Chapter 4
Perceiving Objects and Movements to Generate Actions on a Humanoid Robot

Tamim Asfour, Kai Welke, Aleš Ude, Pedram Azad and Rüdiger Dillmann

4.1 Introduction

To deal with problems in perception and action researchers in the late 80s introduced two new frameworks, one under the heading of active vision (animate, purposive, behavioral) originating in the field of computer vision and the other in AI/robotics under the heading of behavior-based robotics. In both formalisms, the old idea of conceiving an intelligent system as a set of modules (perception, action, reasoning) passing results to each other was replaced by a new way of thinking of the system as a set of behaviors. Behaviors are sequences of perceptual events and actions. These efforts still go on, but only with limited success up to now. One reason for this is that although it was expected that active vision would make many perceptual problems easier, machine perception still remains rather primitive when compared to human perception. A further reason for failure is that behaviors were often designed ad hoc without studying the interplay between objects and actions in depth, which is necessary to develop structures suitable for higher-level cognitive processes. A third reason was that no one succeeded in formulating a general enough theory for behavior-based robotics. Hence, it remains difficult or even impossible to predict how a newly designed behavior-based system will scale and deal with new situations.

In recent years there are renewed efforts to develop autonomous systems and especially humanoid robots (see [7, 1, 11, 14, 12, 3]), i.e. (embodied) robots that perceive, move and perform (simple) actions. The successful attempts in this area are still limited to simple scenarios, very much for the same reasons mentioned

Tamim Asfour, Kai Welke, Pedram Azad and Rüdiger Dillmann
University of Karlsruhe, Institute for Computer Science and Engineering, Industrial Applications for Informatics and Microsystems (IAIM)), P.O. Box 6980, D-76128 Karlsruhe, Germany, e-mail: \{asfour,welke,azad,dillmann\}@ira.uka.de

Aleš Ude
Jožef Stefan Institute, Dept. of Automatics, Biocybernetics, and Robotics, Jamova 39, 1000 Ljubljana, Slovenia, e-mail: ales.ude@ijs.si

above. One should also note the extensive research on visual recognition and categorization in computer vision, which resulted in quite advanced and efficient recognition methods, although seldom tested on real world scenes. Moreover, the relationship between this research and the development of cognitive systems is still weak; since these approaches usually assume that what constitutes objects and categories is given a priori by the external world, they do not pertain to seeing agents and their actions. Research into cognitive robots should combine the study of perceptual representations that facilitate motor control, motor representations that support perception, and learning based on actively exploring the environment and interacting with people that provides the constraints between perception and action. This will then allow, e.g., to learn the actions that can be carried out on and with objects, which leads to what we call Object-Action-Complexes (OAC).

The concept of Object-Action Complexes (OACs) has been introduced by the European PACO-PLUS consortium ([8]) to emphasize the notion that for a cognitive agent objects and actions are inseparably intertwined and that categories are therefore determined (and also limited) by the action an agent can perform and by the attributes of the world it can perceive. The resulting OACs are the entities on which cognition develops (action-centered cognition). Entities (*things*) in the world of a robot (or human) will only become semantically useful *objects* through the action that the agent can/will perform on them.

In this work we present a new humanoid active head which features human-like characteristics in motion and response and mimics the human visual system. We present algorithms that can be applied to perceive objects and movements, which form the basis for learning actions on the humanoid. For action representation we use an HMM-based approach to reproduce the observed movements and build an action library. Hidden Markov Models (HMM) are used to represent movements demonstrated to a robot multiple times. They are trained with the characteristic features (key points) of each demonstration. We propose strategies for adaptation of movements to the given situation and for the interpolation between movements stored in a movement library.

4.2 Active Humanoid Head

The humanoid robot ARMAR III has been designed under a comprehensive view so that it can perform a wide range of tasks and not only a particular task. In the design of the robot, we desire a humanoid that closely mimics the sensory and sensory-motor capabilities of the human. The robot should be able to deal with a household environment and the wide variety of objects and activities encountered in it.

