambda}^{3} - \boldsymbol{\lambda}^{1})$ is used. The synthesized sequence is the sequence $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_N$ of the means of the HMM $\boldsymbol{\lambda}^{uv}$. As the number of states N is small compared to the original recorded sequences, the sequence is linearly interpolated.

5 Experiments

In our experiments we focus our considerations on pointing and grasping actions, which are in common action scenarios the most important movements. The pointing movements were movements such as “This object there...”. Our grasping movements are reaching towards a particular object in order to grasp it.

In our experiments we do not use visual tracking data but limit our considerations on data that is acquired using a motion capture system. This way, we exclude the vision problem and are able to focus only on the representational issues for movement representation.

The motion capture data is acquired with the electro-magnetic motion capture system *Motion Star* by *Ascension*[1]. For the capturing seven markers are placed on the person (see Fig. 3): one

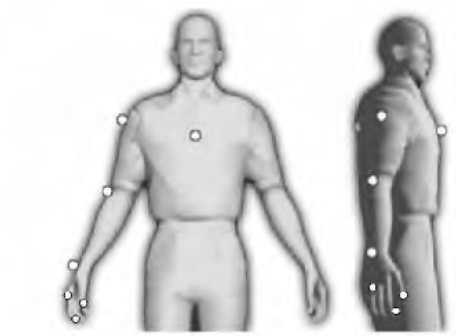

Figure 3: The dots mark the positions of the electromagnetic sensors in our dataset.

at the neck and one at the shoulder, the wrist, in the middle of the hand, at the index finger and the thumb of the right elbow. The captured sequences consist of the 3D point positions for each of the

seven markers recorded with 25Hz. The person or actor sits in front of a table (see Fig. 4). The positions on the table, on which the actions are performed, are covering a region of 30*cm* (in deeps) by 80*cm* (in width). For recognition the 3D point positions of the markers of right arm are directly used for training the HMMs. We have recorded six different exemplar movements (i.e. (action/grasping) movements to six different distinct positions on the table), each one with 9 repetitions. Two table positions are at the left and two are at the right side of the $30cm \times 80cm$ region. The other two exemplars are in the middle. For validation of the approach, additional movements to 10 different random positions on the table were recorded with 4 repetitions, each. All training and test sequences are normalized to 50 samples in length (2 sec.). No additional temporal alignment between the sequences was done.

Figure 4: The image shows the setup of the recording session of our dataset.

5.1 Synchronized Setup of HMMs

The synchronized setup of the special/prototype HMMs for the 6 exemplar positions is done as described in Sect. 4.2. As one of the aims is to provide an approach with fast and simple training, we will focus our investigation of synthesis and recognition performance vs. number of HMM states N, number of exemplar movements and number of repetitions R. In our experiments, we have trained HMMs with $N = 16, 28, 40$ states, with 4 or 6 exemplar movements and with $R = 3, 6, 9$ repetitions[3]. Fig. 5 shows example images of different states of the synchronized HMMs (yellow) and the general HMM (red). Here, one can verify the synchronization visually. In the recognition and synthesis experiments, below, we will implicitly verify the quality of the alignment.

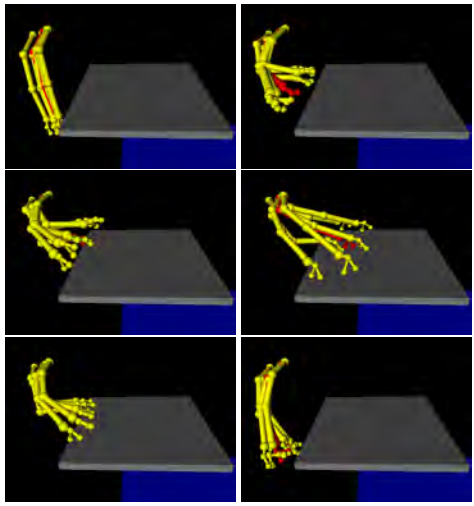

Figure 5: The 6 images (row-wise) are showing the states 1,7,9,14,25 and 30 of six synchronized HMMs with 40 states trained with grasping sequences of the six exemplar positions on the table (yellow arms) and the same states of the general HMM (red), that was used to align the states of the six HMMs.

5.2 Synthesis

In this section we summarize our results for the synthesis experiments. Here, we have tested the quality of the synthesis of movements, given a specific parameterization. The parameters were given as coordinates on the table. As test parameters we used those of the 10 random test movements. The quality of our synthesis approach has been tested based on these 10 random test movements, which serve here as ground truth references with known parameters (coordinates). The synthesis tests have been done for pointing and for grasping movements.

As explained above, we have trained different exemplar HMMs with different number of states and different number of repetitions. To estimate the synthesis quality we have tested based on

[3]The relatively high number of states is used to assure a reasonable good precision of the synthesis. Arguably, we could have chosen HMMs with less states.

1. different number of states $N = 16, 28, 40$,
2. different number of repetitions $R = 3, 6, 9$,
3. 4 and 6 exemplar movements for interpolation.

Tab. 1 and Tab. 2 summarize the experimental results for different number of HMM states, number of repetitions for each exemplar movement and number of exemplars used for the synthesis. As the intuitively best configuration, we chose a setting of 16 states, 6 repetitions for each exemplar and 6 exemplars movements. This setting is highlighted bold for grasping and pointing (top of each table).

The errors in the table were computed as follows: A movement is synthesized for each of the random table positions. Each synthesized movement is compared to the corresponding reference movement by computing the average Euclidean distance between the synthesized movement and the reference movement. The mean and standard deviation of this error are listed in the tables as *Synthesis Error* and is given in *cm*.

There is a natural variance in the human movements. As a reference, we have computed the mean error and the variance for the 40 different reference movements (10 movements with 4 repetitions each). These values are denoted in the tables as *Intrinsic Error* and are again given in *cm*.

Table 1: Synthesis of Grasping. For different numbers of States, Repetitions of exemplars and different number of Exemplars: the average Synthesis Error for 10 randomly chosen positions and the Standard Deviation in regard to the positions are listed in *cm*. Additionally, the Intrinsic Error Means and Deviations are listed.

			Synth. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**2.7**	**0.7**	**1.4**	**0.4**
16	6	**4**	3.2	0.7	1.4	0.4
28	6	6	3.0	0.6	1.4	0.4
40	6	6	3.1	0.6	1.4	0.4
16	**3**	4	2.7	0.7	1.4	0.4
16	**9**	4	2.8	0.7	1.4	0.4

One can see that the synthesis error is rather small $\approx 3cm$, where the intrinsic error is above $1cm$. Furthermore, it is interesting to see that the influence of different number of states or repetitions on the synthesis quality is surprisingly small. Also interesting is that the experiment with only three repetitions and only four exemplars gave one of the best results.

Table 2: Synthesis of Grasping. For different numbers of States, Repetitions of exemplars and number of Exemplars: the Synthesis and Intrinsic Errors for 10 positions are listed in *cm*.

			Synth. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**2.5**	**0.5**	**1.5**	**0.3**
16	6	**4**	2.6	0.5	1.5	0.3
28	6	6	2.6	0.5	1.5	0.3
40	6	6	2.6	0.5	1.5	0.3
16	**3**	4	2.5	0.4	1.5	0.3
16	**9**	4	2.4	0.5	1.5	0.3

5.3 Recognition

In the experiments for recognition, we do not test the ability to recognize the type of movement (pointing/grasping), but how good our approach is able to recover the parameterization of a newly observed movement. Once the parameters are known, the recovery of the movement type is trivially solved with ML. We have used for testing the 80 random movements (10 movements with 4 repetitions each for pointing and grasping) with known ground truth. For each of these movements, the parameters, i.e., the coordinates on the table, were recovered and compared to the ground truth values.

As above, we have run our experiment with different settings (number of states, number of repetitions used for the prototype exemplars and number of prototypes). The recognized positions $\boldsymbol{p} = \boldsymbol{p}(u, v)$ are calculated by using the interpolation parameters u, v, which maximize the likelihood function $f(u, v) = \mathrm{P}(\boldsymbol{X}|\boldsymbol{\lambda}^{uv})$ of the bilinear interpolation between the 4 HMMs of the nearest exemplars. As the grasping/pointing positions of the 6 exemplars are labeled, the recognized position $\boldsymbol{p}$ is calculated through bilinear interpolation.

In Tab. 3 and Tab. 4 the recognized position mean error and standard deviation with respect to the 80 test movements are summarized. It is interesting to note that since the random pointing actions were supposed to be performed in a "natural" way, the in-

Table 3: Recognition of Grasping Positions. The Recognition Error and Intrinsic Error are listed for each setup.

			Recog. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**3.2**	**1.1**	**0.6**	**0.3**
16	6	**4**	5.0	1.5	0.6	0.3
28	6	6	2.7	1.1	0.6	0.3
40	6	6	2.7	1.2	0.6	0.3
16	**3**	4	3.0	1.0	0.6	0.3
16	**9**	4	3.2	1.1	0.6	0.3

Table 4: Recognition of Pointing Positions. Here, the recognition of the meant Pointing Positions on the table (upper block of table) and Fingertip Positions (lower block) are listed, separately. (The Fingertip doesn't touch the table!) The Recognition Error and Intrinsic Error are listed for each setup. However, as the meant pointing positions on the table are given ahead, no intrinsic error is meaningful.

			Recog. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**2.5**	**1.0**		
16	6	**4**	3.7	0.8		
28	6	6	2.5	0.9		
40	6	6	2.4	0.9		
16	**3**	4	2.7	1.2		
16	**9**	4	2.3	0.9		
16	**6**	**6**	**4.2**	**1.9**	**2.0**	**0.9**
16	6	**4**	4.7	1.4	2.0	0.9
28	6	6	4.2	1.7	2.0	0.9
40	6	6	4.0	1.7	2.0	0.9
16	**3**	4	4.7	2.1	2.0	0.9
16	**9**	4	4.1	1.9	2.0	0.9

dex finger did not touch the table but rather stopped some *cm* above! Therefore, we summarize in the Tab. 4 two different errors: the error of the recognized pointing position on the table, which was meant by the person as elongation of the pointing direction, and the index finger tip position itself.

As before, we compute the intrinsic mean errors and standard deviations of the ground-truth coordinates of the 80 test movements. However, as the meant pointing position on the table is predefined and supposed to be correct, no intrinsic errors are listed. On the other hand, the grasping positions and the fingertip positions of the pointing actions are given by the motion sequences.

The entire experiment has been repeated with the random test movements disturbed by noise. The added noise was Gaussian with $\sigma = 5, 10, 15cm$. Tab. 5 and Tab. 6 summarize the results for $\sigma = 15cm$. For $\sigma = 5, 10cm$, no or only a very minor error increase was observable.

The errors of the recognized positions are acceptable ($\approx 3cm$) and as presumed slightly higher for the meant pointing positions on the table. In contrast to the results of synthesis fewer prototype exemplars (4 instead of 6) give significantly less accurate results (errors $\approx 4 - 5cm$).

Table 5: Recognition of Grasping Positions with Noise. The noise is normal distributed with sigma equal 15*cm*.

			Recog. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**4.1**	**1.5**	**0.6**	**0.3**
16	6	**4**	5.7	1.3	0.6	0.3
28	6	6	3.5	1.3	0.6	0.3
40	6	6	3.0	1.3	0.6	0.3
16	**3**	4	4.5	1.7	0.6	0.3
16	**9**	4	3.0	1.2	0.6	0.3

6 Conclusion

In this paper we have presented an exemplar-based parametric hidden Markov model which allows to represent entire classes of movements. We have focused our considerations on human arm movements, but we believe that the approach can also be used in other contexts such as surveillance scenarios.

We have limited our considerations on movement data captured with an electro-magnetic motion capture system. We view the vision and the movement representation issues as two distinct problems and have focused our considerations on the movement representation alone. Presently, we are in the process of combining our movement representation with our 3D human body tracker.

Table 6: Recognition of Pointing Positions with Noise. The noise is normal distributed with sigma equal 15*cm*. Again, the recognition of the meant Pointing Positions on the table (upper block) and Fingertip Positions (lower block) are listed, separately.

			Recog. Err.		Intr. Err.	
St.	Rp.	Ex.	Mean	σ	Mean	σ
16	**6**	**6**	**4.0**	**0.8**		
16	6	**4**	4.3	1.0		
28	6	6	3.2	1.2		
40	6	6	3.2	1.2		
16	**3**	4	5.0	2.0		
16	**9**	4	4.5	2.2		
16	**6**	**6**	**5.1**	**1.2**	**2.0**	**0.9**
16	6	**4**	5.1	1.8	2.0	0.9
28	6	6	4.9	1.6	2.0	0.9
40	6	6	4.9	1.6	2.0	0.9
16	**3**	4	6.2	2.1	2.0	0.9
16	**9**	4	5.3	3.0	2.0	0.9

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

References

[1] *Motion Star Real-Time Motion Capture*, http://www.ascension-tech.com/products/motionstar_10_04.pdf edition.

[2] T. Asfour, K. Welke, A. Ude, P. Azad, J. Hoeft, and R. Dillmann. Perceiving objects and movemetns to generate actions on a humanoid robot. In *Proc. Workshop: From features to actions – Unifying perspectives in compnational and robot vision, ICRA*, Rome, Italy, April 2007.

[3] B. Dariush. Human Motion Analysis for Biomechanics and Biomedicine. *Machine Vision and Applications*, 14:202–205, 2003.

[4] Simon Gunter and Horst Bunke. Optimizing the number of states, training iterations and gaussians in an hmm-based handwritten word recognizer. *icdar*, 01:472, 2003.

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

[6] Xuedong Huang, Yasuo Ariki, and Mervyn Jack. *Hidden Markov Models for Speech Recognition*. Columbia University Press, New York, NY, USA, 1990.

[7] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[8] C. Lu and N. Ferrier. Repetitive Motion Analysis: Segmentation and Event Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):258–263, 2004.

[9] T. Moeslund, A. Hilton, and V. Krueger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–127, 2006.

[10] M. Mozerov, I. Rius, X. Roca, and J. Gonzàlez. 3d human motion sequences synchronization using dense matching algorithm. In *In 28th Annual Symposium of the German Association for Pattern Recognition (DAGM'2006)*, volume LNCS-4174, pages 475–489, Berlin, Germany, September 2006.

[11] D. Ormoneit, H. Sidenbladh, M.J. Black, and T. Hastie. Learning and Tracking Cyclic Human Motion. In *Workshop on Human Modeling, Analysis and Synthesis at CVPR*, Hilton Head Island, South Carolina, June 13-15 2000.

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

[13] S. Schaal. Is Imitation Learning the Route to Humanoid Robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[14] D.D. Vecchio, R.M. Murray, and P. Perona. Decomposition of Human Motion into Dynamics-based Primitives with Application to Drawing Tasks. *Automatica*, 39(12):2085–2098, 2003.

[15] Andrew D. Wilson and Aaron F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Stereo-based Markerless Human Motion Capture for Humanoid Robot Systems

Pedram Azad[1] Aleš Ude[2] Tamim Asfour[1] Rüdiger Dillmann[1]

Abstract— **In this paper, we present an image-based markerless human motion capture system, intended for humanoid robot systems. The restrictions set by this ambitious goal are numerous. The input of the system is a sequence of stereo image pairs only, captured by cameras positioned at approximately eye distance. No artificial markers can be used to simplify the estimation problem. Furthermore, the complexity of all algorithms incorporated must be suitable for real-time application, which is maybe the biggest problem when considering the high dimensionality of the search space. Finally, the system must not depend on a static camera setup and has to find the initial configuration automatically.**

We present a system, which tackles these problems by combining multiple cues within a particle filter framework, allowing the system to recover from wrong estimations in a natural way. We make extensive use of the benefit of having a calibrated stereo setup. To reduce search space implicitly, we use the 3D positions of the hands and the head, computed by a separate hand and head tracker using a linear motion model for each entity to be tracked. With stereo input image sequences at a resolution of 320×240 pixels, the processing rate of our system is 15 Hz on a 3 GHz CPU. Experimental results documenting the performance of our system are available in form of several videos.

I. Introduction

The idea of markerless human motion capture is to capture human motion without any additional arrangements required, by operating on image sequences only. Implementing such a system on a humanoid robot and thus giving the robot the ability to perceive human motion would be valuable for various reasons. Captured trajectories, which are calculated in joint angle space, can serve as a solid base for learning human-like movements. Commercial human motion capture systems such as the VICON system [1], which are popular both in the film industry and in the biological research field, require reflective markers and time consuming manual post-processing of captured sequences. A real-time human motion capture system using the image data acquired by the robot's head would make one big step toward autonomous online learning of movements. Another application for the data computed by such a system is the recognition of actions and activities, serving as a perceptive component for human-robot interaction. However, providing data for learning of movements and actions – often referred to as learning-by-imitation – is the more challenging goal, since transforming captured movements in configuration space into the robot's kinematics and reproducing them on the robot sets the higher demands to smoothness and accuracy.

[1]Institute for Computer Science and Engineering, University of Karlsruhe (TH), Germany
[2]Jozef Stefan Institute, Ljubljana, Slowenia

For application on an active head of a humanoid robot, a number of restrictions has to be coped with. In addition to the limitation to two cameras positioned at approximately eye distance, one has to take into account that an active head can potentially move. Furthermore, computations have to be performed in real-time, preferably at 30 Hz or higher, in order to achieve optimal results.

The general problem definition is to find the correct configuration of the underlying articulated 3D human model for each input image respectively image tuple when using multiple cameras. The main problem is that search space increases exponentionally with the number of Degrees Of Freedom (DOF). A realistic model of the human body has at least 14 DOF if only modeling the upper body without the neck (3 DOF for each shoulder, 1 DOF for each elbow, 6 DOF for base translation and rotation), or 25 DOF for the full body (plus 3 DOF for each hip joint, 1 DOF for each knee, 3 DOF for the neck), leading to a very high-dimensional search space.

There are several approaches to solve the general problem of markerless human motion capture, differing in the sensors incorporated and the intended application. When using multiple cameras, i.e. three or more cameras located around the area of interest, two different systems have shown very good results. The one class of approaches is based on the calculation of 3D voxel data, as done by [2], [3]. The other approach is based on particle filtering and became popular by the work of Deutscher et al. [4]. Other approaches depend on incorporation of an additional 3D sensor and the *Iterative Closest Point* (ICP) algorithm, such as the Swiss Ranger, as presented by [5]. Other approaches concentrate on deriving as much information as possible from monocular image sequences [6], and reducing the size of the search space by applying restrictions to the range of possible movements, e.g. by incorporating a task-specific dynamic model [7]. Our experience is that it is not possible to build a general 3D human motion capture system using monocular image sequences only, since in many cases a single camera is not sufficient to determine accurate 3D information, based on the principle *depth through scaling*. A further strategy to reduce search space is search space decomposition i.e. performing a hierarchical search, as done by [8]. However, by doing this, the power of the system is limited, since in many cases the global view is needed to determine the correct configuration, e.g. for rotations around the body axis, the information provided by the positions of the arms is very helpful.