To achieve the above goals, we use an integrated humanoid robot consisting of a humanoid head with seven degrees of freedom (DOF), two arms (seven DOF per arm) and five-finger hands (eight DOF per hand), a torso with three DOF, and a holonomic mobile platform. In designing the robot, we desire a humanoid that closely mimics the sensory and sensory-motor capabilities of the human. Therefore, the

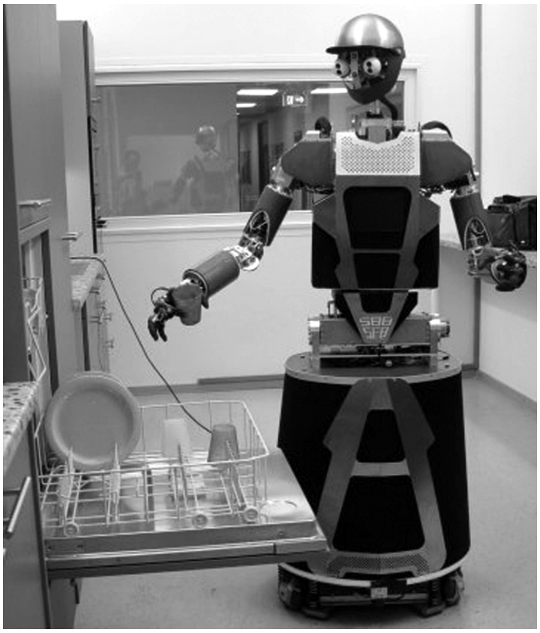


Fig. 4.1 The humanoid robot ARMAR-III. The has 43 DOF. From the kinematics control point of view, the robot consists of seven subsystems: head, left arm, right arm, left hand, right hand, torso, and a mobile platform.

robot is equipped with manipulative, perceptive and communicative skills necessary for real-time interaction with the environment and humans.

4.2.1 System Requirements

We pay special attention to the design of the head since the head can provide rich perceptual input necessary to realize various visuo-motor behaviors, e.g. smooth pursuit and saccadic movements towards salient regions, and also more complex sensory-motor tasks such as hand-eye coordination, gesture identification, human motion perception and linking of visual representations to the motor representations. The major design criteria were as follows:

- The robot head should be of realistic human size and shape while modelling the major degrees of freedom (DOF) found in the human neck/eye system, incorporating the redundancy between the neck and eye DOF.
- The robot head should feature human-like characteristics in motion and response, that is, the velocity of eye movements and the range of motion will be similar to the velocity and range of human eyes.

- The robot head must enable saccadic motions, which are very fast eye movements allowing the robot to rapidly change the gaze direction, and smooth pursuit over a wide range of velocities.
- The optics should mimic the structure of the human eye, which has a higher resolution in the fovea.
- The vision system should mimic the human visual system while remaining easy to construct, easy to maintain and easy to control.

With this set of requirements, a first version of the head have been developed as part of a humanoid robot that will allow for the integration of motor control and perception. This is essential to enable explorative head, hand, and body movements for learning of OACs.

4.2.2 Head Motor System

The head has seven DOF and is equipped with two eyes. Each eye can independently rotate about a vertical axis (pan DOF), and the two eyes share a horizontal axis (tilt DOF). To approximate These two DOF allow for human-like eye movements1. The visual system is mounted on a neck mechanism [1] with four DOF organized as pitch-roll-yaw-pitch.

4.2.3 Head Sensory System

To start learning object-action complexes we must, firstly, identify regions that potentially contain objects of interest and secondly analyze these regions to build higher-level representations. While the first task is closely related to visual search and can benefit from a wide field of view, a narrower field of view resulting in higher-resolution images of objects is better suited for the second task. While the current technology does not allow us to exactly mimic the features of the human visual system and because camera systems that provide both peripheral and foveal vision from a single camera are still experimental, we decided for an alternative which allows to use commercially available camera systems that are less expensive and more reliable. Foveated vision was realized using two cameras per eye, one with wide-angle lens for peripheral vision and one with narrow-angle lens for foveal vision. We use the Point Grey Research Dragonfly IEEE-1394 camera in the extended version (www.ptgrey.com). The extended version allows the CCD to be up to 6 inches away from the camera interface board. This arrangement helps with accessing hard to reach places, and with placing the lens into a small volume. Since the cameras are very light and are extended from the interface board by a flexible extension cable, they can be moved with small and low-torque servos.

The cameras can capture colour images at a frame rate of up to 30 Hz. They implement the DCAM standard, and transmit a raw 8 bit Bayer Pattern with a resolution of 640x480, which is then converted on the PC to a 24 bit RGB image. The cameras have a FireWire interface, which is capable of delivering data rates of up to 400 Mbps. The benefit of transmitting the Bayer Pattern is that only a third of the bandwidth is needed for transmitting the colour image without loosing any information. Thus, it is possible to run one camera pair at a frame rate of 30 Hz and the other at a frame rate of 15 Hz, all being synchronized over the same FireWire bus, without any additional hardware or software effort. Running the foveal cameras, which have a smaller focal length and thus a narrower view angle, at a lower frame rate is not a drawback because these cameras are not crucial for time critical applications such as tracking, but are utilized for detailed scene analysis, which does not need to be performed at full frame rate in most cases anyway. The camera is delivered as a development kit with three micro lenses with the focal lengths 4, 6, and 8mm. In addition, one can use micro lenses with other focal lengths as well. We have chosen a 3 mm micro lens for the peripheral cameras and a 16 mm micro lens for the narrow angle cameras. Furthermore, the head is equipped with six microphones (SONY ECMC115.CE7): two in the ears, two in the front and two in the rear of the head. These microphones will be used in the later phase of the project to achieve a richer multi-sensory representation of objects and environment and to support the integration of speech components in order to provide an additional information for interaction and natural communication.

4.3 Perceiving Objects and Movements

4.3.1 Human Motion Tracking

For the tracking of human motion, an image-based markerless human motion capture system has been developed[6, 5]. The input of the system are stereo colour images of size 320×240 captured at 25 Hz, with two calibrated Dragonfly cameras built-in into the head of the humanoid robot ARMAR III. The input images are pre-processed, generating output for the gradient cue, the distance cue, and an optional region cue, as described in [5]. Based on the output of the image processing pipeline, a particle filter is used for tracking the movements in configuration space. The overall likelihood function to compute the a-posteriori probabilities is formulated as:

$$p(\mathbf{z}|\mathbf{s}) \propto \exp\left\{ -\frac{1}{2} \left(\frac{1}{\sigma_d^2} \sum_{i=1}^{3} d_i(\mathbf{s}, \mathbf{c}_i) + \frac{1}{\sigma_g^2} (1 - \frac{1}{M_g} \sum_{m=1}^{M_g} g_m) \right) \right\}, \tag{4.1}$$

where $\mathbf{s}$ is the configuration to be evaluated, $\mathbf{z}$ is a general denotation for the current observations i.e. the current input image pair, and $\mathbf{c}_i \in \mathbb{R}^3$ with $i \in \{1,2,3\}$ denotes the triangulated 3D position of the hands and the head. The function $d_i(\mathbf{s}, \mathbf{c})$ is defined as:

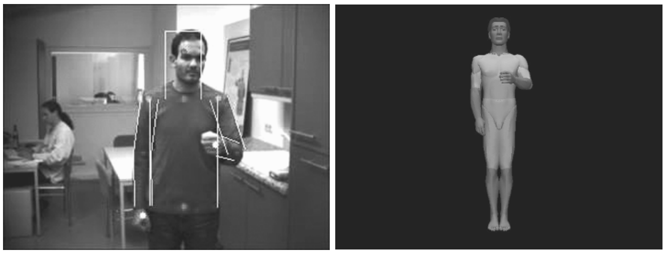


Fig. 4.2 Illustration of the performance of the markerless human motion capture system. Left: projection of the estimated configuration into the left camera image. Right: 3D visualization of the estimated configuration with an articulated human model.

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

where $n := \dim(\mathbf{s})$ is the number of DOF of the human model. The transformation $f_i : R^n \to R^3$ 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 g_m with $m \in \{1,2,...,M_g\}$ denote the intensity values in the gradient image (which is derived from the input images $\mathbf{z}$) at the M_g pixel coordinates of the projected contour of the human model for a given configuration $\mathbf{s}$. This process is performed for both input images using the calibration parameters of each camera. For each image pair of the input sequence the output of the system is the estimation of the particle filter, given by the weighted mean over all particles. A detailed description is given in [5].

In contrast to the acquisition method based on the magnetic tracking system, the joint angle values θ_3, θ_4, θ_5, and θ_6 are calculated *directly* and therefore the position of the elbow does not have to be approximated based on empirical studies but is determined explicitly.

4.3.2 Object Representations for Actions

Our scheme of object representation is driven by the conviction that objects and actions are inseparably intertwined. To facilitate the execution of complex actions in the currently perceived environment, we want our system to learn performing actions on objects in two ways: learning by demonstration and learning by exploration. In the following section we want to emphasize learning by exploration and the consequences for an action related object representation scheme.