Recently, we have started to adapt and extend the particle

filter based system for real-time application on a humanoid robot head ([9], [10]), presenting our newest results in the following. Particle filtering has proven to be an applicable and robust technique for contour tracking in general ([11], [12], [13]), and for human motion capture in particular, as shown in [4], [6]. However, in particle filters, a larger search space requires a greater number of particles. One strategy to cope with this problem is to reduce the dimensionality of configuration space by restricting the range of the subject's potential movements, as already mentioned, or to approach a linear relationship between the dimension of configuration space and the size of the search space by performing a hierarchical search. A general but yet effective way to reduce the number of particles is based on the idea of *Simulated Annealing*, presented in [4], [14]. However, the final system, which uses three cameras at fixed positions in the corners of a room, requires on average 15 seconds to process one frame on a 1 GHz CPU [14].

Theoretically, an edge-based cue would be already sufficient to track the movements of a human – if using an adequate number of particles. To span the search space with a sufficient resolution when using an edge-based cue only, millions of particles would be necessary for a successful tracker. Therefore, the common approach using particle filters for human motion capture is to combine edge and region information within the likelihood function, which evaluates a given configuration matching the current observation. Although this is a powerful approach, the computational effort is relatively high. Especially the evaluation of the region based cue is computationally expensive.

Our strategy is to combine as many cues derivable from the input images as possible to reduce search space implicitly by achieving a faster convergence. We present a running system on our humanoid robot ARMAR using the benefits of a stereo setup and combining edge, region and skin color information. The initial configuration is found automatically – a necessity for any perceptive component of a vision system. The system is able to capture real 3D motion without using markers or manual post-processing. The processing rate of our algorithm is 15 Hz on a 3 GHz CPU using stereo input image sequences with a resolution of 320×240 pixels.

II. Using Particle Filters for Human Motion Capture

Particle filtering has become popular for various visual tracking applications – often also referred to as the Condensation Algorithm. The benefits of a particle filter compared to a Kalman filter are the ability to track non-linear movements and the property to store multiple hypotheses simultaneously. The price one has to pay for these advantages is the higher computational effort. The probability density function representing the likelihood of the configurations in configuration space matching the observations is modeled by a finite set of N particles $S = \{(\mathbf{s_1}, \pi_\mathbf{1}), ..., (\mathbf{s_N}, \pi_\mathbf{N})\}$, where $\mathbf{s_i}$ denotes one configuration and π_i the likelihood associated with it. The core of a particle filter is the likelihood function $p(\mathbf{z}|\mathbf{s})$ computing the a-posteriori probabilities π_i, where $\mathbf{s}$ denotes a given configuration and $\mathbf{z}$ the current observations i.e. the current image pair. This likelihood function must be evaluated for each particle for each frame i.e. $N \cdot f$ times per second. As an example this means for $N = 1000$ particles and $f = 30$ Hz $N \cdot f = 30000$ evaluations per second.

For resampling the particle set, N particles from the last generation are picked proportional to their probability. The configuration of each of these particles is used as a base to build a new configuration, incorporating a motion model and adding Gaussian noise. We use a first-order linear motion model together with adaptive Gaussian noise, whose amount is decreased or increased depending on the current errors. A detailed description about the use of particle filters for human motion capture can be found in [9].

A. Edge Cue

Given the projected edges of a configuration $\mathbf{s}$ of the human model and the current input image $\mathbf{z}$, the likelihood function $p(\mathbf{z}|\mathbf{s})$ for the edge cue calculates the a-posteriori probability that the configuration leading to the set of projected edges is the proper configuration i.e. the one taht best matches the gradient image.

The approach we use is to spread the gradients in the gradient image with a Gaussian filter or any other suitable operator and to sum the gradient values along a projected edge, as done in [4]. Assuming that the spread gradient map has been remapped between 0 and 1, the modified likelihood function can be formulated as:

$$p_g(\mathbf{z}|\mathbf{s}) \propto \exp\left\{ -\frac{1}{2\sigma_g^2 M_g} \sum_{m=1}^{M_g} (1 - g_m)^2 \right\}$$

where g_m denotes the remapped gradient value for the mth point.

B. Region Cue

The second cue commonly used is region-based, for which a foreground segmentation technique has to be applied. The segmentation algorithm to be picked is independent from the likelihood function itself. The most common approach is background subtraction. However, this segmentation method assumes a static camera setup and is therefore not suitable for application on a potentially moving robot head. Another option is to segment motion by using difference images or optical flow, but this method also assumes a static camera setup. It has to be mentioned that there are extensions of the basic optical flow algorithm that allow to distinguish real motion in the scene and ego-motion [15]. However, the problem with all motion-based methods – which does not include background subtraction – is that the quality of the segmentation result is not sufficient for a region-based cue. Only those parts of the image that contain edges or any other kind of texture can be segmented, and the silhouette of segmented moving objects often contains parts of the background, resulting in a relatively blurred segmentation result.

Having segmented the foreground in the input image, where foreground pixels are set to 1 and background pixels

are set to 0, the likelihood function commonly used can be formulated as [4]:

$$p_r(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_r^2 M_r}\sum_{m=1}^{M_r}(1-r_m)^2\right\} \quad (1)$$

where r_m denotes the segmentation value of the mth pixel from the set of pixels of all projected body part regions. Although this function can be optimized further, using the fact that $r_m \in \{0,1\}$, its computation is still rather inefficient. The bottleneck is the computation of the set of all M projected pixels together with reading the corresponding values from the segmentation map.

C. Fusion of Multiple Cues

The both introduced cues are fused by multiplying the two likelihood functions resulting in:

$$p_{g,r}(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2}\left(\frac{\sum_{m=1}^{M_g}(1-g_m)^2}{\sigma_g^2 M_g} + \frac{\sum_{m=1}^{M_r}(1-r_m)^2}{\sigma_r^2 M_r}\right)\right\}$$

Any other cue can be fused within the particle filter with the same rule. One way of combining the information provided by multiple cameras is to incorporate the likelihoods for each image in the exact same manner [4]. In our system, we additionally use 3D information which can be computed explicitly by knowing the stereo calibration. This separate cue is then combined with the other likelihoods with the same method, as will be described in Section III.

III. Multiple Cues in the proposed System

In this section, we want to introduce the cues our system is based on. Instead of the commonly used region-based likelihood function p_r, as introduced in Equation (1), we incorporate the result of foreground segmentation in a more efficient way, as will be introduced in Section III-A. In Section III-B we will present the results of studies regarding the effectivity of the introduced cues, leading to a new likelihood function. As already mentioned, we use the benefits of a stereo system in an additional explicit way, as will be introduced in III-C. The final combined likelihood function is presented in Section III-D.

A. Edge Filtering using Foreground Segmentation

When looking deeper into the region-based likelihood function p_r, one can state two separate abilities:

- Leading to a faster convergence of the particle filter
- Compensating the failure of the edge-based likelihood function in cluttered backgrounds

The first property is discussed in detail in Section III-B, and an efficient alternative is presented. The second property can be implemented explicitly by using the result of foreground segmentation directly to generate a filtered edge map, containing only foreground edge pixels. In general, there are two possibilities:

- Filtering the gradient image by masking out background pixels with the segmentation result
- Calculating gradients on the segmentation result

While the first alternative preserves more details in the image, the second alternative computes a sharper silhouette. Furthermore, in the second case, gradient computation can be optimized for binarized input images, which is why we currently use this approach. As explained in Section II-B, the only commonly used foreground segmentation technique is background subtraction, which we do not intend to use, since the robot head can potentially move. It has to be mentioned that taking into account that the robot head can move is not a burden, but there are several benefits of using an active head, which will be discussed in Section VI. As an alternative to using background subtraction, we are using a solid colored shirt, which allows us to perform tests practically anywhere in our lab. Since foreground segmentation is performed in almost any markerless human motion capture system, we do not restrict ourselves compared to other approaches, but only trade in the restriction of wearing a colored shirt for the need of having a completely static setup. Experiments have shown that the segmentation result using a colored shirt leads to a slightly smoother output of the human motion capture system compared to background subtraction. However, if the shirt color is not to be used, background subtraction still results in a robust system.

B. Cue Comparison and Distance Likelihood Function

In order to understand the benefits and drawbacks of each likelihood function and thus getting a feeling of what a likelihood function can accomplish and what not, it is helpful to measure their effectivity in a simple one-dimensional example. The experiment we have used in simulation is tracking a square of fixed size in 2D, which can be further simplified to tracking the intersection of a square with a straight line along the straight line i.e. in one dimension.

The model of the square to be tracked is defined by the midpoint (x, y) and the edge length k, where y and k are constant and x is the one dimensional configuration to be predicted. In [10], we have compared three different likelihood functions separately: the gradient-based cue p_g, the region-based cue p_r, and a third cue p_d, which is based on the Euclidian distance:

$$p_d(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_d^2}|f(\mathbf{s})-\mathbf{c}|^2\right\}$$

where $\mathbf{c}$ is an arbitrary dimensional vector which has been calculated previously on the base of the observations $\mathbf{z}$, and $f : R^{\dim(\mathbf{s})} \rightarrow R^{\dim(\mathbf{c})}$ is a transformation mapping a configuration $\mathbf{s}$ to the vector space of $\mathbf{c}$. In our example, $\mathbf{c}$ denotes the midpoint of the square in the observation $\mathbf{z}$, $\dim(\mathbf{s}) = \dim(\mathbf{c}) = 1$, and $f(\mathbf{s}) = \mathbf{s}$. For efficiency considerations, we have used the squared Euclidian distance, practically resulting in the SSD. Evidently, in this simple case, there is no need to use a particle filter for tracking, if the configuration to be predicted $\mathbf{c}$ can be determined directly. However, in this example, we want to show the characteristic properties of the likelihood function p_d, in order to describe the performance in the final likelihood function of the human motion capture system, presented in the sections III-C and

III-D. In the update step of the particle filter, we applied Gaussian noise only, with an amplification factor of $\omega = 3$. The task was to find a static square with $k = 70$, based on the pixel data at the intersection of the square with the x-axis. The number of iterations needed depending on the initial distance to the goal is given in Figure 1 for each likelihood function. While with starting points in a close neighborhood of the goal the gradient cue leads to the fastest convergence, the region cue and the distance cue converge faster the farther the starting point is away from the goal. The results of the comparison of the cues is explained in detail in [10].

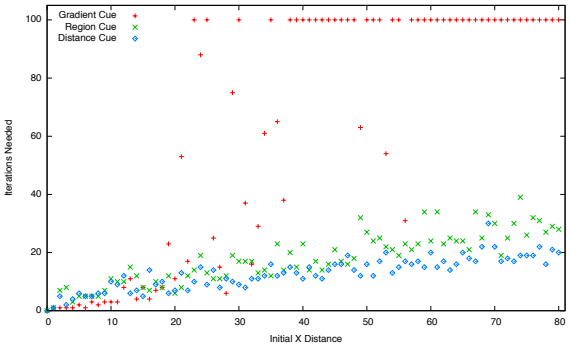

Fig. 1. Comparison of iteration numbers: an iteration number of 100 indicates that the goal was not found

As a conclusion, one can state that whenever possible to determine a discrete point directly, it is the best choice to use the likelihood function p_d rather than p_r. A second drawback of the region cue is explained in Section III-D. While it is not possible to do a successful tracking without the edge cue – especially when scaling has to be taken into account – it is also not possible to rely on the edge cue only. The higher the dimensionality of search space is, the more drastic the lack of a sufficient number of particles becomes. Thus, in the case of human motion capture with dimensions of 14 and greater, the configurations will never perfectly match the image observations. Note, that the simulated experiment examined a static case. In the dynamic case, the robustness of the tracker is always related to the frame rate at which images are captured and processed, and to the speed of the subject's movements. In the next section, we show how the likelihood function p_d is incorporated into our system using 3D points, leading to a significant implicit reduction of the search space.

C. Using Stereo Information

There are various ways to use stereo information in a vision system. One possibility is to calculate depth maps, however, the quality of depth maps is in general not sufficient and only rather rough information can be derived from them. Another option in a particle filter framework is to project the model into both the left and the right image and evaluate the likelihood function for both images and multiply the the resulting likelihoods, as already mentioned in Section II-C. This approach can be described as *implicit stereo*. A third alternative is to determine correspondences for specific features in the image pair and calculate the 3D position for each match explicitly by triangulation.

In the proposed system, we use both implicit stereo and stereo triangulation. As features we use the hands and the head, which are segmented by color and matched in a pre-processing step. Thus, the hands and the head can be understood as three natural markers. The image processing line for determining the positions of the hands and the head in the input image is described in Section IV.

There are two alternatives to use the likelihood function p_d together with skin color blobs: apply p_d in 2D for each image separately and let the 3D position be calculated implicitly by the particle filter, or apply p_d in 3D to the triangulated 3D positions of the matched skin color blobs. We have experienced that the first approach does not lead to a robust acquisition of 3D information. This fact is not surprising, since in a high dimensional space the mismatch between the number of particles used and the size of the search space is more drastic. This leads, together with the fact that the prediction result of the likelihood function p_d is noisy within an area of 1-2 pixels already in a very simple experiment, to a considerable error in the implicit stereo calculation in the real scenario. For this reason we apply p_d in 3D to the triangulation result of matched skin color blobs. By doing this, the particle filter is forced to always move the peak of the probability density function toward configurations in which the positions of the hands and the head from the model are very close to the real 3D positions, which have been determined on the base of the image observations.

D. Final Likelihood Function

In the final likelihood function, we use two different components: the edge cue based on the likelihood function p_g, and the distance cue based on the likelihood function p_d, as explained in the sections III-B and III-C. The region cue is left out for two reasons: it reduces the efficiency of the system significantly, and it can diminish the accuracy of the estimation. The reason for this is that the region cue is less precise than the edge cue; in many cases it evaluates wrong configurations with a similar or even equal likelihood as it does for the proper configuration, as illustrated in Figure 2.

Fig. 2. Illustration of a wrong and a proper configuration, which are assigned the same likelihood by the region cue

We have experienced that when leaving out the square in Equation (II-A), i.e. calculating the Sum of Absolute Differences (SAD) instead of the Sum Of Squared Differences (SSD), the quality of the results remains the same for our application. In this special case one can optimize p_g further,

resulting in:

$$p'_g(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_g^2}(1-\frac{1}{M_g}\sum_{m=1}^{M_g} g_m)\right\}$$

For a system intended for real-time application, we have decided to replace the region-based cue based on p_r completely by the distance cue based on p_d. In order to formulate the distance cue, first the function $d_i(\mathbf{s},\mathbf{c})$ is defined as:

$$d_i(\mathbf{s},\mathbf{c}) := \begin{cases} |f_i(\mathbf{s})-\mathbf{c}|^2 & : \quad \mathbf{c} \neq \mathbf{0} \\ 0 & : \quad \text{otherwise} \end{cases}$$

where $n := \dim(\mathbf{s})$ is the number of DOF of the human modal, $\dim(\mathbf{c}) = 3$, $i \in \{1,2,3\}$ to indicate the function for the left hand, right hand or the head. The transformation $f_i : R^n \rightarrow R^3$ transforms the n-dimensional configuration of the human model into the 3D position of the left hand, right hand or head respectively, using the forward kinematics of the human model. The likelihood function for the distance cue is then formulated as:

$$p'_d(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_d^2}(d_1(\mathbf{s},\mathbf{c_1})+d_2(\mathbf{s},\mathbf{c_2})+d_3(\mathbf{s},\mathbf{c_3}))\right\}$$

where the vectors $\mathbf{c_i}$ are computed on the base of the image observations $\mathbf{z}$ using skin color segmentation and stereo triangulation, as explained in Section III-C. If the position of a hand or the head can not be determined because of occlusions or any other disturbance, the corresponding vector $\mathbf{c_i}$ is set to the zero vector. Note that this does not falsify the resulting probability density function in any way. Since all likelihoods of a generation k are independent from the likelihoods calculated for any previous generation, the distribution for each generation is also independent. Thus, it does not make any difference that in the last image pair one $\mathbf{c_i}$ was present, and in the next image pair it is not. The final likelihood function is the product of p'_g and p'_d:

$$p(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_d^2}\sum_{i=1}^{3} d_i(\mathbf{s},\mathbf{c_i}) + \frac{1}{\sigma_g^2}(1-\frac{1}{M_g}\sum_{m=1}^{M_g} g_m)\right)\right\}$$

IV. Image Processing Pipeline and Tracking of Hands and Head

The image processing pipeline transforms each image of the stereo pair into a skin color map and a gradient map, which are then used by the likelihood function presented in Section $III-D$. In Figure 3, the pipeline is shown for one image; in the system, the pipeline is applied twice: once for each image of the stereo pair. After the input images are smoothed with a 3×3 Gaussian kernel, the HSV image is computed. The HSV image is then filtered twice, once for skin color segmentation and once for foreground segmentation by segmenting the shirt color. A simple combination of a 2×1 and a 1×2 gradient operator is applied to the segmented foreground image, which is sufficient and the most efficient for a binarized image. Finally, a gradient pixel map is generated by blurring the gradient image, as done in [4].

Currently, the hands and the head are segmented using a fixed interval color model in HSV color space. Similar to the idea presented in [16], each color blob is tracked with a linear motion model, which makes it possible to track through images in which blobs are occluded or blobs fall together. In these images, the state of the affected blobs from the last image is propagated to the current image, predicting their current position by using the linear motion model. The resulting color blobs are matched between the left and right image, taking into account their size, the ratio between the height and width of the bounding box, and the epipolar geometry. By doing this, false regions in the background can be discarded easily. Finally, the centroids of matched regions are triangulated using the parameters of the calibrated stereo setup.

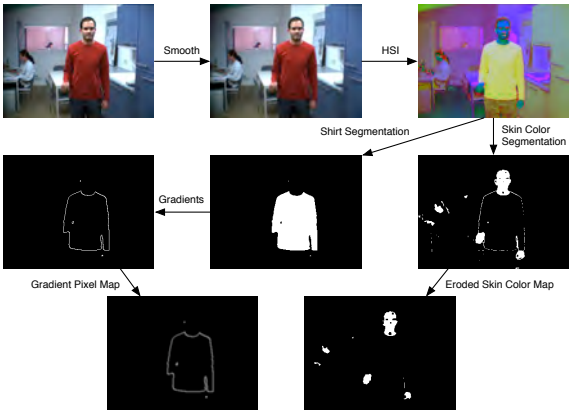

Fig. 3. Visualization of the image processing line

All image processing routines and mathematical computations are implemented using the *Integrating Vision Toolkit* (IVT) [17]. It is an Open Source vision library, offering a clean interface to image devices and camera calibration, and a variety of filters, segmentation methods, stereo routines, mathematical function and data structures.