While autonomously exploring possible actions on an object and finally associating successful actions with an object, the robot retrieves a set of object action relations. The related actions for an object can be considered object affordances[9]. Associating possible actions to prior percepts only will not result in a general representation of object affordances. Therefore, a mechanisms is necessary which allows the determination of affordances for unknown percepts on the basis of previously experienced object action relations. Below we will introduce a conceptual way to generalize object action relations to a representation, which can be used to determine the affordance of an unknown percept.

Furthermore, to allow autonomous exploration of objects, a mechanism to measure success for an executed action is necessary. This ability can not be learned in a completely autonomous way. For a new action the system initially needs a feedback, whether an action executed on an object has been successful or not. This can be provided either by an assistant who judges the action after execution or by demonstrating successfully executed actions. Once the system learned this measure of success, it can judge itself if the execution of an action on an unknown object was successful or not.

A system which allows both, generalizing for object affordances and learning of how to measure success, has to rely on two distinct sets of object features: features which are stable and features which are varying during execution. If we consider the example of filling a cup, the shape and colour of the cup itself will be stable during action execution, while the fill level changes. In the following, we denote features of an object which are stable during action execution by $F(P) = (f_1, \ldots, f_{N_f})$ and features that are varying during execution with $G(P) = (g_1, \ldots, g_{N_g})$. Each feature vector component (f_n or g_n) will be called feature channel.

Considering the invariant feature vector $F(P)$ for an action A performed on an percept P, it is clear that the affordance $\mathit{afford}_A(P)$ has to be triggered by elements of $F(P)$. To determine, which feature channels are responsible for the affordance, the system has to acquire enough experience with the action A on different percepts $P_1, \ldots, P_N$ and to generalize over the resulting vectors $F(P_1), \ldots, F(P_N)$. During the generalization process, two things will be determined: the relevance R_n of each feature channel f_n for the affordance and the feature values for all channels, which frequently co-occur with the action and thus are strong indicators for the affordance. To determine significant domains in feature space, clustering is performed for each feature channel using all previous percepts which are associated with successful executions of the action. For feature channel n, the resulting clusters are combined in an extended signature containing the cluster centroid $c_{n,i}$, the number of samples $w_{n,i}$ associated to the cluster, and the distance from the cluster's centroid to the farthest cluster element $d_{n,i}$:

$$S_n = \{s_i = (c_{n,i}, w_{n,i}, d_{n,i})\} \tag{4.2}$$

For each cluster the probability $p_{n,i}$ is assigned which captures how probable an element which belongs to the cluster affords the action:

$$p_{n,i} = \frac{w_{n,i}}{N} \tag{4.3}$$

A cluster with high probability $p_{n,i}$ shows that the corresponding feature channel captures relevant information for the object affordance. We calculate the relevance

R_n of a feature channel n by:

$$R_n = p_{n,max} \tag{4.4}$$

where *max* is the index of the cluster center with largest number of elements $w_{n,i}$.

With the probabilities and the feature channel relevance, the affordance of an unknown percept X can be determined. First the invariant feature vector $F(X)$ is calculated. For each feature vector component the closest cluster $c_{n,r}$ is searched. The distance e_r to the centroid is calculated and used to determine if the component is significant for the cluster. To express this in our calculations, we define a binary function of significance:

$$k_n = \begin{cases} 1 \text{ if} e_r < \alpha d_{n,r} \\ 0 \text{ otherwise} \end{cases} \tag{4.5}$$

The affordance of the percept X for the action A can then be calculated by:

$$afford_A(X) = \frac{\sum_{n=0}^{N_f} k_n p_{n,r}}{\sum_{n=0}^{N_f} R_n} \tag{4.6}$$

The system has to hold an inner model which allows to determine affordances for percepts. For the calculations we only need the extended signatures S_n. To keep only relevant channels in the inner model, we threshold the relevance R_n of each channel and discard channels with low relevance. Channels with low relevance will usually have many small clusters and discarding them helps in keeping the inner model small. Thus as inner model for affordances *IA* for the action A we can write:

$$IA_A = \{(S_n) : R_n > minrelevance\} \tag{4.7}$$

Once an action has been performed on the object, the robot has to determine if the action was successful. For this, a measure of success which relates percepts prior to execution with percepts after execution is necessary. The action is considered as continuous process over time. Thus the change of a prior percept P_0 to a percept during action execution P_t can be written in the following way:

$$P_t = C_A(P_0, t) \tag{4.8}$$

where C_A describes the change of the percept when applying the action A. The success can be measured between two percepts $P_{t_i}, P_{t_{i+1}}$, where the interval $\Delta t = t_{i+1} - t_i$ is large enough to perceive the change triggered by the action. The invariant features $F(P)$ are not relevant for the measuring of success. We use the varying feature set $G(P)$ of previously perceived successful action executions as input to the measurement. Since we want to measure a relation of percepts between points in time, we observe the difference between the feature sets $g_n(P_{t+\Delta t}) - g_n(P_t) = d_n(t)$. The generalization of $d_n(t)$ is performed in a similar way as mentioned above. During the generalization process the expected values $E_n(t) = \{e_{n,i}(t)\}$ which correspond to cluster centers and relevances of feature channels R_n are determined. We assume, that all observed changes $d_n(t)$ for a relevant feature channel have the same course in time for actions applied to different percepts. This has to be ensured in a normalization step where all $d_n(t)$ are mapped to a common time basis and has to be taken into account during generalization.

In the inner model for the measurement of success IM_A the expected values $E_n(t)$ are stored, if the corresponding relevance is above a threshold:

$$IM_A = \{(E_n(t)) : R_n > minrelevance\} \tag{4.9}$$

Critical in the realisation of the proposed scheme is the implementation of the feature extractors G and F. For the example of cup filling we use the problem specific features shape and colour as invariant features ($F(P) = (f_{shape}, f_{colour})$) and fill level as varying feature ($G(P) = g_{filllevel}$). Future challenges comprise the proposal of feature extraction methods, which follow the requirements formulated in this section for a broader range of problems. The proposed relevances allow to evaluate methods on their applicability for the extraction of affordances and the measurement of success for an action.

4.4 Action Representation

Our approach to generate, represent and reproduce actions makes use of Hidden Markov Models. We use three different HMM for each arm, one to encode the position of the TCP (Tool Center Point, a reference point on the hand), i.e. the hand path, with the Cartesian coordinates being represented by three-dimensional output distributions, one for the orientation of the TCP (described by three angles) and another one for the joint angle trajectories where the dimension of the output distributions is equal to the number of observed joint angles of the arm.

HMMs are used to generalize movements demonstrated to a robot multiple times [2]. Characteristic features of the perceived movement, so-called key points, are detected in a pre-processing stage and used to train the HMMs. By doing so, we avoid having a high number of states and facilitate the matching of (or between) multiple demonstrations. We use continuous HMMs and model the observations in each state with multivariate Gaussian density functions. Each HMM is trained with the key points of all demonstrations using the Baum-Welch algorithm for multiple observations. Each training sequence consists of the key points of the respective demonstration. For a given observation sequence, the Viterbi algorithm returns the optimal state sequence of the HMM with respect to that observation sequence, i.e. the sequence of states most likely to generate that observation sequence. For the reproduction of a perceived movement, key points that are common to all (or almost all) demonstrations, so-called common key points, are used. To determine the common key points across $d = 1, \dots, D$ key point sequences $K_{d,1}, \dots, K_{d,n(d)}$, where $n(d)$ denotes the number of key points for a demonstration d, we use the Viterbi algorithm D times to find sequences of HMM states that correspond best to these key point sequences. The common key points are determined by comparing these

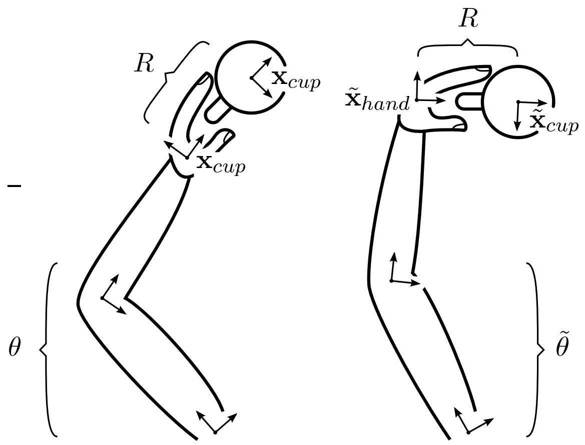


Fig. 4.3 Two arm postures grabbing a cup in the same way. It is obvious that the relation R to the object is much more important than the joint angles θ.

state sequences and selecting only those states that appear in every sequence. The actions that were considered initially are simple actions. In general, it is not possible to learn an HMM for all possible action imaginable. To treat a large set of complex actions, we will need to brake down the actions into very simple ones. These simple actions would define an alphabet based on which complex actions can be defined by concatenation.