V. Experimental Results

The experiments being presented in this section were performed on the humanoid robot ARMAR. In the robot head, two Dragonfly cameras are positioned at a distance of approximately eleven centimeters. As input for the image processing line, we used a resolution of 320×240, captured at a frame rate of 25 Hz. The particle filter was run with a set of $N = 1000$ particles. The computation times for one image pair, processed on a 3 GHz CPU, are listed in Table I. As one can see, the processing rate of the system is 15 Hz, which is not yet real-time for an image sequence captured at 25 Hz, but very close. Of course, when moving slowly, a processing rate of 15 Hz is sufficient.

In Figure 4, six screenshots are shown which demonstrate how the system automatically initializes itself. No initial configuration is given; it autonomously finds the only possible configuration matching the observations. Figure 5 shows four screenshots of the same video sequence, showing the performance of the human motion capture system tracking a punch with the left hand. We have also run successful experiments with a half a minute sequence with a resolution of 640×480 captured at 57 Hz.

	Time [ms]
Image Processing Line	14
1000 Forward Kinematics and Projection	23
1000 Evaluations of Likelihood Function	29
Total	**66**

TABLE I

PROCESSING TIMES WITH $N = 1000$ PARTICLES ON A 3 GHZ CPU USING A 320×240 STEREO INPUT IMAGE SEQUENCE

Fig. 4. Screenshots showing automatic initialization

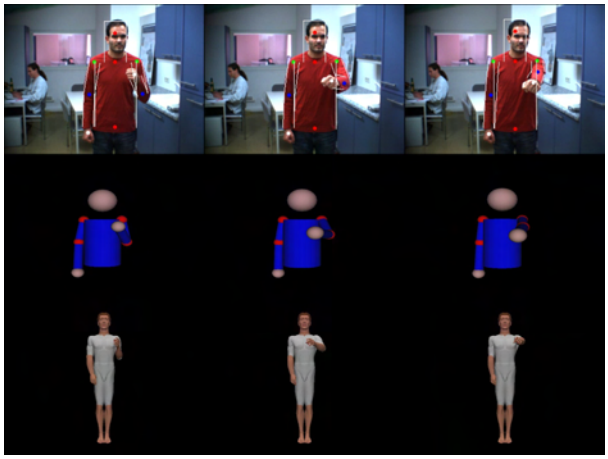

Fig. 5. Screenshots showing tracking performance. Top: input images of the left camera with projected estimation. Middle: Visualization with a simple 3D model. Bottom: Visualization by mapping to a realistic human model

VI. CONCLUSION

We have presented an image-based markerless human motion capture system for application on a humanoid robot. The system is capable of computing motion trajectories in the configuration space of a human body. We presented our strategy for fusing multiple cues within the particle filter. The proposed strategy is supported by the results of a study examining the properties of commonly used image cues and the newly introduced distance cue. We showed that by using the 3-D distance cue we could find optimal configuration withs less particles than standard approaches. This implicit reduction of the search space allows us to capture human motion with a particle filter using as few as 1000 particles, which results in a processing rate of 15 Hz on a 3 GHz CPU for a 320×240 stereo input video stream.

In the near future, we plan to extend the system by incorporating the legs and feet into the human model. Furthermore, we intend to make use of the active head to follow the human subject, thus keeping her/him in the robot's field of view. This would not be possible with a static head.

ACKNOWLEDGMENT

The work described in this paper was partially conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) and funded by the European Commission and the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft). We also would like to thank Gordon Cheng of ATR/JST for his comments and suggestions.

REFERENCES

[1] "Vicon Peak," http://www.vicon.com.
[2] F. Caillette and T. Howard, "Real-Time Markerless Human Body Tracking with Multi-View 3-D Voxel Reconstruction," in *British Machine Vision Conference*, vol. 2, Kingston, UK, 2004, pp. 597–606.
[3] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human Body Model Acquisition and Tracking using Voxel Data," *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, 2003.
[4] J. Deutscher, A. Blake, and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering," in *Computer Vision and Pattern Recognition (CVPR)*, Hilton Head, USA, 2000, pp. 2126–2133.
[5] S. Knoop, S. Vacek, and R. Dillmann, "Modeling Joint Constraints for an Articulated 3D Human Body Model with Artificial Correspondences in ICP," in *International Conference on Humanoid Robots (Humanoids)*, Tsukuba, Japan, 2005.
[6] H. Sidenbladh, "Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences," Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 2001.
[7] K. Rohr, "Human movement analysis based on explicit motion models," *Motion-Based Recognition*, pp. 171–198, 1997.
[8] D. Gavrila and L. Davis, "3-D model-based tracking of humans in action: a multi-view approach," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, USA, 1996, pp. pp. 73–80.
[9] P. Azad, A. Ude, R. Dillmann, and G. Cheng, "A Full Body Human Motion Capture System using Particle Filtering and On-The-Fly Edge Detection," in *International Conference on Humanoid Robots (Humanoids)*, Santa Monica, USA, 2004.
[10] P. Azad, A. Ude, T. Asfour, G. Cheng, and R. Dillmann, "Image-based Markerless 3D Human Motion Capture using Multiple Cues," in *International Workshop on Vision Based Human-Robot Interaction*, Palermo, Italy, 2006.
[11] A. Blake and M. Isard, *Active Contours*. Springer, 1998.
[12] J. MacCormick, "Probabilistic models and stochastic algorithms for visual tracking," Ph.D. dissertation, University of Oxford, UK, 2000.
[13] J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *European Conference Computer Vision (ECCV)*, Dublin, Ireland, 2000, pp. 3–19.
[14] J. Deutscher, A. Davison, and I. Reid, "Automatic Partitioning of High Dimensional Search Spaces associated with Articulated Body Motion Capture," in *Computer Vision and Pattern Recognition (CVPR)*, Kauai, USA, 2001, pp. 669–676.
[15] K. Wong and M. Spetsakis, "Motion Segmentation and Tracking," in *International Conference on Vision Interface*, Calgary, Canada, 2002, pp. 80–87.
[16] A. A. Argyros and M. I. Lourakis, "Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera," in *European Conference on Computer Vision (ECCV)*, vol. 3, Prague, Czech Republic, 2004, pp. 368–379.
[17] P. Azad, "Integrating Vision Toolkit," http://ivt.sourceforge.net.

Gradient-Enhanced Particle Filter for Vision-Based Motion Capture

Daniel Grest and Volker Krüger

Aalborg University Copenhagen, Denmark
Computer Vision and Machine Intelligence Lab
{dag,vok}@cvmi.aau.dk

Abstract. Tracking of rigid and articulated objects is usually addressed within a particle filter framework or by correspondence based gradient descent methods. We combine both methods, such that (a) the correspondence based estimation gains the advantage of the particle filter and becomes able to follow multiple hypotheses while (b) the particle filter becomes able to propagate the particles in a better manner and thus gets by with a smaller number of particles. Results on noisy synthetic depth data show that the new method is able to track motion correctly where the correspondence based method fails. Further experiments with real-world stereo data underline the advantages of our coupled method.

1 Introduction

Motion tracking and human pose estimation are important applications in motion analysis for sports and medical purposes. Motion capture products used in the film industry or for computer games are usually marker based to achieve high quality and fast processing.

Marker-less motion capture approaches often rely on gradient based methods [13, 3, 19, 7, 10, 18]. These methods estimate the parameters of a human body model by minimizing differences between model and some kind of observations, e.g. depth data from stereo, visual hulls or silhouettes. Necessary for minimization are correspondences between model and observed data. The main problem of these correspondence based optimization methods is, that they often get stuck in wrong local minima. From this wrong estimated pose they can usually not recover.

Other approaches like particle filters [5, 11] try to approximate the probability distribution in the state space by a large number of particles (poses) and are therefore unlikely to get stuck in local minima, because they can follow and test a large number of hypotheses. However, to be sure that the "interesting" region (*typical set*) of a high-dimensional state space is properly sampled, a large amount of particles is usually necessary [15, 2].

To take the advantages of both approaches, we combine a particle filter based approach [15, 11, 6] with correspondence based gradient estimation.

The effect can be interpreted in two ways: (1) Following the local gradient within the particle filter allows to find minima (including the global minimum) with less particles and (2) enhancing the gradient descent method with multiple hypothesis helps to avoid to get stuck in local minima.

The key to our approach and the main difference to a particle filter like CONDENSATION [11] is the gradient descent for each particle and the merging of particles. Particles, which are close to each other after the gradient descent, are merged into a single particle. Then, the idea is to re-distribute (propagate) the particles in a way that takes into account the local shape of the likelihood function. That way the number of particles can be greatly reduced while maintaining the ability to follow multiple hypotheses.

We will discuss our new approach in the context of marker less motion tracking from a single stereo view with two cameras. There is a wide variety of stereo algorithms available differing in quality and processing time. Commercial stereo cameras calculate depth data on chip using a simple algorithm [20] in real-time and keep the CPU free. Other more sophisticated algorithms need up to multiple minutes per image pair [8].

We will at first discuss relevant work within the field of motion tracking. Then, introduce our body model and the motion parameterization, which are used in the correspondence based optimization. The next section briefly explains important aspects of particle filter tracking methods, which are necessary for the combination. Then, results on synthetic data and real motion sequences are given, that show the advantages of the combined motion tracking. The last section concludes the paper with a short discussion of the presented achievements.

2 Related work

Motion tracking of the human body is addressed in the literature with different methods. A recent and extensive survey of vision based human motion tracking can be found in [16]. Approaches relevant to this work arise from different directions depending on the kind of input data, e.g. depth data or images, and the number of cameras. Visual hull approaches have shown to give very accurate results [13, 3]. They build the visual hull of the person from segmented images of multiple cameras and then fit a template model to the 3D hull. Usually some kind of gradient based optimization is utilized to estimate the motion parameters of the model. Fitting the template model directly to segmented images is another possibility as done in [19], where it was shown, that the marker-less approach has an accuracy similar to marker based tracking.

Similar to the visual hull approach is [12], where multiple stereo cameras observe the motion of a person. The resulting 3D points are then used with an Extended Kalman Filter to estimate the upper body motion.

When only two or less cameras are used for motion tracking, particle filters [11] or particle filtering methods are utilized [5]. Their advantage is, that multiple tracking hypotheses can be followed, because a large number of particles approximates the posterior probability of the system's state. Therefore, the tracking is not prone to get stuck in local minima and multiple hypotheses can be followed. This ability to track multiple hypotheses is very useful, because body parts can become occluded, if only a single viewpoint is used. Then, the occluded motion has to be 'guessed' in order to track successfully, when the occluded body part becomes visible again. However, if full body motion with 30 DOF is to be estimated the number of particles can approximate the posterior distribution only within a small region in the state space. Therefore, once again it is likely to face the problem of local minima. Otherwise the number of parti-

cles has to be increased, which increases computation time. For 30 DOF the necessary amount of particles can result in a computation time of up to multiple hours per image frame of a video sequence. However a parallel processing of particles is possible. Our approach decreases the amount of necessary particles significantly and allows such processing in reasonable time.

Motion tracking from a single stereo view as in this work has been addressed before in [4], where the human is modeled with 6 cylinders and motion can be roughly tracked with 10Hz. The authors use projective methods, which are inferior to direct methods as they state themselves in [4], but are easier to implement and require less computation time (no comparison is made). In [18] a direct approach is taken, where a human model consisting of spheres (meta-balls) is fitted to stereo data silhouettes from a single view. Due to the high number of estimated parameters, the method is not real-time capable. Both methods use a correspondence based gradient descent method, which requires manual initialization and can get stuck in local minima.

In [17], 3D body tracking is done using particle propagation. The Zakai Equation is applied to model the propagation of probability density function (pdf) over time in order to reduce the number of particles.

In a previous work [10] we showed that our direct approach is able to track upper arm motion with 5Hz and can therefore compete with the projective method of [4] according to processing time. Here we present a combination with a particle filter that allows us to track very noisy arm motion and complex full body motion of the whole body from stereo data alone, even though body parts are temporarily occluded.

A nice review on Monte Carlo-based techniques can be found in [15]. Classical papers about particle filtering are Condensation[5, 11], Sequential Importance Sampling [6] and sequential Monte Carlo [14].

3 Body Model and Motion Parameterization

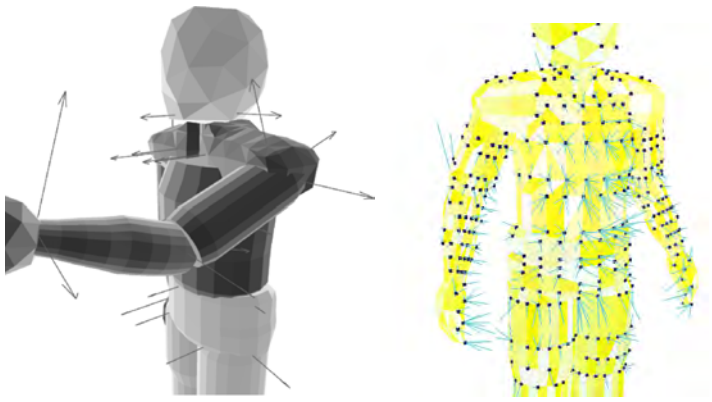

Fig. 1. Left: The body model with rotation axes shown as arrows. Right: The difference (small blue lines) between observed depth point and nearest model point (black boxes) is minimized.

The motion capabilities of the human model is based on the MPEG4 standard, with up to 180 DOF. An example model is shown in Fig. (1) left. The MPEG4 description allows to exchange body models easily and to re-animate other models with the captured motion data. The model for a specific person is obtained by silhouette fitting of a template model as described in [9].

The MPEG4 body model is a combination of kinematic chains. The motion of a point, e.g. on the hand, may therefore be expressed as a concatenation of rotations [10]. As the rotation axes are known, e.g. the flexion of the elbow, the rotation has only one degree of freedom (DOF), i.e. the angle around that axis. In addition to the joint angles, there are 6 DOF for the position and orientation of the object within the global world coordinate frame. For an articulated object with p joints we describe the transformation of the point $\boldsymbol{p}$ within the chain [10] as

$$\begin{aligned}\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p}) = &(\theta_x, \theta_y, \theta_z)^T + \\ &(R_x(\theta_\alpha) \circ R_y(\theta_\beta) \circ R_z(\theta_\gamma) \circ R_{\omega_1, q_1}(\theta_1) \circ \cdots \\ &\cdots \circ R_{\omega_p, q_p}(\theta_p))\boldsymbol{p}\end{aligned}$$

where $(\theta_x, \theta_y, \theta_z)^T$ is the global translation, R_x, R_y, R_z are the rotations around the global x, y, z-axes with Euler angles α, β, γ and $R_{\boldsymbol{\omega},\boldsymbol{q}}(\theta_i), i \in \{1..p\}$ denotes the rotation around the known axis with angle θ_i. The axis is described by the normal vector $\boldsymbol{\omega_i}$ and a point $\boldsymbol{q_i}$ on the axis.

Eq. 1 gives the position of a point $\boldsymbol{p}$ on a specific segment of the body (e.g. the hand) with respect to joint angles $\boldsymbol{\theta}$ and an initial body pose.

If the current pose is $\boldsymbol{\theta}_t$ and only relative motion is estimated the resulting Jacobian is:

$$J = \begin{bmatrix} 1 & 0 & 0 & \frac{\partial m_x}{\partial \theta_\alpha} & \frac{\partial m_x}{\partial \theta_\beta} & \frac{\partial m_x}{\partial \theta_\gamma} & \frac{\partial m_x}{\partial \theta_1} & .. & \frac{\partial m_x}{\partial \theta_p} \\ 0 & 1 & 0 & \frac{\partial m_y}{\partial \theta_\alpha} & \frac{\partial m_y}{\partial \theta_\beta} & \frac{\partial m_y}{\partial \theta_\gamma} & \frac{\partial m_y}{\partial \theta_1} & .. & \frac{\partial m_y}{\partial \theta_p} \\ 0 & 0 & 1 & \frac{\partial m_z}{\partial \theta_\alpha} & \frac{\partial m_z}{\partial \theta_\beta} & \frac{\partial m_z}{\partial \theta_\gamma} & \frac{\partial m_z}{\partial \theta_1} & .. & \frac{\partial m_z}{\partial \theta_p} \end{bmatrix} \tag{1}$$

The derivatives at zero are:

$$\left.\frac{\partial \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p})}{\partial \theta_j}\right|_{\boldsymbol{\theta}=\boldsymbol{0}} = \boldsymbol{\omega_j} \times (\boldsymbol{p} - \boldsymbol{q_j}) \tag{2}$$

where $j \in \{1, .., p\}$ and $\boldsymbol{q}_j$ is an arbitrary point on the rotation axis. The simplified derivative at zero is valid, if relative transforms in each iteration step of the *Nonlinear Least Squares* are calculated and if all axes and corresponding point pairs are given in world coordinates.

4 Gradient Enhanced Particle Filtering

Because our method combines the advantages of gradient based optimization methods with particle filtering, we give now a brief overview of important aspects of both. Then, we present the combined algorithm and discuss the main differences to particle filters.

4.1 Correspondence Based Pose Estimation

Correspondence based methods for pose estimation of articulated objects minimize an error function with respect to motion parameters. The human body can be modeled as an articulated object, consisting of multiple kinematic chains.

It is common to assume that the shape and size of the body model is known for the observed person, such that the minimization is only with respect to joint angles and the global transform. In that case, the kinematic chain simplifies to a chain of rotations around arbitrary axes in space Given here is a short description of the estimation algorithm, for more details see [10].

Estimating the motion of the human body from given 3D-3D correspondences $(\boldsymbol{p_i}, \tilde{\boldsymbol{p}}_{\boldsymbol{i}})$ is done here by solving a Nonlinear Least Squares Problem. The minimization yields the joint angles and the global orientation and position. For n correspondences the minimization problem is given as:

$$\min_{\boldsymbol{\theta}} \sum_{i}^{n} |\boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}}|^2 \quad . \tag{3}$$

To find the minimizer with the iterative *Gauss-Newton* method the Jacobian of the residual functions, Eq. (1), is necessary.

The points $\boldsymbol{p_i}$ and $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}$ form a correspondence. For each observed point $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}$ the closest point $\boldsymbol{p_i}$ on the model is sought. Therefore, different observed points can have the same corresponding model point. This is shown in Fig. 1 right, where each correspondence is shown as a small blue line and the model points are drawn as black boxes.

The minimization problem is solved with the dampened Gauss-Newton method[1], which is similar to the Levenberg-Marquardt[1] method. Dampening ensures that the parameter change does not increase infinitely, if the determinant of the Gram matrix $J^T J$ is close to zero, which can happen when a body part is largely occluded. The solution is found by solving iteratively:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - (J^T J + \lambda I)^{-1} J^T \boldsymbol{r}(\boldsymbol{\theta}_t) \tag{4}$$

where the Jacobian J is given in equation (1), I is the identity matrix, λ is the dampening value (set to 0.1) and $\boldsymbol{r}(\boldsymbol{\theta}_t)$ is the vector with current residuals. For each point there are three residuals, one for each component (x, y, z):

$$\boldsymbol{r_i}(\boldsymbol{\theta}_t) = \boldsymbol{m}(\boldsymbol{\theta}, \boldsymbol{p_i}) - \tilde{\boldsymbol{p}}_{\boldsymbol{i}} \tag{5}$$

and $\boldsymbol{r_i} = (r_{ix}, r_{iy}, r_{iz})$.

The optimization consists of two loops: The Gauss-Newton method (GN) loops until it converges on the given set of point correspondences. Because the correspondences are not always correct, new correspondences are calculated again after convergence of the GN. Then, GN starts anew with the improved set of correspondences. This Iterative Closest Point (ICP) method [1] can be repeated until convergence. However, the gain in more then 3 ICP iterations is very small, therefore we usually apply only 2 or 3 ICP optimizations. We will use in the following the term "gradient tracking" in order to refer to this technique.

It is important to note, that the Gauss-Newton optimization is more efficient than a standard Gradient Descent and requires less control parameters. However we will refer to it as "gradient tracking", because both methods rely on the gradient.

4.2 Particle Filter

The new method borrows some ideas from particle filter methods like CONDENSATION [11]. Particle filter approaches aim at estimating the posterior probability distribution of a system state $\boldsymbol{z}_t$ at time t, the person's current pose in our case, from observations $I_1, \ldots, I_t$:

$$\begin{aligned} & p(\boldsymbol{z}_t | I_1, I_2, \ldots, I_t) \equiv p_t(\boldsymbol{z}_t) \\ & = \int_{\boldsymbol{z}_{t-1}} p(I_t|\boldsymbol{z}_t) p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) p_{t-1}(\boldsymbol{z}_{t-1}) \ . \end{aligned} \tag{6}$$

In this equation the state space is randomly sampled according to $p_{t-1}(\boldsymbol{z}_{t-1})$ and propagated according to a motion/diffusion model $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$. A likelihood probability $p(I_t|\boldsymbol{z}_t)$ for each particle is calculated, which reflects how good the observations fit to the hypothesis (the position of the particle in the state space). The posterior probability is then approximated by the weight and density of particles within the state space.

The major problem with these approaches is, that the number of necessary particles usually needs to reflect the dimensionality of the state space[15, 2]. If full body motion with 30 DOF is to be estimated the particles can usually approximate the posterior distribution only within a small region in the state space.

If depth data is the only input, calculation of the likelihood requires computation of differences between observed points and model surface. One way to compute these differences is a nearest neighbor search as described in the previous section. This search is, however, expensive in terms of computation time. Therefore, methods are desirable, which reduce the number of particles and allow to distribute the reduced set of particles in the most important regions of the state space.

4.3 Combination

In order to get by with a smaller number of particles, we apply the gradient tracking (section 4.1) to each particle. Since similar particles move to the *same* optimum, they can be merged with an appropriate adaption of the particle weight. The state space posterior probability is approximated in a particle filter by the particle weights and their spatial density. A possible weight adaption is the average of merged particle weights. However, all merged particles are nearly at the same position in the state space and therefore have nearly the same weight (likelihood). As a result it is sufficient to assign them the same weight. Another possible weight adaption is the addition of weights. However, these would favor large flat valleys in the posterior probability surface, because all particles within in this valley will descent towards the same minimum. This will also lead to a clustering of particles at specific positions, which is not desired, because we want to track as many hypotheses as possible. If the merged particles are redistributed in the next time step only according to a fixed motion model and diffusion model, it is likely that they end up in the same locally convex region (valley) and again merge at the same position in state space. Each valley can be understood as one likely pose hypothesis. It is desirable to track as many hypotheses as possible with a fixed amount of particles. To increase the number of tracked hypotheses and decrease

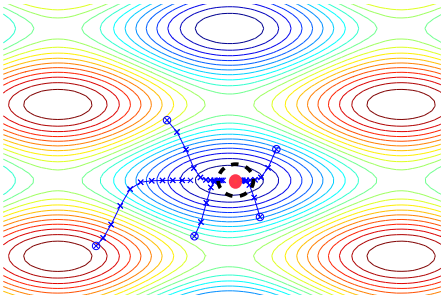

Fig. 2. Principle of the combined method. Particles whose position is within a specific area after optimization (black dashed circle) are merged into one particle (red circle).

the amount of particles per valley, the particles need to be redistributed at the next time step of Eq. (6) according tho the size and shape of their valley. This can be understood as enhancing our rather simple motion model (fixed velocity) to include the shape of the local probability surface.

In order to achieve this, we merge all particles after optimization, which are close to each other according to some distance d. In detail, if we use N particles $\boldsymbol{z}_1^{t-}, \dots \boldsymbol{z}_N^{t-}$ in our particle filter, then we have after the merging $M < N$ meaningful particles $\boldsymbol{z}_1^t, \dots \boldsymbol{z}_M^t$ left, and $N - M$ particles were merged into the remaining M particles.

Then, in order to distribute the particles in the next time step optimally, we estimate the size of the locally convex region by computing the covariance Σ_i of all those particles. Let $\boldsymbol{z}_{i_1}^{t-}, \dots \boldsymbol{z}_{i_k}^{t-}$ be the particles that merged together into the particle $\boldsymbol{z}_i^t$. The $^-$ at the top denotes the particles *before* their gradient descent and merging, t denotes the time step. The final particle $\boldsymbol{z}_i^t$ (without the $^-$) after gradient descent and merging (red circle in Fig.2) is the one with highest likelihood and is used as the mean for the covariance:

$$\Sigma_i = \frac{1}{K} \sum_{j}^{K} (\boldsymbol{z}_j^{t-} - \boldsymbol{z}_i^t)(\boldsymbol{z}_j^{t-} - \boldsymbol{z}_i^t)^T \tag{7}$$

where K is the amount of particles, which merged into $\boldsymbol{z}_i^t$. It is important to note that the covariance is calculated from the particles *before* the gradient descent and merging. Fig. 2 illustrates the merging. The blue lines show a few steps of the gradient descent for five particles. They are within a certain region (the black dashed circle) after optimization and therefore merged. The idea is to distribute particles in the next frame outside that locally convex region (valley), because otherwise they would end up again at the same position and give no additional information about the state space. Thus, at the next time step $t + 1$ N new particles are drawn from the remaining M particles of time step t according to the prior p_{t+1}. Then, each particle is propagated according to some motion model $f(\boldsymbol{z}^{t+1-})$ with added Gaussian noise. Let the particle $\boldsymbol{z}^{t+1-}$ be spawned off from the particle $\boldsymbol{z}_i^t$. The covariance of the above Gaussian is given by the covariance Σ_i of the original particle $\boldsymbol{z}_i^t$.

Our gradient enhanced particle filter can be summarized as follows: Input to our algorithm are the depth points for the current image frame and an initial pose in the beginning. The steps of the algorithm are for each frame similar to a particle filter method except for the gradient descent and the merging of particles.

1. Only at the first frame: Distribute particles according to an initial distribution in the vicinity of the given initial pose.
2. Draw new particles $\boldsymbol{z}_1^{t-}, \ldots \boldsymbol{z}_N^{t-}$ according to $p_t(\boldsymbol{z}_t)$.
3. Distribute and propagate each particle according to the covariance Σ_i of the original particle and propagate according to a diffusion/propagation model: $p(\boldsymbol{z}^{t+1-}|\boldsymbol{z}^{t-}, \Sigma) = Gauss(f(\boldsymbol{z}^{t-}), \Sigma)$. Here, the motion model consists of a deterministic motion model f plus Gaussian noise, and Σ_i defines the Gaussian covariance matrix.
4. Gradient Descent for each particle $\boldsymbol{z}_i^{t+1-}$ with 2 or 3 ICP optimizations:
   (a) Render model in current pose
   (b) Calculate visible model points
   (c) For each observed point find the closest model point, which makes a correspondence
   (d) Calculate a new pose by minimizing the differences of the correspondences
5. Assign the likelihood, calculated from the residual error.
6. Merge particles $\boldsymbol{z}_{i_1}^{t+1-}, \ldots \boldsymbol{z}_{i_k}^{t+1-}$, which are within a certain distance d to each other into one particle $\boldsymbol{z}_i^{t+1}$.
7. For each particle $\boldsymbol{z}_i^{t+1}$: calculate the covariance matrix Σ_i from all the particles $\boldsymbol{z}_{i_1}^{t+1-}, \ldots \boldsymbol{z}_{i_k}^{t+1-}$ which merged into $\boldsymbol{z}_i^{t+1}$.

The motion model $f(\boldsymbol{z}^{t-1})$ predicts the new pose of the human. In our experiments, we assume constant velocity. The distance d was chosen to be 4 degrees, such that particles are merged only if each single joint angle differs less than 4 degrees to a neighboring particle. The distance check is only applied on joint angles, not on the position of the body model in the world, because it is highly unlikely, that the same pose is estimated at different positions. This way additional control parameters are avoided.

The initial covariance for each particle is chosen to equal the distance d, such that particles are definitely distributed outside the merge area. The covariance is also reduced by 10% each frame. Without this reduction the covariance can increase indefinitely. Especially if only one particle is left in a valley, it is desirable to reduce the covariance, such that it is ensured, that nearby poses are tested.

5 Synthetic data with noise

In order to show the robustness of the new method to noisy measurements we conduct an experiment on depth data generated with openGL on a synthetic sequence. The motion involves four DOF, the elbow flexion and the shoulder flexion abduct and twisting. Three example images out of the 176 frame sequence are shown in Fig. 3. The depth data is calculated from the z-buffer values after rendering the model.

For testing, Gaussian noise with different standard deviations is added to the original depth data. Fig. 4 shows two views on the depth data and the model in starting pose.

Fig. 3. First frame (left). Frame 60 and frame 176 of the synthetic sequence.

Fig. 4. Two views on the depth point cloud with Gaussian noise (deviation 15cm). Right: The standard tracking method loses track after a few frames and gets stuck in the pose shown.

The z-buffer image was sub-sampled in order to generate approximately 10000 depth points. Approximately 750 points are visible on the right arm each frame.

At a deviation rate of 15cm in depth and 1cm in the other directions the normal tracking methods loses track after a few frames and gets stuck in an arbitrary pose as shown in the right of Fig. 4.

The multi-hypotheses tracking with 20 particles is able to track the motion correctly in spite of the heavy noise. The difference to the ground truth is shown in Fig. 5. In the beginning the estimate is far off with 50 degrees however the plot shows, that the tracking recovers from the wrong local minimum. For all hypotheses the minimum difference to the ground truth is taken, because the particle with the largest likelihood does not always give the correct pose. Plotted is the absolute error in degree over the whole sequence. The elbow difference is high around frames where the arm is close to the body as at frame 60 and 105, because the correspondences established with nearest neighbor are then most ambiguous. Where the arm is far away from the body the noise

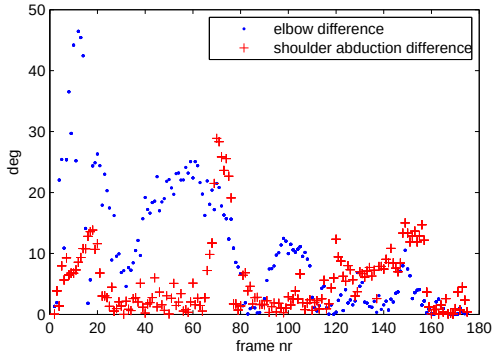

Fig. 5. Difference to ground truth with noise (deviation 15cm). The multi hypotheses tracking is able to track the whole sequence.

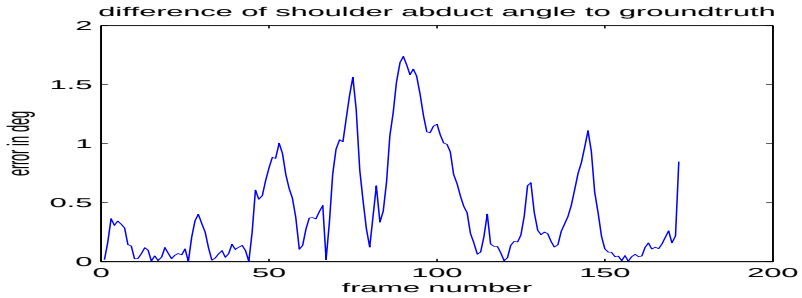

Fig. 6. Difference of the shoulder abduct angle to ground truth without noise.

on the data does not result in so many wrong correspondences. Therefore, the error is decreased. Though the error is still large for most frames, because of the heavy noise, the results show, that the new method can track the motion over the whole sequence.

Without noise the gradient tracking estimates joint angles, whose difference to the ground truth is close to zero error as shown in Fig. 6 for the abduction angle of the shoulder. The error is not zero, because the model surface is approximated by the vertices of the model's triangles. Therefore, the nearest neighbor correspondences are not perfect.

6 Real Data

In order to show the possibilities with our new method, we give further results for a video sequence, which was recorded in a motion capture lab with 8 cameras at 25fps. Two of the cameras were arranged approx. 25 cm next to each other, such that stereo depth estimation can be performed. The stereo algorithm[8] produces dense accurate

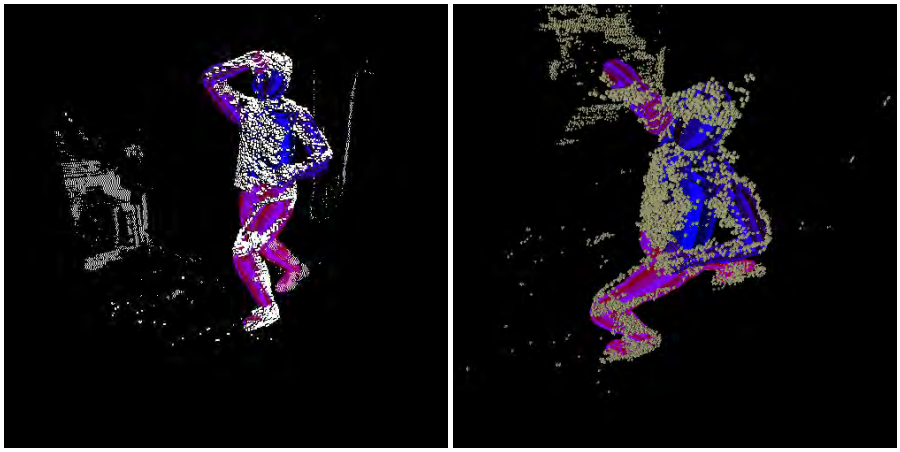

Fig. 7. Two views on the input data. Approx 10000 points are shown as white boxes.

results in non-homogenous regions within approx. one minute computation time per frame. The used effective image size is 512x512. Fig. 7 shows the depth data that is used as input. The only assumption made here is, that no scene objects are within 80cm range of the person. Also knowledge about the floor position was incorporated from camera calibration, which was conducted for the internal parameters with a small checkerboard pattern according to [21]. This calibration also yields the orientation and distance of the stereo cameras. The external parameters (orientation of the floor) were then estimated with a large checkerboard pattern lying on the floor.

The initial pose of the person is provided manually. Estimated are 24 DOF, these are in detail 3 at each shoulder, one elbow angle, 3 at each hip, one angle for each knee, one angle at the ankle and 6 parameters for the global orientation and position.

The estimation time on a 2Ghz intel Core2 Duo (1 CPU) was about 2 seconds per particle and approximately 10000 data points. For 1000 data points and 14 DOF the computation time is about 200ms per particle.

Fig. 8 shows a few frames from the resulting estimation. The multi-hypotheses tracking with 100 particles is able to track the whole sequence of 180 frames (first 3 rows in the Fig.), even though one arm and one leg are almost completely occluded temporarily. Approximately 10000 data points from the complete set of 40000 reliable data points are used per frame, resulting in a Jacobian of size 30000×24.

The gradient tracking method (row 4) is able to track correctly up to 130 frames, but loses track when the person turns and the body parts become occluded (second image last row). The gradient tracking method is lost in that case and is unable to recover.

7 Conclusions

We presented a new method for motion tracking, that combines gradient based optimization from correspondences and motion tracking with particle filters. The combined approach allows to track arm motion in spite of heavy noise, where a normal gradient

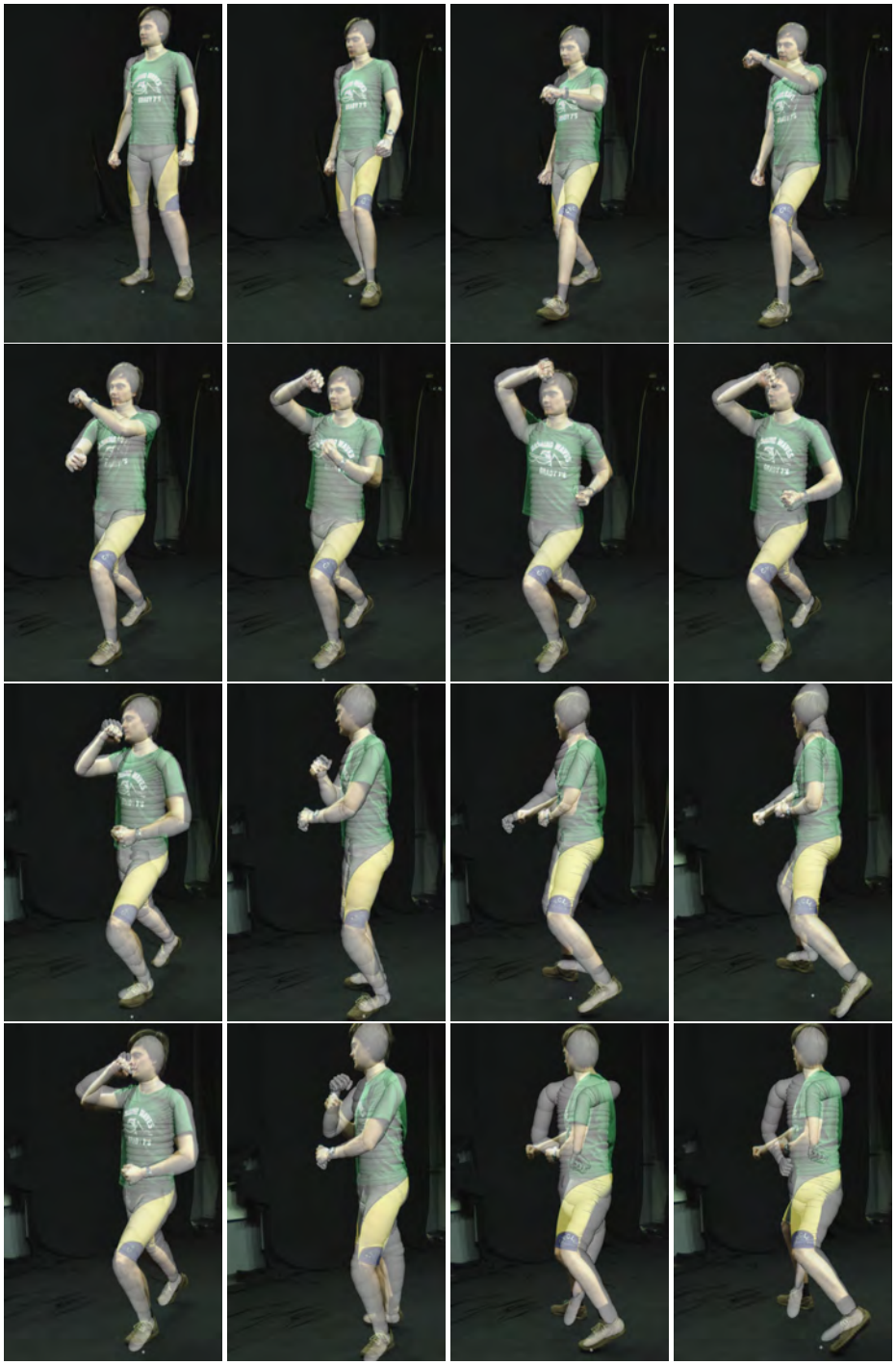

Fig. 8. Estimation results of the new combined method with 24 DOF (first 3 rows). The figure shows the original images together with the projected model in the estimated pose (overlayed in white). The last row shows the estimation results for the ”gradient tracking” for the same images as in the third row. The ”gradient tracking” loses track, because the right arm gets largely occluded, and is unable to recover.

descent method fails. Further results on stereo video sequences showed that motion with 24 DOF can be tracked from a single viewpoint. At frames where the normal tracking gets stuck in local minima and thus loses track, the new method recovers and estimates correct poses.