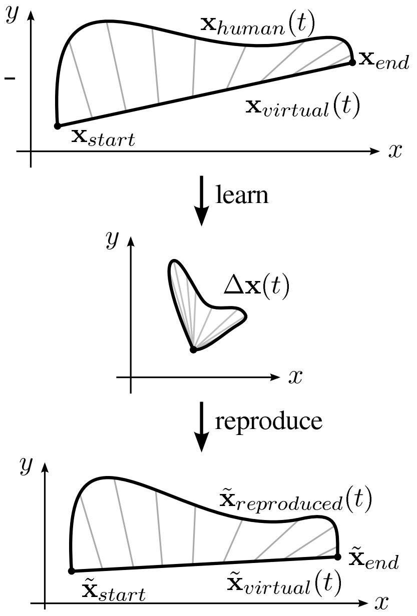
4.5 Imitation on Objects

Learning trajectories both in the work space and in the joint space is one common approach in imitation learning approaches. Movement are often demonstrated to a robot by a human instructor, subsequently generalized (using the data from all demonstrations) and finally reproduced by the robot without trying to infer the goal of the movement. One main goal of imitation learning is to understand simple movements. Taking the example of grabbing a cup, the path has to be altered if the cup has a different position. Static learning does not fulfil these needs. In figure 4.3 one can see that for this example the exact arm posture is less important than the relation to the effected object. The joint trajectory is only a minor condition. It should not be used to calculate the hand position. Instead it could be used to solve the problem of the redundancy in computing the inverse kinematics.

In this section we present an novel way for imitation learning on objects.

4.5.1 Adaptation of Movements to the Given Situation

If an action is repeated in a different situation, the learned path has to be adjusted. Our idea is to learn paths only relative to the affected object (see Fig. 4.4). While a new trajectory is processed the linear path between $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_{\text{end}}$ is calculated. For the adaptation, the system is trained with the difference $\Delta\mathbf{x}$ between the observed and the linear path. The reproduction is done by using the linear path between the new $\tilde{\mathbf{x}}_{\text{start}}$ and $\tilde{\mathbf{x}}_{\text{end}}$ and the learned difference. The result would be a similar path into a different direction.



y
$\mathbf{x}_{human}(t)$
$\mathbf{x}_{end}$
$\mathbf{x}_{virtual}(t)$
$\mathbf{x}_{start}$
x
learn
y
$\Delta\mathbf{x}(t)$
x
reproduce
y
$\tilde{\mathbf{x}}_{reproduced}(t)$
$\tilde{\mathbf{x}}_{end}$
$\tilde{\mathbf{x}}_{start}$
$\tilde{\mathbf{x}}_{virtual}(t)$
x

Fig. 4.4 Adaptation of an observed movement to the given new situation. Instead of the original sequence, the gray indicated difference to the linear path is learned. The normalized sequence is extracted to new parameters in the lower diagram.

Using the linear path as reference is the easiest and fastest possibility. The quality will be increased if the calculated path is already human-like. A system like the VITE model created by Hersch and Billard [10] seems to be convenient. The joint angle information of such a system would be ignored. Only the hand path would be used as a basis of the difference.

4.5.2 Generalization Across Movements

The methodology described in Section 4.4 allows us to reproduce the observed movements by a humanoid robot and build a library specifying a complete set of the desired trajectories for an observed action class. Motion capture has been used successfully to reproduce motions that may require a lot of skill and practicing, but do not need to fulfill a specific goal, such as for example dancing [15]. However, in tasks involving the manipulation of objects, it is often necessary to adapt the observed trajectories with respect to the current state of the 3-D world. It is highly unlikely that an appropriate movement would be observed a priori and included in the library. Hence it is necessary to generalize over the movements stored in the library and generate a new movement that can attain the goal of an action. With this in mind we designed a strategy to interpolate between movements stored in the library, with the goal of generating appropriate new movements that were not recorded in the data collection phase, but need to be executed to attain the goal of an action.