The main contribution of the new method is the enhanced motion model, which includes the shape of the locally convex regions of the probability surface by estimation of the covariance matrix from merged particles. In that way the number of particles is used more efficiently and allows to track a higher number of hypotheses.

We will exploit in the future further aspects of the new method: (1) the likelihood probability of the particle filter allows to easily include image information into the tracking, as for example motion areas from temporal image differences, (2) increase speed by parallel processing of particles and (3) include motion recognition probabilities into the motion model.

References

1. Edwin K.P. Chong and Stanislaw H. Zak. *An Introduction to Optimization, Second Edition*. Wiley, 2001.
2. T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.
3. E. de Aguiar, C. Theobalt, M. Magnor, and H.-P. Seidel. Reconstructing Human Shape and Motion from Multi-View Video. In *2nd European Conference on Visual Media Production (CVMP)*, London, UK, 2005.
4. D. Demirdjian, T. Ko, and T. Darrell. Constraining Human Body Tracking. In *Proceedings of ICCV*, Nice, France, October 2003.
5. J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *CVPR*, volume 2, 2000.
6. A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. 10:197–209, 2000.
7. Lars Mündermann et al. Validation of a markerless motion capture system for the calculation of lower extremity kinematics. In *Proc. American Society of Biomechanics*, Cleveland, USA, 2005.
8. L. Falkenhagen. Hierarchical block-based disparity estimation considering neighbourhood constraints. In *International workshop on SNHC and 3D Imaging*, 1997.
9. D. Grest, D. Herzog, and R. Koch. Human Model Fitting from Monocular Posture Images. In *VMV*, Nov. 2005.
10. D. Grest, J. Woetzel, and R. Koch. Nonlinear Body Pose Estimation from Depth Images. In *Proc. of DAGM*, Vienna, Sept. 2005.
11. M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *International Journal of Compputer Vision*, 29:5–28, 1998.
12. R. Stiefelhagen J. Ziegler, K. Nickel. Tracking of the Articulated Upper Body on Multi-View Stereo Image Sequences. In *CVPR*, volume 1, pages 774–781, New York, June 2006. IEEE Computer Society.
13. Roland Kehl, Matthieu Bray, and Luc J. Van Gool. Full Body Tracking from Multiple Views Using Stochastic Sampling. In *Proc. CVPR*, pages 129–136, 2005.
14. J.S. Liu and R. Chen. Sequential monte carlo for dynamic systems. 93:1031–1041, 1998.
15. D. MacKay. *Learning in Graphical Models, M.Jordan (ed.)*, chapter Introduction to Monte Carlo Methods, pages 175–204. MIT Press, 1999.

16. T. Moeslund, A. Hilton, and V. Krueger. A Survey of Advances in Vision-based Human Motion Capture and Analysis. *Computer Vision and Image Understanding*, 104(2-3):90–127, 2006.
17. H. Moon, R. Chellappa, and A. Rosenfeld. 3d object tracking using shape-encoded particle propagation. 2001.
18. R. Plänkers and P. Fua. Model-Based Silhouette Extraction for Accurate People Tracking. In *Proc. of ECCV*, pages 325–339. Springer-Verlag, 2002.
19. B. Rosenhahn, U. Kersting, D. Smith, J. Gurney, T. Brox, and R. Klette. A System for Marker-Less Human Motion Estimation . In W. Kropatsch, editor, *DAGM*, Wien, Austria, Sept. 2005.
20. Videre. Videre Design: Stereo On A Chip. www.videredesign.com, 2007.
21. Z. Zhang. Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. In *ICCV*, pages 666–673, Corfu, Greece, 1999.

Dynamic Time Warping for Binocular Hand Tracking and Reconstruction

Javier Romero, Danica Kragic
CAS-CVAP-CSC
KTH, Stockholm, Sweden

Ville Kyrki
Dept. of Information Technology
LUT, Lappeenranta, Finland

Antonis Argyros
Institute of Computer Science
FORTH, Crete, Greece

***Abstract*— Extraction and tracking of hand contours represents an important part of sign language and gesture recognition systems. In robotics, recent developments in imitation learning show the need for complete hand pose estimation such that recognition and evaluation of grasps applied to object can be performed. It has been demonstrated that the use of stereo information improves the depth extraction and provides robustness to occlusion. Once hand contours are identified in each of the stereo images, the points along the contours have to be matched in order to compute their 3D position. The goal of our work is to show how matching and reconstruction of contour points can be performed using Dynamic Time Warping (DTW) for the purpose of 3D hand contour tracking while performing various object manipulation activities. We evaluate the performance of the proposed algorithm and perform comparison with the Iterative Closest Point (ICP) algorithm in situations with a different degree of complexity.**

I. Introduction

In a learning by demonstration context, a robot observes a human performing a task, after which it is supposed to perform the action and thus learn through an imitation process. In order to imitate a human action, the robot needs to retrieve information about of how a specific task was performed – it needs to register the movement of the whole or parts of a human body and an object, and the sequence of different actions performed on the object. Many of the human actions involve object handling, which involves principally the movement of the hands and the arms. The goal of our current work is to develop a real-time stereo based hand tracking system that can be used for full 3D hand pose estimation.

Tracking and reconstructing hands in 3D requires solutions to a number of different problems: hand modeling and detection, temporal association, representation and extraction of 2D data, data association, and matching for 3D reconstruction. Most of the approaches of 3D hand tracking are either model-based or view-based [4]. The former rely on articulated 3D hand models, used to minimize an error function between the model and the observed image data in a sequence of images. This approach requires a model initialization in the first frame which is commonly performed manually. View-based approaches perform pose estimation and classification using a limited number of selected hand poses collected in the training stage [6]. Systems using a combination of the above methods have also been demonstrated [20]. Related to the number of cameras used, both monocular [7] and stereo systems [3] have been used. In the case of the former, assumptions about the size of the hand have to be made to facilitate the 3D reconstruction problem. The stereo-based approaches, on the other hand, require a data association and matching step prior to the reconstruction. Matching can be performed in several ways, depending on whether the reconstruction of the whole surface of the hand is necessary [1], [2].

This work is based on further developments of a system presented in our previous work [3]. In the tracking system, the hands are first identified separately in each of the stereo images and their contours are extracted. This is followed by stereo-based blob matching techniques and shape matching through contour alignment. The particular objective of the work presented here is the development and evaluation of the shape matching and contour alignment step. In the original work [3], Iterative Closest Point (ICP) algorithm and an assumption of the affine motion model were used. However, this approach is not suitable for cases where the inherent assumption of planarity of the hand due to the affine motion model is not valid. This is commonly the case in object grasping and manipulation activities or when a full 3D pose estimation of the hand is required. This paper presents a new approach for contour alignment based on dynamic programming, considered in this paper in the Dynamic Time Warping (DTW) context. The approach is not dependent on the validity of the planarity assumption and an extensive experimental evaluation shows that the performance of the new approach is clearly superior, increasing the robustness to occlusions and relaxing the planarity assumptions. While dynamic programming approaches have been widely used in dense stereo matching, we are not aware of applications in closed contour based matching in stereo.

This paper is organized as follows. In Section II, a short review of related work is given. In Section III, the proposed methods are described. The results of the experimental evaluation are presented in Section IV and the conclusions drawn in Section V.

II. Related work

Pose tracking and 3D reconstruction of hands is a difficult problem: hands are textureless objects, with many degrees of freedom, usually self-occluded or occluded by other objects when object manipulation actions are considered. Since full 3D reconstruction based only on the hand depth estimation is a complex and time-consuming process, view-based approaches have been extensively used in the literature [6]. The tracking problem is then solved through a

classification framework, relating image information directly to the pose space of the hand. Approaches that make use of databases and deformable templates fall in this group [12], [13], [14]. Model-based approaches [6] are based on building an articulated hand model. This model is then used in the tracking process where the incremental change in pose between consecutive images is estimated by minimizing an error function between the model and the observed image data.

For robotic imitation scenarios, where it is expected that a human demonstrates to a robot of how to manipulate a certain object in its workspace, model-based approaches offer a better solution. Apart from not having the need for extensive training, database generation and storage, model-based methods offer a more flexible framework once the mapping between different kinematic chains is needed. Even if a considerable amount of work has been put on the development of humanoid robots during the past few years, robot hands are still simple and do not offer the full complexity of human hands. The simplest form of mapping from a human to a robot hand may then be to just disregard those degrees of freedom that are not articulated on the robot hand. Model-based trackers also offer the capability of continuous pose estimation while view-based methods, if they are not extended with some local fitting step, only provide classification to the nearest dictionary pose.

Within the model-based approaches, we can differentiate the systems based on the extracted features and the methods used to reconstruct the hand model based on these features. Regarding the features, we can differentiate between low level and high level (semantic) features [17]. High level features are desirable since they compress a lot of information about the hand pose in few parameters, and they allow high processing speed for the fitting process. The drawback of high level features is that it is difficult to extract them from images in a general and robust way. One of the most common examples of a high level feature is the position of the fingertips [10], [18].

There are some important differences between monocular and multi-camera systems. The main difference between those systems is that while the monocular systems usually use some predefined hand parameters, such as the length of phalanxes and the size of the palm, to reconstruct the hand in 3D [7], [10], [11], the stereo systems can extract depth information from the 2D data directly, without assumptions about the hand parameters. This makes the multi-camera methods less constrained than the monocular ones at the computing cost of the matching procedure and reconstruction. Multi-camera methods have another important advantage: They are more robust to occlusions. Stereo ”humanoid” heads like the one we use, see Figure 1, can effectively avoid some of the occlusions. If the camera baseline is large and/or the number of cameras is more than two as in [16], even some significant occlusions can be tolerated. This is important in hand tracking since the self-occlusion is a common problem.

The extraction of depth information can be done in many different ways. There are approaches such as [1] where correlation methods are applied to extract dense depth maps of the hand, but the lack of texture usually makes the use of correlation matching difficult. Sometimes, special hardware is used in order to extract a dense depth map [8], [9]. As we said before, in [3] a different approach is applied since only the hand contour is reconstructed. This makes the matching process easier and the reconstruction faster. In the original system this paper is based on, the ICP algorithm was used to match the points along the contours, and then the 3D positions were reconstructed by classical triangulation. However, this approach has an important drawback: It assumes that the contour points lie on a single plane. This can be avoided using a more complex transformation in the ICP process (affine transformations were used in the original system), or using another matching process. This paper proposes an improved method for matching the contour points based on the dynamic time warping algorithm [15]. The main advantage of the proposed approach is that it does not make any assumptions about the the hand contour geometry as only the epipolar constraint is used.

Fig. 1. Stereo Head used in our system

III. Contour Matching

In this section, we briefly introduce the algorithms used. First, an introduction to ICP and its application to hand contour matching is given. This algorithm was used in our previous work presented in and more details can be found in [3]. After that, we introduce the use of dynamic time warping for hand contour matching.

A. ICP

ICP algorithm computes a motion which transforms one set of points into another one according to a model, for example, an affine transformation is assumed in our original work. The algorithm is iterative: At each iteration, it first computes a motion which minimizes the current matching error and then applies the estimated motion to update the point correspondences. Two strong assumptions are made:

- An initial approximation for the point correspondences is available in order to compute the initial motion.

- A suitable motion model is available to perform the contour alignment.

The procedure followed in [3] is shortly outlined below. The initial approximation performed in order to compute the initial error measure is based on the first and second moments of the contours. This means that the hand contours are approximated by ellipses and their centroids and principal axes are matched. This approximation is fast, but it has some problems: the more circular this ellipse approximation of the contour is, the more uncertain is the alignment of the axes.

The motion model chosen in [3] is an affine transformation. This means that in the alignment process, six parameters have to be estimated. In terms of stereo matching, this works fine as long as the contour points lie in a plane which also restricts the algorithm to work only for some hand poses. The problem can be solved by choosing a more general transformation. However, more general transformations require more parameters to be optimized. For this reason, the number of iterations until the convergence is achieved may increase drastically. Finally, ICP needs a measure of the error during the matching process. In the original work, the 2D squared distance between a point and the transformed point was used to measure the similarity between points. This algorithm is relatively fast, since it computes a motion that is applied to all the points at the same time. On the other hand, it is iterative and sometimes the number of iterations required to reach the convergence may be high.

B. DTW

Another approach adopted here, dynamic time warping, is conceptually quite different. It is not iterative, and it computes the point correspondences without any assumption of the underlying motion model. The algorithm consists of four steps:

1) Compute the pairwise distances from each point in one set to all the points in the other set.
2) Select a pair of points that are supposed to be a good match, in order to initiate the matching process.
3) For each possible pair of points, compute the accumulated cost of reaching this pair, based on the accumulated cost of previous points and the cost of the jump from the previous pair (the pair with minimum accumulated cost previously computed).
4) The optimal path corresponding to the minimum total cost of matches can be extracted by tracing back from the end point.

The algorithm has time complexity $O(n^2)$ where n is the number of points to be matched. In the remaining part of the section, we present the principal details of the proposed approach: the building of the distance matrix and the choice of the starting point. More information about DTW can be found in [15].

1) Distance Matrix: DTW algorithm finds the set of correspondences with the least total cost (or distance) between the matched points. For this reason, it is very important to choose a distance measure that in a good way represents the similarity of two hand contours in a stereo pair of images.

From a geometric point of view, the relation between points in a calibrated stereo pair is given by the essential matrix. For a given point, this matrix provides the epipolar line on where the corresponding point must lie in the other image. This matrix depends on the extrinsic calibration of the cameras: the translation vector and the rotation matrix between them:

$$P_R^{\mathrm{T}} R [t]_x P_L = 0 \;\Leftrightarrow\; P_R^{\mathrm{T}} E P_L = 0 \tag{1}$$

$$[t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2}$$

The relations are applied to points P_R and P_L in the normalized camera coordinate systems, and are also valid for their normalized image plane projections p_R and p_L. We are interested in this relation expressed in the image coordinate system in order to determine the correspondence between the left and right camera images. Considering the camera intrinsic parameters K^{-1}, we obtain the well known fundamental geometry relationship:

$$p_L = K_L^{-1}\overline{p}_L \qquad p_R = K_R^{-1}\overline{p}_R \tag{3}$$

$$p_R^{\mathrm{T}} E p_L = 0 \tag{4}$$

$$\overline{p}_R^{\mathrm{T}} K_R^{-\mathrm{T}} E K_L^{-1} \overline{p}_L = 0 \tag{5}$$

$$\overline{p}_R^{\mathrm{T}} F \overline{p}_L = 0 \tag{6}$$

$$\overline{u}_R = F\overline{p}_L \tag{7}$$

The points $\overline{p}_R$ and $\overline{p}_L$ represent pixel coordinates and $\overline{u}_R$ represents the epipolar line where $\overline{p}_R$ must lie. The fundamental matrix F can be computed from the intrinsic and extrinsic parameters of the camera or estimated directly from a set of calibration images. The calibration of the stereo rig was performed with the tool available from [19].

The distance from a point to the epipolar line generated by another point can be used in order to build the distance matrix. The main advantage of this measure compared to other measures such as the coordinates of the points or the coordinates with the origin in the hand centroid is that the measure is not based on any assumption except the epipolar geometry: if the camera calibration is correct (and there are no occlusions), the corresponding point lies on the epipolar line. The main disadvantage is that this measure might be ambiguous: in the case of point denoted 0 in Figure 2, it will perfectly match both points denoted 0 and 8 in another image.

However, this problem is solved by the DTW algorithm. Although two points would have similar distances to each other, the subsequent ones will probably have very different distances (as points 1 and 9 in Figure 2). This means that despite the distance in the first pair will be low, the following matches will increase a lot the total cost of this matching, rejecting finally this set of correspondences. When there are points with similar distances close to each other, the system has errors. This can be seen in Figure 13, where the wrist segment of the contour has a lot of noise since this segment is almost parallel to epipolar lines.

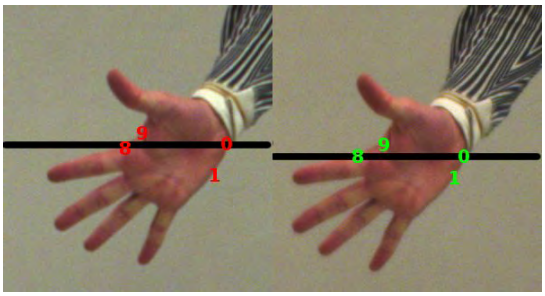

Fig. 2. Point 8 has distance 0 to point 0 since it is based on the distance between the epipolar lines the points lie on.

Another problem solved by the DTW algorithm are the occlusions. If some points are occluded just in one of the images from the stereo pair, the number of contour points will be different in each image. When the algorithm reaches the point when the partial occlusion begins, it detects that the distance between the next pairs increases until the occlusion finishes. If a suitable distance measure is chosen, DTW will drop those points, that is, it will leave them without any correspondence in the other image. The matching will continue normally when contour points are visible in both images, see Figure 3.

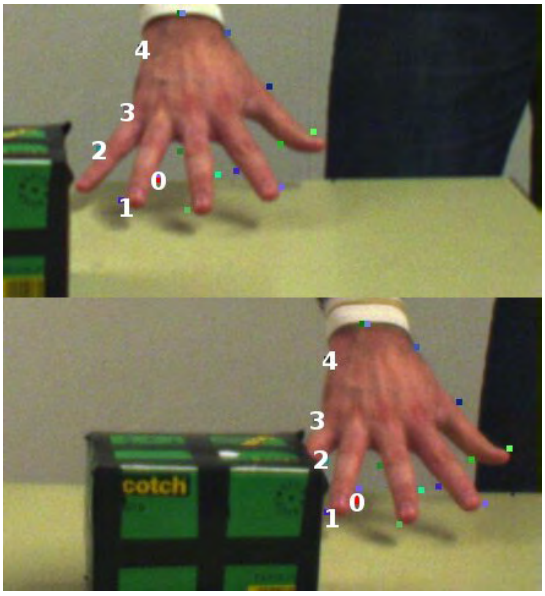

Fig. 3. Only point 2 is wrongly matched due to the occlusion.

2) Starting Point Selection: Matching cyclic sets of points with DTW has one prerequisite: the beginning and end matching pair have to be chosen and it should be the same point pair. Although DTW can align sequences with disaligned starting points, the points that are dropped while aligning the sets are lost. In Figure 4 we show an example of that: the real starting point pair should be the pair $(5, 0)$ instead of $(0, 0)$. As the consequence, only the central point pairs of the contours, from $(5, 0)$ to $(14, 9)$, are well matched. There are solutions to this problem ([5]), but they usually require at least two times the computations required by the original DTW algorithm. For this reason, having a good starting point pair selection is important.

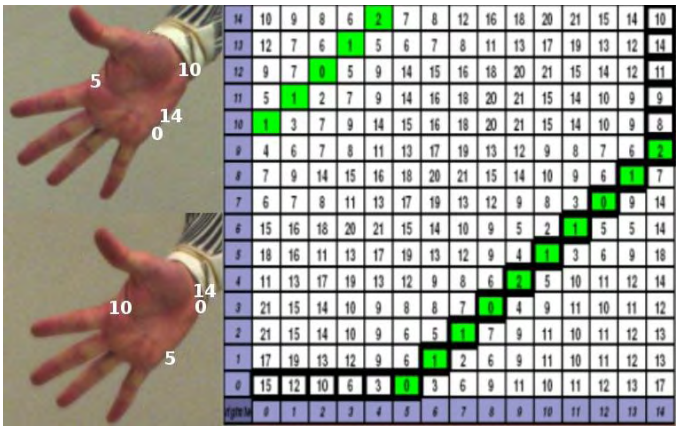

Fig. 4. Green cells corresponds to the ideal correspondences, bordered cells are the real correspondences

It seems reasonable to use the distance matrix built with distances to epipolar lines in order to use the point pairs with lowest distance as starting points. Unfortunately, as said, this measure is ambiguous, so we have many pairs with very low distances which do not represent good matches. One can think about applying another measures such as using the coordinates centered in the middle of the hand to fix this ambiguity. However, there is another problem: since the distance to an epipolar line is a continuous real value, a threshold should be applied to consider a point "close" or "far" to the epipolar line and it may be difficult to find such a threshold.

For this reason a different approach was taken. If we assume that the hand pose does not change too much between the consecutive frames, a good match from the current frame after applying the DTW algorithm, which avoids the ambiguity of the distance measure, can be used as a starting point. In the first frame, the starting point is selected based on the distance matrix. Although the matching can be wrong in some points, there is a region between the alignment portion in the beginning and the end, see Figure 4, where the points are well matched. With this procedure, the selection of starting point is fast and accurate and the problems may occur only in the first frame.

C. Accumulated distance computation

Once the starting point pair has been selected, we can begin the computation of the accumulated cost until each possible point pair in the matrix has been visited. The allowed transitions and their related costs J have to be defined. There are different possibilities related to the allowed transitions in a DTW system. It is interesting to discuss what each of the possible transitions means, for example:

- From pair (m, n) to pair $(m + 1, n)$: we advance one point in the first contour but stay in the original point on the second contour. We denote this an "alignment" jump.

- From pair (m, n) to pair $(m + 1, n + 1)$: we advance one point on both contours. We denote this a "matching" jump.
- From pair (m, n) to pair $(m + 2, n)$: we advance two points in the first contour but stay in the original point on the second contour.
- From pair (m, n) to pair $(m + 2, n + 2)$: we advance two points on both contours.

In our system, we only allow the transitions from (m, n) to $(m, n + 1), (m + 1, n)$ (alignment) and $(m + 1, n + 1)$ (matching). We have tested the performance of the system by allowing longer transitions and we concluded that the improvement was not significant.

In the proposed approach, the cost associated to a transition serves as a multiplier of the distance associated with a point pair. It can be used to favor shorter transitions, or to favor "matching" transitions, for example. For example, if there are no occlusions, we would want to favor matching transitions where we advance one point in both contours, and not transitions for alignment, where only one contour advances for one point. However, we experienced that the behavior with this approach is worse in cases of partial occlusions, where a lot of alignment transitions are required to match the contours properly. A good balance in our system was a factor of 1.5 for alignment, and a factor of 1 for matching, meaning that alignment has an additional cost. This improved considerably the matching process in the fingertips, where there are points with similar distances very close.

The accumulated cost C for a pair (i, j) if the last matched point was (m, n) can then be expressed as:

$$C(i, j) = C(m, n) + \tag{8}$$

$$+min_{allowed\ jumps}(J(m - i, n - j) \times c(i, j)) \tag{9}$$

$$\tag{10}$$

An example with constant cost for the different transitions J is shown in Figure 5. In order to compute the accumulated cost in the white bordered cell $(4, 4)$, we add the intrinsic cost of the pair $c(4, 4) = 1$ and the minimum accumulated cost for the possible predecessors, that in this case is an alignment transition from $(4, 3)$. J is set to one in this case for simplicity. This accumulated cost is calculated for all the point pairs until the end point pair is reached.

1	5 | 26	7 | 22	12	1
4	13 | 21	9 | 15	1 | (1+3)	17
3	8 | 8	5 | 6	2 | 3	15 | 18
2	0 | 0	1 | 1	7 | 8	11|19
right/left	2	3	4	1

Fig. 5. Distance matrix with accumulated costs.

D. Backtracking

Once all the accumulated distances have been calculated, the backtracking process begins. The set of best correspondences is extracted by backtracking the "best predecessor" from the end point pair and repeating the process until the starting point pair is reached.

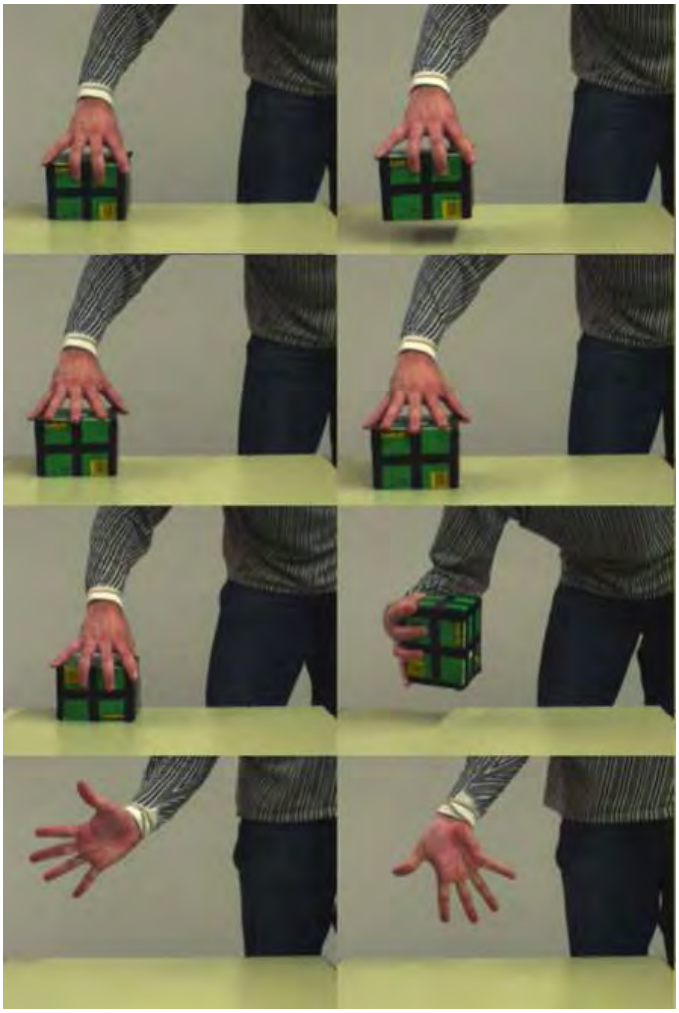

Fig. 6. Two frames of each sequence used in evaluation: moving, pushing and rotating and object and simple hand waving.

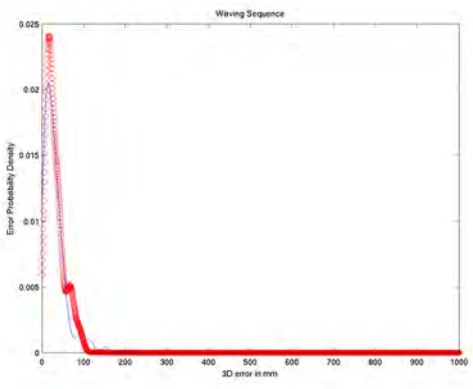

Fig. 7. Results for the waving sequence: ICP(thick/circle line) and DTW (thin line).

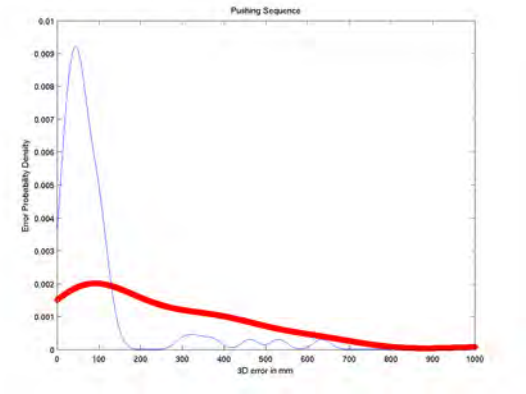

Fig. 8. Results for the pushing sequence: ICP(thick/circles line) and DTW (thin line).

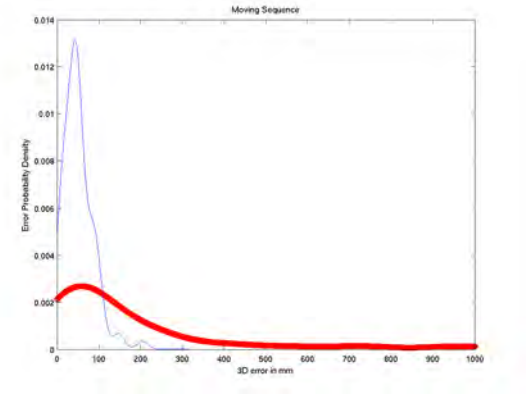

Fig. 9. Results for the moving sequence: ICP(thick/circles line) and DTW (thin line).

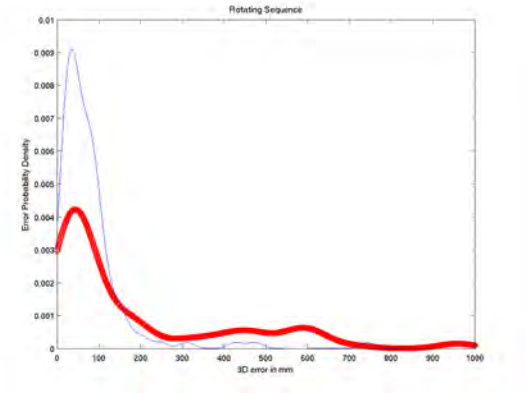

Fig. 10. Results for the rotating sequence: ICP(thick/circles line) and DTW (thin line); Rotating sequence

IV. Experimental evaluation

We compared the performance of the ICP algorithm developed in [3] and the DTW algorithm developed proposed here in a number of hand gesture and object manipulation sequences. For the ICP algorithm, we allow the maximum number of iterations to be 50. The sequences on which the performance of the algorithms was evaluated included moving, pushing and rotating an object, and simple hand waving, see Figure 6. Ground truth was provided by manually marking the matched pairs in all frames. The results are represented by plotting the error probability density function.

The waving sequence is the simplest one. The hand contour points lie in the same plane, facing the cameras and without any object involved. These are the ideal conditions of the ICP algorithm, and the performance of this algorithm in this sequence is very good. The performance of DTW in this sequence is slightly worse, see Figure 7.

The rest of the sequences consist on the manipulation of an object, so there are occlusions. ICP is less robust to occlusions than DTW, since usually the hand contour shape is considerably different in stereo images when there are occlusions present, see Figure 3. But the main advantage of DTW is that it shows a good performance for the case when the hand contour points do not lie on a plane, see Figures 8, 9 and 10. Those figures represent the probability density of the distance between the extracted fingertips and the ground truth. As we can see, the distance error in DTW is concentrated in lower values than in ICP.

We have also performed a qualitative evaluation of the two methods. Some of the sample results are presented in Figures 11-13. Here, the reconstruction of hand contours performed with DTW and ICP is compared to the ground truth which is represented as a line skeleton from the wrist to the fingertips.

In Figure 11, it is visible that the performance of DTW is better than ICP when the planar assumption is not satisfied anymore. Even if the extracted contours are almost the same in the image, the reconstructed 3D contour is clearly much better in the case of DTW approach. Figures 13 and 12 also show that the DTW approach not only outperforms ICP but also performs well in cases of occlusions. While the thumb is occluded by the rest of the fingers, it is partially visible in the 3D reconstruction since the right camera image (not present in the figure) had a better angle to visualize the thumb.

V. Conclusions

Extraction and tracking of human hands is an important part of various interaction and instruction systems. Many systems have been proposed in literature, based both on single and multiple cameras. However, most of them are designed for a specific purpose such as extraction of the hand contour without the full reconstruction of the hand's pose. The systems that can extract full pose of the hand are mostly run off-line or require parallel processing on several machines to achieve real-time performance.

Our current work aims at developing a full hand pose tracking system that performs in real-time without any special hardware. For this purpose, we use a stereo setup and built upon our previous work on hand contour tracking that assumed only cases where the hand was kept planar. To allow for more complex cases of object manipulation, we propose to replace the original ICP algorithm with a DTW approach that clearly shows a better performance in the considered sequences. Based on the contour extraction and

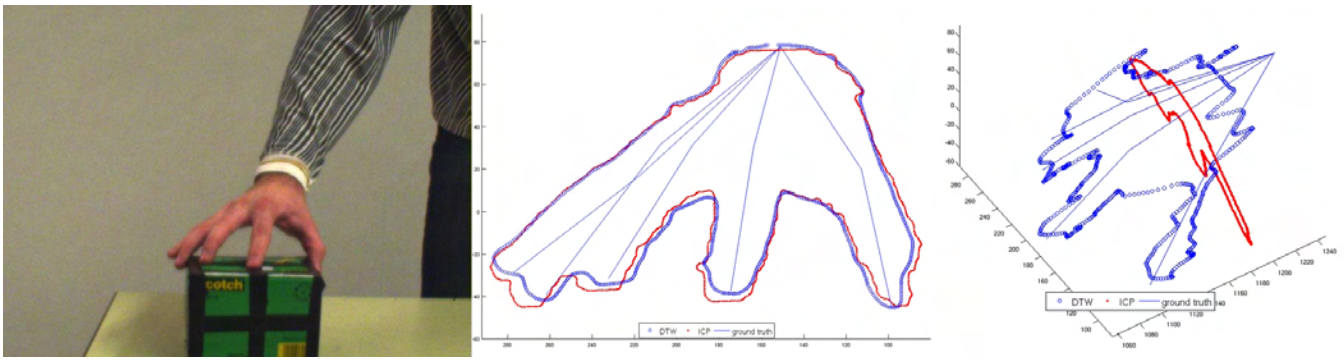

Fig. 11. ICP, DTW and ground truth reconstruction for the pushing sequence.

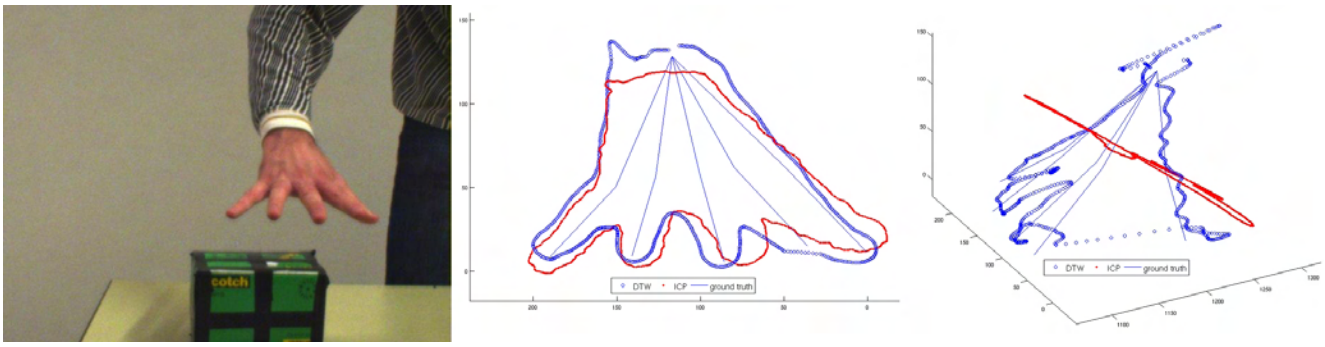

Fig. 12. ICP, DTW and ground truth reconstruction for the moving sequence.

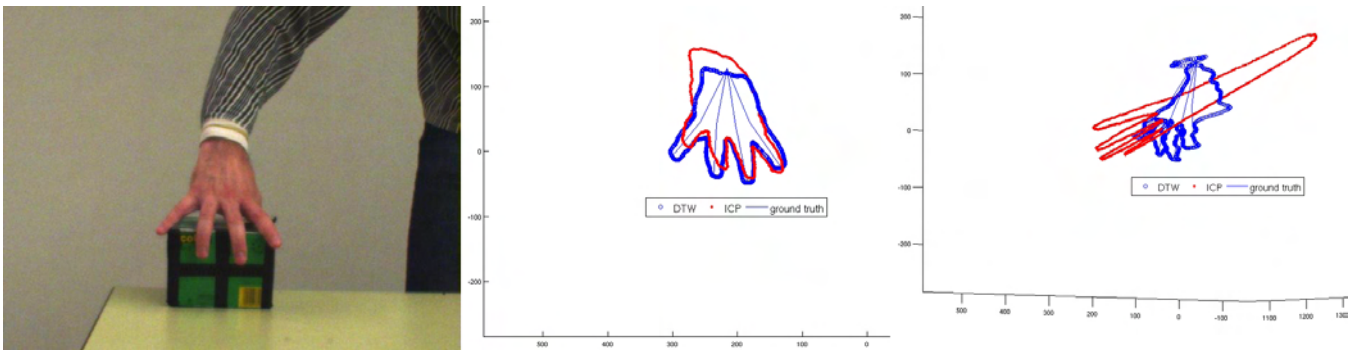

Fig. 13. ICP, DTW and ground truth reconstruction for the rotating sequence.

fingertip detection in 2D, followed by the 3D matching and reconstruction step, the system will be used together with an articulated model of the hand to estimate the state of all the joints of the hand with the use of inverse kinematics.