Let each example motion $\mathbf{M}_i,\ i = 1,\dots,NumEx$, be given by key points $\mathbf{p}_{ij}$ at times $t_{ij},\ j = 1,\dots,n_i,$. With each movement we also store the duration of motion T_i. Such data can be collected by the proposed motion capture system. The key points can be specified in various ways, for example as joint space postures or as end-effector poses in 3-D. We start by time normalizing the captured movements to an interval $[0,1]$. Similarly to Rose et al. [13], we encode the example trajectories using uniform cubic B-splines

$$\mathbf{M}_i(t) = \sum_{k=1}^{N} \mathbf{b}_{ik} \mathrm{B}_k(t), \tag{4.10}$$

where N is the number of spline basis functions. Linear least squares approximation can be used to approximate all trajectories with the same number of splines. The optimal number of splines N can be determined experimentally, but more sophisticated methods are also possible (see for example [15]).

In the following we propose a method for the generation of goal-directed arm reaching movements. The method is, however, much more general and we discuss how to apply it to other actions at the end of the section.

In the case of arm reaching movements, the start point $\mathbf{x}_{\text{start}}$ and the end point $\mathbf{x}_{\text{end}}$ of the end-effector in Cartesian space are very important. Therefore it makes sense to represent goal-directed arm reaching movements as end-effector trajectories $\mathbf{M}_i$ in a 3-D space. Another important factor is the duration of movement T. We use this information as a query point $\mathbf{q}$ into the database when generating new movements from example movements

$$\mathbf{q} = \left[\mathbf{x}_{\text{start}}^T, \mathbf{x}_{\text{end}}^T, T\right]^T. \tag{4.11}$$

Given a query point $\mathbf{q}$, we would like to determine movement $\mathbf{M}(\mathbf{q};t)$ defined as

$$\mathbf{M}(\mathbf{q};t) = \sum_{k=1}^{N} \mathbf{b}_k(\mathbf{q}) \mathrm{B}_k(t), \tag{4.12}$$

which starts at $\mathbf{x}_{\text{start}}$ and ends at $\mathbf{x}_{\text{end}}$. For each of the example trajectories $\mathbf{M}_i$, we calculate its start point $\mathbf{x}_{i,\text{start}}$ and its end point $\mathbf{x}_{i,\text{end}}$. We apply locally weighted regression [4] to generate new reaching movements. This results in the following optimization problem

$$\min_{\mathbf{b}} \mathrm{C}(\mathbf{q}) = \sum_{i=1}^{NumEx} \mathrm{L}\left(\mathbf{M}_i, \mathbf{M}(\mathbf{q})\right) \mathrm{K}\left(\mathrm{d}_i(\mathbf{q}_i, \mathbf{q})\right), \tag{4.13}$$

subject to

$$\mathbf{M}(\mathbf{q};0) = \mathbf{x}_{\text{start}},\ \mathbf{M}(\mathbf{q};1) = \mathbf{x}_{\text{end}}. \tag{4.14}$$

Here L is the loss function, K is the weighting function, and d_i are the distance functions between the query point and the data points $\mathbf{q}_i = \left[\mathbf{x}_{i,\text{start}}^T, \mathbf{x}_{i,\text{end}}^T, T_i\right]^T$. The unknown parameters we minimize over are $\mathbf{b} = \{\mathbf{b}_k(q)\}$.

We define the loss function by the Euclidean distance between the spline coefficients

$$\mathrm{L}\left((\mathbf{M}_i, \mathbf{M}(\mathbf{q})\right) = \sum_{k=1}^{N} \|\mathbf{b}_{ik} - \mathbf{b}_k(\mathbf{q})\|^2. \tag{4.15}$$

Distance function d_i is given as the weighted Euclidean distance between the data points, i. e.

$$\mathrm{d}(\mathbf{q}, \mathbf{q}_i) = \frac{1}{a_i}\|\mathbf{q} - \mathbf{q}_i\|,\ a_i > 0. \tag{4.16}$$

There are many possibilities to define the weighting function K [4]. We chose the tricube kernel

$$\mathrm{K}(d) = \begin{cases} (1 - |d|^3)^3 & \text{if} |d| < 1 \\ 0 & \text{otherwise} \end{cases}. \tag{4.17}$$

This kernel has finite extent and continuous first and second derivative. Combined with distance (4.16), these two functions determine how much influence each of the movements $\mathbf{M}_i$ has as the query point $\mathbf{q}$ moves away from the data point $\mathbf{q}_i$. It is best to select a_i so that there is some overlap between the neighboring query points. One possibility is

$$a_i = \min_j \|\mathbf{q}_i - \mathbf{q}_j\| \tag{4.18}$$

By selecting a_i in this way we ensure that the influence of neighboring movements in (4.13) overlaps, that $\mathbf{M}(\mathbf{q}_i) = \mathbf{M}_i$, and that as the query point transitions from one data point to the other, the generated movement also transitions between movements associated with data points.