VI. ACKNOWLEDGMENTS

This research has been supported by EU through the project PACO-PLUS, FP6-2004-IST-4-27657 and the Swedish Research Council.

REFERENCES

[1] Q. Delamarre and O. Faugeras, Finding Pose of Hand in Video Images: A Stereo-Based Approach, *FGR*, 1998, pp 585-590.

[2] T. Kanade and A. Yoshida and K. Oda and H. Kano and M. Tanaka, A Stereo Machine for Video-Rate Dense Depth Mapping and Its New Applications, *Proc. of the 15th Computer Vision and Pattern Recognition Conference, CVPR*, 1996, pp.196-202.

[3] A. A. Argyros and M. I. A. Lourakis, Binocular Hand Tracking and Reconstruction Based on 2D Shape Matching, *Proc. of the IEEE International Conference on Pattern Recognition*, 2006, pp.207-210.

[4] B. Stenger. Model-based hand tracking using a hierarchical Bayesian filter, PhD thesis, department of Engineering, University of Cambridge, March 2004.

[5] N. Arica, Dynamic time warping for cyclic sequence comparison, *Proceedings of the IEEE 13th Signal Processing and Communications Applications Conference*, pp.127- 130.

[6] V. Pavlovic and R. Sharma and T. S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review", *IEEE Trans. Pattern Analysis Machine Intelligence*, 1997, pp 677-695

[7] S. Ahmad. A Usable Real-Time 3D Hand Tracker, *Asilomar Conference on Signals, Systems and Computers*, 1995, pp 1257-1261

[8] R. Feris and R. Raskar and L. Chen and K. Tan and M. Turk, Discontinuity Preserving Stereo with Small Baseline Multi-Flash Illumination, *ICCV*, 2005, pp 412-419

[9] M. Bray and E. Koller and L. Van Gool, Smart Particle Filtering for 3D Hand Tracking, *Face and Gesture Recognition*, 2004, pp 675-680

[10] J. M. Rehg and T. Kanade, DigitEyes: Vision-Based Hand Tracking for Human-Computer Interaction, *CMU-CS*, 1993

[11] Y. Wu and T. Huang, Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach, *ICCV*, 1999, pp.606-611.
[12] V. Athitsos and S. Sclaroff, Estimating 3D Hand Pose from a Cluttered Image, *CVPR*, 2003, pp 432-442
[13] P. R. S. Mendonça and R. Cipolla, Model-Based 3D Tracking of an Articulated Hand, B. Stenger and *CVPR*, 2001, pp 310-315
[14] C. Tomasi and S. Petrov and A. Sastry, 3D Tracking = Classification + Interpolation, *ICCV*, 2003, pp 1441-1448
[15] C. S. Myers, A comparative study of several dynamic time warping algorithms for speech recognition, *M.S. at MIT*, 1980
[16] T. Campos, Art tracker demos. Website. ”http://www.robots.ox.ac.uk/ teo/art/”.
[17] A. Erol, G. Bebis, M. Nicolescu, R. Boyle, X. Twombly, Vision-Based Hand Pose Estimation: A Review, submitted to Elsevier Science, 2006.
[18] C. Nolker, H. Ritter, Grefit, Visual recognition of hand postures, in: A. Braffort, R. Gherbi, S. Gibet, J. Richardson, D. Teil (Eds.), Gesture-Based Communication in Human-Computer Interaction, Proc. International Gesture Workshop, France, Springer Verlag, LNAI 1739, 1999, pp. 61-72.
[19] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. Camera calibration toolbox for matlab. “http://www.vision.caltech.edu/bouguetj/calib_doc/index.html”.
[20] R. Rosales, V. Athitsos, L. Sigal and S. Sclaroff. 3D hand pose reconstruction using specialized mappings, *In. Proc. ICCV*, vol.1, pp.378-385, 2001.

Modeling and Recognition of Actions through Motor Primitives

David Martínez and Danica Kragic
Computational Vision and Active Perception Lab
Centre for Autonomous Systems, KTH, Stockholm, Sweden

***Abstract*— We investigate modeling and recognition of object manipulation actions for the purpose of imitation based learning in robotics. To model the process, we are using a combination of discriminative (support vector machines, conditional random fields) and generative approaches (hidden Markov models). We examine the hypothesis that complex actions can be represented as a sequence of motion or action primitives. The experimental evaluation performed with five object manipulation actions and 10 people, investigates the modeling approach of the primitive action structure and compares the performance of the considered generative and discriminative models.**

I. Introduction

Imitation learning finds its applications in natural systems regarding development of language and communication, learning and relearning skills for children, athletes, and those recovering from injuries. In artificial systems, imitation has been frequently used for automated programming and control of robots, for finding more natural ways for interacting with robots and task learning in general, [1]–[9].

Imitation has also been viewed as the capability to acquire new skills by observation based on some existing behavioral repertoire, [10]. In [11], it has been shown that an action perceived by a human can be represented as a sequence of clearly segmented action units. These action units can then serve as the basis for building up the behavioral repertoire. Thus, the action recognition process may be considered as an interpretation of the continuous human behaviors which, in its turn, consists of a sequence of action primitives [7] such as *reaching*, *picking up*, *putting down*. The notion of actions and action primitives is thus of significant importance for building structures that directly link sensory and motor systems of artificial systems since they define the necessary mapping for the implementation of the perception-action mechanism.

In this work, we study the problem of modeling and recognition of actions and action primitives using Support Vector Machines (SVM), [12], Hidden Markov Models (HMM), [13] and Conditional Random Fields (CRF), [14]. To start with, SVM is used to model and recognize individual action primitives. Actions, build from a set of action primitives, are then modeled using HMMs and CRFs and their performance is evaluated and compared. We also evaluate the plausibility of using CRFs for recognition of composite actions. The measurements are based on four magnetic sensors where each of the sensors provide a complete pose estimate.

The contributions of this work are as follows:

- We investigate modeling strategies for object manipulation actions that are very similar to each other (*grasping*, *pushing*, *moving*). Most of the current work oh arm/hand action recognition concentrates on actions that are easy to discriminate (*waving*, *pointing*).
- We implement, evaluate and compare both generative and discriminative approaches while most of the reported work concentrates on one of the approaches.
- We consider the problem of recognition both on the primitive and on the composite action level.
- Our measurements are based only on a four magnetic sensors while most other systems use complex motion capture systems providing full body joint data.

The paper is organized as follows. First, we review related work in Section II. The theoretical basis for the work and different approaches to activity modeling are presented in Section III. Section IV describes data collection and the implementation details of the system and Section V presents experimental evaluation. The paper is concluded in Section VI.

II. Related work

There is a large body of work on the problem of human action recognition from images or from 3D positions on the human body; for a recent survey, see [15].

Examples from computer vision community include that of Sullivan and Carlsson [16] where actions are represented as a sequence of key postures. Segmentation is then performed implicitly by searching the observation sequence for key postures that then are used for recognition. The key postures are represented as topological edge maps extracted from video frames. With our simpler magnetic sensor observation space, it would be possible to learn key postures directly from the SVM classes described in the next section.

An alternative approach is to avoid the segmentation problem altogether by employing a discriminative action recognition approach. For example, Sminchisescu et al. [17] use conditional random fields (CRF) for recognition of full human body actions. This method for modeling sequential data is similar to HMM but has the advantage that no explicit model of the sequence of observations has to be learned, thereby rendering explicit data segmentation unnecessary as well. The downside of CRF, as with any discriminative approach, is however the inability to generalize to previously unseen action examples when the detailed imitation of the pose is needed.

In the robotics area, the work presented in [4], [5], proposes a framework for acquiring hand-action models by integrating multiple observations based on gesture spotting. The work of [6], presents a gesture imitation system where the focus is put on the coordinate system transformation so that the gesture

induced by the teacher is transformed into the robot's ego-centric system. A system for deriving behavior vocabularies or simple action models that can be used for more complex task extraction and learning was presented in [7], while [9] presents a learning system for one and two-hand motions where the robot's body constraints are considered as a part of the optimal trajectory generation process. A common trend in the above systems is that the studies are based on a single user generating the motion. A natural question to pose here is how the underlying modeling methods scale and apply for cases when the robot is supposed to learn from multiple teachers that have not been specifically trained prior to the training process. The experimental evaluation conducted in our work is based on 10 people.

In terms of the theoretical framework, support vector machine (SVM) has been applied to several different application areas such as visual and speech data modeling and recognition, [18]–[20]. One of the early works on SVMs, [18] presented a drawback of the method when working with sequential data, namely, that SVM lacks a way of handling the time dependencies in the data. In order to use time dependent sequences as SVM input, variable length time sequences can be either normalized to same length before applying the SVM. Another approach is to embed dynamic time warping (DTW) directly into the SVM kernel function [21]. Third, probably most common way to handle the "time problem" is to combine a SVM with Hidden Markov Models (HMM) [18], [20], [22]. SVM is still used to classify single points or brief time windows, but the output of the SVM is then used an input to a HMM which then finds the most probable path or sequence in consideration of time. It is also well known that the choice of the SVM kernel function has a significant effect on the results but unfortunately the best choice is application dependent.

From the point of view of imitation learning or "learning by showing", the primitives are an attractive option since they can alleviate mapping motion from humans to robots which differ in their embodiment. In addition, having a common vocabulary of primitives can aid in task understanding and planning as the task can be then described as a sequence of events. For this reason, we now concentrate on this body of work. Vecchio et al. [23] model two-dimensional drawing actions as dynamical systems and classify and segment motions according to a priori known motion classes. Representation and segmentation of repetitive movements has been studied by Lu et al. [24] using an auto-regressive model and detecting changes in the model parameters. Finally, stochastic parsing has been proposed for primitive-based action recognition and understanding, [25], [26].

III. Modeling Actions

In the work presented here, we consider five different object manipulation actions: a) pick up an object from a table, b) rotate an object on a table, c) push an object forward, d) push an object to the side, and e) move an object to the side by picking it up. To include variation in the actions, each action is performed in 12 different conditions, namely on two different heights, two different locations on the table, and having the demonstrator stand in three different locations (0, 30, 60 degrees), see Fig. 1. All actions are demonstrated by 10 people.

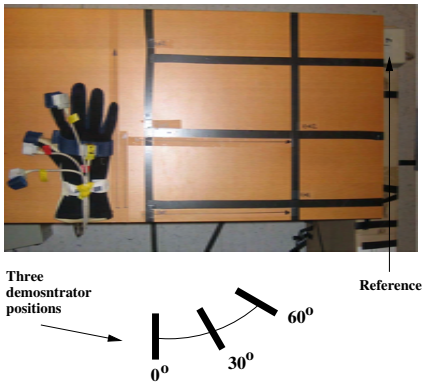

Fig. 1. Glove with the sensors, the table and the three demonstrator locations.

The movement is measured using a Nest-of-Birds magnetic tracker. The test subject is endowed with four sensors each registering their full 3-dimensional pose with respect to a reference, see Fig. 1. The sensors are located on: a) chest, b) back of hand, c) thumb, and d) index finger. The chest sensor is used to provide a reference to the demonstrator position while the back of the hand can be used as a reference for the thumb and index finger. The measured sequences have been annotated by hand such that the current action primitive is known for training.

Below, we present the theoretical basis on recognizing individual primitives using SVMs and the time sequence modeling using HMMs and CRFs. Different approaches of primitive based modeling of actions are also described.

A. Support vector machines

Support vector classification aims at separating data classes, mapped into a high dimensional feature space, by hyperplanes with a maximal margin to the classes. A hyperplane represents the decision boundary of the classifier with feature vectors on one side belonging to one class and vectors on the other side to another one. To represent complex decision boundaries, the mapping (kernel) from the original feature space to the high dimensional space is nonlinear. In this work, a standard SVM with Gaussian kernel is used.

In our work, we apply SVM classification to multiple classes each representing an action primitive class. For this purpose, we adopt one-against-one approach. In other words, by denoting the number of classes by k, $k(k-1)/2$ classifiers are trained using all pairs of classes. To classify a sample from an unknown class, it is classified by all classifiers, and each result is a vote for the class. Majority voting is then used to decide the class of the sample. The one-against-one approach has been found very successful with SVMs but it suffers from increased number of individual classifiers when the number of classes is very high.

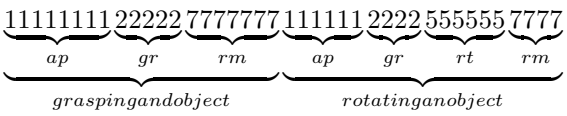

Fig. 2. Example output of the SVM classifier.

The output of the SVM is then a sequence of classified primitive actions at each time instant. In Fig. 2 we show an example of two concatenated actions: grasping and rotating an object. A grasping action is composed of three primitives (approach, grasp and remove), while the rotation is composed of four primitives (approach, grasp, rotate and remove hand). In terms of lengths, a real grasp ,am be composed of 30 approach, 20 grasp and 50 "remove hand with object" primitives.

Our SVM implementation uses 7 different classes, which correspond to the 7 primitive actions: approach, grasp, rotate, push forward, push side, remove and remove with object. It is worth mentioning the difference between a primitive and an action: a primitive is our basic unit, like a letter in a word, while an action is a composite of primitives, like a word.

B. Markov chain and hidden Markov models

To model the temporal dependencies of actions, the first approach we adopt are time-homogeneous Markov chain models where the state transition probabilities are invariant over time. As a short reminder, denoting the state i by ω_i, the time evolution of states can then be described using the state transition probabilities $P(\omega_j(t+1)|\omega_i(t)) = a_{ij}$. The states themselves are hidden, not directly observable. Instead, in each state, an observation $\mathbf{x}(t)$ is made. The observation depends only on the current state according to a selected probabilistic model, that is, $P(\mathbf{x}(t)|\omega_i(t)) = P(\mathbf{x}|\omega_i)$. If the set of observations X is discrete and finite, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the observation probabilities can be written more shortly as $P(\mathbf{x}_j|\omega_i) = b_{ij}$. Finally, the probability of starting in state ω_i can be defined as $\pi_i = P(\omega_i(1))$. Thus, the parameters can be collected to matrices $\mathbf{A}$ and $\mathbf{B}$ and a vector $\boldsymbol{\pi}$.

The objective of the work presented here is to model actions based on motor primitives. Motor primitives correspond to individual states of the HMM. A typical approach for using HMMs in recognition is to build a single HMM for each class to be recognized and then determine the class of an unknown sample by using the maximum likelihood method. In our work, we take another approach and represent the whole set of actions with a single HMM, such that different paths through the HMM correspond to different actions. This is because many actions contain similar parts and since the HMMs are not considering dependencies between the observations as such, we believe that this model may account for some of the problems caused by this. A simple example is shown in Fig. 3. Here, both rotating and pushing an object both require first the hand to approach the object.

As mentioned earlier, a hypothesis followed in this work is also that more complex actions can be modeled using a set of motor primitives. Thus, instead of making a choice between several HMMs, the most probable path through the HMM is sought. The path is found by the Viterbi algorithm [13], a dynamic programming based algorithm for determining the maximum likelihood path through a HMM given a sequence of observations $(\mathbf{x}(1), \mathbf{x}(2), \ldots)$. It finds the state sequence $(\omega(1), \ldots)$ for which

$$P\left(\mathbf{x}(1), \ldots, \mathbf{x}(T)|\omega(1), \ldots, \omega(T)\right) \tag{1}$$

is maximal. We employ the computationally tractable solution based on defining (1) recursively.

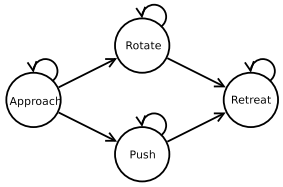

Fig. 3. Modeling two actions (rotate, push) using primitives.

For initial learning of the HMM parameters, an alternative approach to the traditional Baum-Welch learning is adopted. We assume that the training data is labeled, that is, for each time step, the current motor primitive is known. Then, the transition probability matrix $\mathbf{A}$ can be directly estimated from the training data, as if in the case of a Markov chain model instead of a HMM. We use the maximum likelihood estimate, in other words, the transition probabilities are calculated directly from the training data. The output of the SVM is used as the observations of the HMM. The observation probabilities need also be estimated as it is not expected that the classifier will be able to classify all samples correctly. Maximum likelihood estimation using the known correct classes is also used to estimate the observation probabilities. Therefore, the observation matrix $\mathbf{B}$ corresponds to the confusion matrix of the classifier.

1) Action modeling using HMMs: The hypothesis in the modeling is that each of the manipulation primitives is generic and that their number is limited. However, the best applicable set of primitives is not known and one of the goals of our previous work was to study how the manipulation actions can be considered in terms of primitives. In [27], we have investigated two different models of action representation, see Fig. 4. Approach 1 considered each of the manipulation actions as a primitive. In addition to the manipulation actions, two assisting actions, *approach* and *remove* are inherent in all action sequences (see Fig. 4). The assisting actions alleviate the segmentation of the manipulation part of the action. Approach 2 considered therefore that the manipulation part of the action can be composed of multiple primitives. The model on the right in Fig. 4 can be chosen based on the knowledge that the rotation and moving the object require grasping. Our working hypothesis in [27] was that Approach 2 would be more effective in recognizing actions compared to the first approach. In addition it would allow learning of new actions based on the known primitives. Considering both approaches,

an action was represented by a separate path through the left-to-right Markov model. We have shown how the process of embedding new actions given primitives can be formalized.

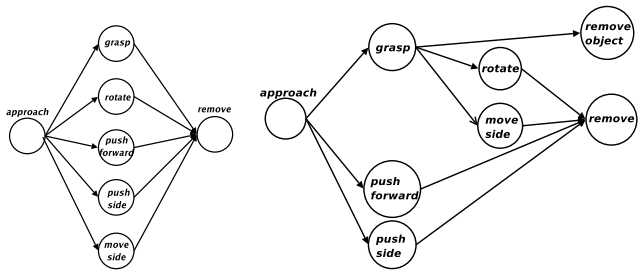

Fig. 4. (left) Approach 1: Actions as primitives; (right) Approach 2: Composite actions.

C. Conditional Random Fields

A conditional random field, [14] is a stochastic process for segmenting and labeling data; it is a discriminative framework that describes the probability of having a set of labels S given the set of observations O, $\mathrm{P}(S|O)$. It is modeled as an undirected graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ where each vertex is associated with a label S_i. Only vertices connected by an edge are conditionally dependent. Although the graph can be as complex as desired, in this work we focus on linear CRFs, where any vertex v_i is connected to the previous and the following ones (v_{i-1} and v_{i+1}). Linear CRFs are adequate when data is a sequence $O = o_1, \ldots, o_n$, and then the resulting set of labels will be also a sequence $S = s_1, \ldots, s_n$. Each s_i is an element of a finite alphabet γ.