Our choice of L, K, and d_i makes the optimization problem (4.13) a weighted linear least-squares problem with equality constraints, which can be solved using standard approaches. In this way we can generate new arm reaching movement that were not observed in the data collection phase. It can be clear from the above explanation that the method is not limited to arm reaching movements. For a given collection of movements, it is only necessary to specify reasonable query points and impose any constraints that are necessary to achieve the goal of an action. The proposed movement interpolation technique can then be applied.

4.6 Discussion and Conclusions

As the goal of an action changes it is necessary to adapt the captured movements to new situations. Sometimes it is possible to attain the goal of an action by moving and scaling the desired trajectories in space and time. For this purpose we propose a method for adaptation of movements to the given situation (Sec. 4.5.1). In some situations, however, the movement changes more substantially depending on the goal of an action, e.g. the amplitude could increase or the frequency of oscillation could change. Such modification cannot be captured by the first approach, therefore we introduce a strategy to interpolate between movements in stored in the movement library (Sec. 4.5.2).

Future work will concentrate on both the evaluation of the proposed methods for the generation of actions and a complete implementation of a real-time imitation learning system using the active humanoid head.

Acknowledgements 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).

References

1. Akachi, K., Kaneko, K., Kanehira, N., Ota, S., Miyamori, G., Hirata, M., Kajita, S., Kanehiro, F.: Development of humanoid robot HRP-3. In: IEEE/RAS International Conference on Humanoid Robots (2005)
2. Asfour, T., Gyarfas, F., Azad, P., Dillmann, R.: Imitation learning of dual-arm manipulation tasks in humanoid robots. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006). Genoa, Italy (2006)
3. Asfour, T., Regenstein, K., Azad, P., Schröder, J., Bierbaum, A., Vahrenkamp, N., Dillmann, R.: ARMAR-III: An integrated humanoid platform for sensory-motor control. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006). Genoa, Italy (2006)
4. Atkeson, C.G., Moore, A., Schaal, S.: Locally weighted learning. AI Review **11**, 11–73 (1997)
5. Azad, P., Ude, A., Asfour, T., Cheng, G., Dillmann, R.: Image-based Markerless 3D Human Motion Capture using Multiple Cues. In: International Workshop on Vision Based Human-Robot Interaction. Palermo, Italy (2006)
6. Azad, P., Ude, A., Dillmann, R., Cheng, G.: 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)
7. Cheng, G., Hyon, S., Morimoto, J., Ude, A., Jacobsen, S.: CB: a humanoid research platform for exploring neuroscience. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006). Genoa, Italy (2006)

8. The PACO-PLUS project: Perception, Action and Cognition through Learning of Object-Action Complexes. European Cognitive Systems project, www.paco-plus.org.
9. Gibson, J.: The ecological approach to visual perception. Houghton Mifflin, Boston (1979)
10. Hersch, M., Billard, A.: A biologically-inspired model of reaching movements. In: IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (2006)
11. I.W. Park J.Y. Kim, J.L., Oh, J.: Mechanical design of humanoid robot platform KHR-3 (kaist humanoid robot-3: Hubo). In: IEEE/RAS International Conference on Humanoid Robots (2005)
12. Nishiwaki, K., Sugihara, T., Kagami, S., Kanehiro, F., Inaba, M., Inoue, H.: Design and development of research platform for perception-action integration in humanoid robots: H6. In: IEEE/RSJ International. Conference on Intelligent Robots and Systems, pp. 1559–1564 (2000)
13. Rose, C., Bodenheimer, B., Cohen, M.F.: Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. IEEE Computer Graphics and Applications **18**(5) (1998)
14. Sakagami, S., Watanabe, T., Aoyama, C., Matsunage, S., Higaki, N., Fujimura, K.: The intelligent ASIMO: System overview and integration. In: IEEE/RSJ International. Conference on Intelligent Robots and Systems, pp. 2478–2483 (2002)
15. Ude, A., Atkeson, C.G., Riley, M.: Programming full-body movements for humanoid robots by observation. Robotics and Autonomous Systems **47**(2-3), 93–108 (2004)