At the use stage, we need to obtain the labeled sequence with the highest probability given the set of observations O. That probability is defined as

$$\mathrm{P}_\lambda(\mathrm{S}|\mathrm{O}) = \frac{\exp\left(\sum_{i\in 1..N} \lambda_i \cdot \mathrm{f}(s, O, i)\right)}{\mathrm{Z}_\lambda(O)} \tag{2}$$

where $\mathrm{Z}_\lambda(O)$ is a normalization factor of the form

$$\mathrm{Z}_\lambda(O) = \sum_{s\in S} \exp\left(\sum_{i\in 1..N} \lambda_i \cdot \mathrm{f}(y, O, i)\right) \tag{3}$$

Here, $\Lambda = \{\lambda_i, i \in 1..N\}$ are the parameters that define the CRF model (those that will we estimated at training), while $\mathrm{f}(s, O, i)$ are feature functions. Those features are used to describe the attributes of the data. The sequence of labels with the highest probability is then found by

$$\arg\max_s \mathrm{P}_\lambda(\mathrm{S}|\mathrm{O}) \tag{4}$$

Regarding training, we have a set $\mathrm{T} = \{(\mathrm{O}^{(i)}, \mathrm{S}^{(i)})\}_{i=1}^{L}$. Each $\mathrm{S}^{(i)}$ is the correct labeling for the set of observations $\mathrm{O}^{(i)}$, and L is the number of training pairs. The goal of the training algorithm is to estimate all the parameters Λ that define the CRF model and there is one parameter per feature. One way of determining these parameters is by maximizing the log-likelihood function

$$\mathrm{L}(\lambda) = \sum_{i=1}^{L} \log \mathrm{P}_\lambda(\mathrm{S}^{(i)}|\mathrm{O}^{(i)}) \tag{5}$$

1) Data representation and action modeling using CRFs: First, the training data has to be labeled for the estimation of the CRF parameters. The easiest way is to have an alphabet of labels $\gamma = \{\mathtt{g\_}, \mathtt{r\_}, \mathtt{pf}, \mathtt{ps}, \mathtt{m\_}\}$; where each member of γ represents one of the 5 actions presented in Section III). All the observations that belong to the same action receive then the same label. Using this data labeling approach, denoted as Format 1, we have compared the performance of the CRF based approach with the HMM approach for action recognition purposes.

In this work, we also perform recognition of continuous actions. In other words, we assume that the observation sequence is built of multiple actions which are not segmented. This corresponds to having multiple words in a sentence but with no process of segmentation of individual words being applied prior to the classification process. The purpose of this evaluation was to investigate if the CRF approach may be used in cases where a robot observes a human in a, for example, *setting-a-dinner-table* scenario and label individual actions during the task execution process.

For this purpose, we use a second format to label the data, denoted Format 2. As in the case of Format 1, we have first compared the performance of the CRF based with the HMM approach for action recognition purposes. Here, the label set is denoted as $\gamma = \{1, 2, 3, 4, 5, 6, 7\}$; there are as many elements in the alphabet as primitives have been considered in this work. The label sequence given to a sequence of observations is that sequence of observations without errors. Table I shows an example of both formats for an input sequence consisting of two actions, a grasping and a rotation.

Once the CRF returns the most probable labeling sequence, we have to post-process it in order to obtain a sequence of actions; with the first format this is trivial, since keeping the label that received the majority of votes performs well in most of the cases. For the second format, the approach is more complex, as we still have a sequence of primitive labels. In this case we could use a finite state machine to get the sequence of actions, but we have taken another approach, the use of regular expressions, as our results can be seen as text strings. We match several regular expressions and make appropriate substitutions; first we make substitutions of the kind $\mathtt{1546} \rightarrow \mathtt{r\_}$ (full isolated actions); second, $\mathtt{154} \rightarrow \mathtt{r\_}$ (actions without retreat); third, actions without approach, and finally, actions without either approach or retreat primitives, like $\mathtt{54} \rightarrow \mathtt{r\_}$. The order is important, because if we first substitute `154` in the sequence `1546136`, at the end we will have an isolated `6`, which will be considered as a mistake while in fact it is not. Table I shows the final solution after this post-processing step.

Input sequence	1 2 5 5 5 7 7 1 1 5 5 7 4 4 6
CRF output, format 1	g_g_g_g_g_g_g_r_r_r_r_r_r_r_r_
CRF output, format 2	1 1 5 5 5 7 7 1 1 5 5 4 4 4 6
Final result	g_ r_

TABLE I

FINAL RESULT FROM AN OUTPUT LABELED SEQUENCE

IV. SYSTEM DESIGN AND IMPLEMENTATION

The goal of an imitation system is to recognize a set of continuous actions performed by a human and in this work we focus on hand movements in particular. We start by pre-processing the input data for noise removal and continue with the SVM classification; the output is a sequence of primitives that we use to feed a HMM and a CRF. Two cases considered in the experimental evaluation are i) comparison of the HMM and a linear CRF for action recognition, and ii) the use of CRF for simultaneous segmentation and classification of continuous action sequences. With the first approach, we also obtain a baseline for evaluation of how good the CRF with continuous sequences may perform, which is our main goal.

A. Pre-processing

The following sensor measurements are used:

- position of the hand relative to the chest: x, y and z
- position of the index relative to the hand: x, y and z
- position of the thumb relative to the hand: x, y and z
- velocity of the hand: v_x, v_y and v_z.

We start by applying the median filter of length 7 twice to the data so to eliminate the noise peaks. After filtering, the hand and finger locations were transformed into the chest reference frame. Next, the position of both the thumb and index was calculated with respect to the back of the hand. A Gaussian filter was then applied for the finger positions to reduce the noise, which was found to be most apparent in the finger position measurements. The velocity was estimated by time differences between two consecutive time instants. It was then filtered by a Gaussian filter to decrease the noise due to the differential nature of the estimation process. Finally, every dimension was linearly scaled to interval $[0, 1]$.

B. Action recognition

The training data collected are representing 'isolated' actions; we did not collect continuous sequences of actions. The process of labeling complex continuous sequences of actions for the purpose of using them as the input for the CRF training would have to be done by hand, which is a time-consuming task. Instead, we generate continuous action sequences by concatenating primitive actions. While an action always begins with an approaching an object, and ends with a retreating of the hand in a movement involving two consecutive actions on the same object, we do not move away the hand after the first action and approach again to the object, as the hand is already in the vicinity of the object. This adds more complexity to the classification system, as now when we mix actions and action primitives in a sequence, some of them are very short (there are continuous grasping actions with 5 observations) while others (mainly rotations) are very long (some of them have more than 180 observations). We also anticipate that for the future we will have to consider a hierarchical approach where both actions and primitive actions are considered.

C. Classification of continuous action sequences

In order to evaluate the performance of the CRF based approach, we compare it with the ground truth data using Needleman-Wunsch algorithm [28] to align the sequences. This algorithm is based on the dynamic programming approach and it is commonly used to find the most probable global alignment between two sequences of data.

We need to measure differences in terms of the number of inserted, deleted and misclassified actions. After running the algorithm, it returns both sequences aligned in the optimal way and a score, which gives us the similarity or dissimilarity between both sequences, depending on a set of predefined parameters. In Figure 5, we show an example of the best alignment of two sample sequences; the three possible types of errors are labeled as D (Deletion), I (Insertion) and M (Misclassification).

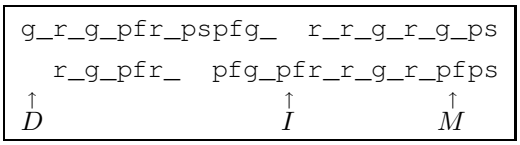

Fig. 5. Best alignment of two sequences.

Two parameters must be set, one called a gap, associated with the weight given to an insertion or a deletion, and S, a similarity matrix. S_{ij} (and S_{ji}) indicates how similar symbols t_i and t_j are. Each t_i belong to a finite alphabet T that contains all the possible symbols that can be inside the sequences to be compared; in our case T = {g_,r_,pf,ps,m_}.

The values for both parameters in this work are $gap = 1$ and

$$S_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

With these parameters, the algorithm will return the sum of insertions, deletions and misclassified actions found after the alignment. After the alignment process has finished, different metrics could be used to obtain a numerical measure of how good the solution is; we have chosen the F1-score, [29]:

$$F1 = \frac{2PR}{P + R}$$

where P denotes precision and R denotes recall.

V. EXPERIMENTAL EVALUATION

We gathered the training data with 10 people, performing all actions in 12 different conditions. For each action, there are thus 120 different samples. Based on these, we generated 240

mixed sequences consisting of actions and primitive actions. Prior to executing the actions, the demonstrators were given only an oral explanation of the task; for this reason, both the inter- and cross-personal variance in the data is high. For SVM learning the training sequences were segmented and labeled manually.

A. HMMs and CRFs for isolated recognition

We start with a comparison of recognition rates for action modeling using HMMs and CRFs. Although we use input sequences with only one action, that action can be either with or without the approach/remove part. In our previous work, [27] we have evaluated different structures of HMMs for action modeling. Based on that, we kept the composite action model that performed best. We note that this model is applicable for recognition of both composite and primitive actions since parsing through the model and state change depends on the probability of observations. We performed leave-one-out cross validation for all ten cases and averaged the results.

Tables II– IV show the results in form of confusion matrices. We added a sixth column, 'unrec', used to count all the actions that could not be classified. For instance, in some cases, the CRF returns a sequence with two different labels having the same probability; in that case, we assume that we cannot disambiguate.

	g_	r_	pf	ps	m_	unrec
g_	63.8	7.5	2.5	1.7	19.1	5.4
r_	0	98.7	0	0	1.3	0
pf	2.1	0	94.6	0.8	0.8	1.7
ps	0	1.7	4.1	56.3	37.5	0.4
m_	0	3.4	0	5.8	90.8	0

TABLE II
HMM RESULTS FOR INDIVIDUAL ACTION RECOGNITION.

	g_	r_	pf	ps	m_	unrec
g_	95.9	0.5	0.5	2.2	0.9	0
r_	0	98.3	0.4	0	1.3	0
pf	1.7	1.7	92.9	3.7	0	0
ps	2.1	0.8	5	59.2	32.9	0
m_	1.7	1.3	0	19.5	77.1	0.4

TABLE III
CRF RESULTS FOR INDIVIDUAL ACTION RECOGNITION USING FORMAT 1.

The strength of the CRFs in being able to take several observations into account, significantly improves the recognition of grasping actions (g_). On the other hand, CRFs experience more difficulty in recognizing move-to-side actions(m_) which is often confused with push-to-side. The explanation is that the move-to-side action is explicitly embedded in the HMM model and requires grasping primitive action to occur before the actual side movement. When parsing through a HMM model, the right route will be taken and the probability of recognition move-to-side rather than push-to-side will be higher. On the other hand, when training the CRF, inter- and cross-personal variations in performing these two actions affect the representation more significantly. We have also performed the evaluation of the CRF based on the format 2 of the data. The results are shown in Table IV. A slightly improved performance compared to format 1 can be seen.

	g_	r_	pf	ps	m_	unrec
g_	92.2	1.8	0.9	3.2	0.9	0.9
r_	0	95	0.4	0	2.5	2.5
pf	2.1	0	91.9	3.4	0.9	1.7
ps	2.5	0	3.3	62.1	31.3	0.8
m_	1.3	0.8	0	17.5	79.6	0.8

TABLE IV
CRF RESULTS FOR INDIVIDUAL ACTION RECOGNITION USING FORMAT 2.

B. CRFs for continuous recognition

The second evaluation consisted in testing CRFs with continuous sequences of actions, once more using the mixed training sequences of composite and primitive action samples. For this purpose, we consider only the format 2 of the data, as explained in Section III-C.1.

CRFs are able to learn a grammar if the training process is modeled suitably and we can benefit from it in our case. For instance, they can learn that having a grasping after a rotation is very common, but that having two consecutive graspings is not probable. Once this system is to be implemented and used on a robot, designing action grammars for most typical tasks will be of utmost importance. In our experiments, we have followed this line of thinking and constructed a few simple grammars or task models. Each task is composed of 10 sentences, and each sentence contains a random amount of actions, always between 3 and 10 where the actions are randomly chosen. For each task, training and testing datasets have been made. We can see each sentence corresponding to a specific tasks such as, for example, serving somebody a coffee or setting up a dinner table. Training data contains 10 samples of each task; for each sample we have only used actions from the same person, as this is what we anticipate will happen in realistic applications. Test data consist of 300 sequences, each one following one of the tasks. For testing, actions inside the task are chosen amongst all the people in the original dataset. In short, we assume training with one person but test with several. The overall averaged results are shown in Table V.

Since in the previous section we have observed that the confusion between move to a side and push side is an important problem, we attempted to analyze the system with and without that specific action. Another feature analyzed is what happens when two consecutive actions are equal; the hypothesis is that if both actions are action primitives, they will be mixed and then, the number of deletions when estimating the final classification results will increase.

Test	With m_	Repeat actions	F1 score
1	No	No	92.6
2	Yes	No	84.8
3	No	Yes	92.8
4	Yes	Yes	85.2

TABLE V

CRF RESULTS FOR CONTINUOUS SEQUENCES

From the results in Table V, it can be seen that the difference between having and omitting move-to-side action is significant. The results also show that when we consider repeated continuous actions results are worse which means that the segmentation and recognition can be performed simultaneously. The fact that they are even a bit better in this case is that the training and test data are different compared to the previous experiment.

Although comparing results for individual and continuous recognition is not completely fair, as data representation differs, we see that continuous recognition is still rather good. Moreover, these results can be considered as a lower bound of the results we would obtain in a real scenario. This is due to the fact that many consecutive continuous actions are mixed, and in the solution they appear as only one action. For instance, a primitive grasping action followed by a primitive rotation action on the same object can be considered as a single composite rotation action. This suggests that, for future work, a hierarchical model may be considered.

VI. CONCLUSION

We have presented a system for modeling and recognition of primitive and composite actions using generative and discriminative machine learning approaches. We have started by using support vector machines for primitive action classification and integrated this with models that can take care of the temporal aspects of actions, namely hidden Markov models and conditional random fields. We have built upon our previous work where only hidden Markov models where considered for isolated action recognition - apart from using the conditional random fields we are also investigating their use in a continuous action recognition scenario. We follow the assumption that we can build a system consisting of different sensory primitives from which a more complex actions can be built. These sensory primitives should be natural and easily used in programming motor primitives for robots.

The experimental evaluation performed with 7 primitive and 5 composite actions is based on the training data obtained with 10 people in 12 different conditions. The recognition rates on isolated actions show similar performance between hidden Markov models and conditional random fields, with latter having the capability of classifying very short activities due to the ability of modeling the dependence between the observations. In the case of the continuous action recognition, conditional random fields show high recognition rates. Currently, we are evaluating the system considering typical imitation scenarios and tasks in a humanoid robot framework.

REFERENCES

[1] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching," in *IEEE Transactions on Robotics and Automation*, vol. 10(6), pp. 799–822, 1994.

[2] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[3] A. Billard, "Imitation: A review," *Handbook of brain thory and neural network, M. Arbib (ed.)*, pp. 566–569, 2002.

[4] K. Ogawara, S. Iba, H. Kimura, and K. Ikeuchi, "Recognition of human task by attention point analysis," in *IEEE Int. Conf. on Intelligent Robot and Systems IROS'00*, pp. 2121–2126, 2000.

[5] K. Ogawara, S. Iba, H. Kimura, and K. Ikeuchi, "Acquiring hand-action models by attention point analysis," in *IEEE Int. Conf. on Robotics and Automation*, pp. 465–470, 2001.

[6] M. C. Lopes and J. santos Victor, "Visual transformations in gesture imitation: What you see is what you do," in *IEEE Int. Conf. on Robotics and Automation, ICRA04*, pp. 2375– 2381, 2003.

[7] O. C. Jenkins and M. J. Mataric, "Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion," *Int. Journal of Humanoid Robotics*, vol. 1, pp. 237–288, Jun 2004.

[8] S. Ekvall and D. Kragic, "Grasp recognition for programming by demonstration tasks," in *IEEE Int. Conf. on Robotics and Automation, ICRA'05*, pp. 748 – 753, 2005.

[9] S. Calinon, A. Billard, and F. Guenter, "Discriminative and adaptative imitation in uni-manual and bi-manual tasks," in *Robotics and Autonomous Systems*, vol. 54, 2005.

[10] M. Mataric, "Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics," 2000.

[11] D. Newton et al, "The objective basis of behavior unit," *Journal of Personality and Social Pshychology*, vol. 35, no. 12, pp. 847–862, 1977.

[12] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, UK: Cambridge University Press, 2000.

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

[14] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th International Conf. on Machine Learning*, pp. 282–289, Morgan Kaufmann, San Francisco, CA, 2001.

[15] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Underst.*, vol. 104, no. 2, pp. 90–126, 2006.

[16] J. Sullivan and S. Carlsson, "Recognizing and tracking human action," in *European Conference on Computer Vision*, 2002.

[17] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Conditional models for contextual human motion recognition," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, 2005.

[18] S. E. Golowich and D. X. Sun, "A support vector/hidden Markov model approach to phoneme recognition," in *ASA Proceedings of the Statistical Computing Section*, pp. 125–130, 1998.

[19] P. Clarkson and P. J. Moreno, "On the use of support vector machines for phonetic classification," *Compaq Computer Corporation, Cambridge Research Laboratory USA*, 1999.

[20] M. Bartlett, G. Littlewort, B. Braathen, T. Sejnowski, and J. Movellan, "A prototype for automatic recognition of spontaneous facial actions," in *Advances in Neural Information Processing Systems, NIPS 2003*, pp. 1271–1278, 2002.

[21] H. Shimodaira, K. ichi Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in support vector machine," in *Advances in Neural Information Processing Systems 14, NIPS2001*, pp. 921–928, 2001.

[22] D. Surendran and G.-A. Levow, "Dialog act tagging with support vector machines and hidden Markov models," in *Interspeech 2006 — ICSLP*, (Pittsburgh, PA, USA), 2006.

[23] D. Del Vecchio, R. M. Murray, and P. Perona, "Decomposition of human motion into dynamics-bbased primitives with application to drawing tasks," *Automatica*, vol. 39, no. 12, pp. 2085–2098, 2003.

[24] C. Lu and N. Ferrier, "Repetitive motion analysis: Segmentation and event classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 258–263, 2004.

[25] Y. Ivanov and A. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 852–872, 2000.

[26] M. Yamamoto, H. Mitomi, F. Fujiwara, and T. Sato, "Bayesian classification of task-oriented actions based on stochastic contxt-free grammar," in *Int. Conf. on Automatic Face and Gesture Recognition*, (Southampton, UK), April 10–12 2006.

[27] V. Kyrki, I. Serrano, D. Kragic, and J.-O. Eklundh, "Action recognition and understanding using motor primitives," in *In RO-MAN'07: The 16th IEEE International Symposium on Robot and Human Interactive Communication, Jeju Island, Korea*, 2007.

[28] S. Needleman and C. Wunsch, "A general method applicable to the search for similarity in the amino acid sequences of two proteins," *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.

[29] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, (Morristown, NJ, USA), pp. 134–141, Association for Computational Linguistics, 2003.