Project no.:	**027657**
Project full title:	**Perception, Action & Cognition through Learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**

Deliverable no.: D8.2.1

Title of the deliverable: Augmented Human Action

Contractual Date of Delivery to the CEC:	**31. January 2007**
Actual Date of Delivery to the CEC:	**31. January 2007**
Organisation name of lead contractor for this deliverable:	**UniKarl**

Author(s): Rüdiger Dillmann, Tamim Asfour, Pedram Azad, Aleš Ude, Gordon Cheng, Danica Kragic, Volker Krüger, Jan-Olof Eklundh, Florentin Wörgötter, Norbert Krüger, Bernhard Hommel, Carme Torras, and Mark Steedman.

Participants(s):	**UniKarl, KTH, JSI, AAU, BCCN, CSIC, UEDIN, UL**
Work package contributing to the deliverable:	**WP8.2**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**66**
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:

The goal of WP8.2 is to develop an artificial cognitive system acting in a human like environment and capable of learning objects and the actions it can apply to them. This report summarizes the work in WP8.2 in the first twelve months. It reports on the development and implementation of human motion capture and motion mapping techniques necessary for the realisation of a cognitive system acting in a human like environment and capable of learning objects and the actions it can apply to them. The report presents result related to human motion tracking, representation, and recognition of 3D human motion as well as imitation learning of human actions and coaching. Furthermore, mapping strategies between the human hand and different hands are described. To support easy design of manipulation and exploration tasks, an integrated approach that includes an offline grasp planning system with a visual object recognition system is presented.

Keyword list: Human motion tracking, imitation learning, coaching and grasp mapping.

Table of Contents

1. Introduction

Using artificial cognitive systems to enhance or substitute human actions is and will become even more an important field in an ageing society and generally in a "high-tech" society. However, it requires truly cognitive capabilities of the artificial agent including learning of scenarios, objects, and actions as well as communication and interaction with humans. The goal of WP8.2 "*Augmenting Human Action*" is to develop an artificial cognitive system acting in a human like environment capable of learning objects and the actions it can apply to them. The focus is on structuring actions by learning, imitating, and understanding actions of humans, in contrast to the emphasis on exploration in WP8.1. The system should be able to perform learned actions over categories of objects and joint actions with other agents, such as reaching. In the longer term it should be able to communicate with other agents and perform actions as communicated.

WP8.2 and Demo 2 deal with the representation, recognition, and adaptation of OACs in an imitation and interaction based framework. Main scientific issues are the capturing of human motion, the modelling and representation of human motion (both arm movements and hand grasps), the generating of early sensory and motor representations of actions by imitation and coaching as well as action representation, understanding and imitation of actions.

2. Markerless Human Motion Capture

For the purpose of acquiring human motion data for imitation learning in real world scenarios, we have developed a purely image-based markerless human motion capture system with a particle filter in its core, as done in ([1], [2]). The problem is that the number of required particles grows exponentially with the dimensionality of the search space, which reduces the processing rate of the system drastically. For example, in [1], processing an input image tuple requires 15 seconds on a 1 GHz CPU. Other problems arise from the restriction of using the cameras incorporated in the robot head only, resulting in a relatively small stereo baseline and the occurrence of occlusions, since the person can be seen from essentially one viewpoint only. Furthermore, the system has to initialize itself automatically. Therefore, we have adapted and extended the existing particle filter approach for real-time application on a humanoid robot head.

The input of the system is a stereo colour image pair captured at 30 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 [A]. 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-posterior probabilities is formulated as:

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

where $\boldsymbol{s}$ is the configuration to be evaluated, $\boldsymbol{z}$ is a general denotation for the current observations i.e. the current input image pair, and $\boldsymbol{c}_i \in R^3$ with $i \in \{1, 2, 3\}$ denotes the triangulated 3D position of the hands and the head. The function $d_i(\boldsymbol{s}, \boldsymbol{c})$ 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 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, and $n = \dim(\boldsymbol{s})$ denotes the number of DoF of the human model.

The g_m with $m \in \{1, 2, ..., M_g\}$ denote the intensity values in the gradient image (which are derived from the input images $\boldsymbol{z}$) at the M_g pixel coordinates of the projected contour of the human model for a given configuration $\boldsymbol{s}$. 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 [A]. In contrast to the acquisition method based on the magnetic tracking system, the joint angle values θ_1, θ_2, θ_3, and θ_4 are calculated *directly* and therefore the position of the elbow does not have to be approximated on the base of empirical studies but is determined explicitly.

The system can process a stereo image sequence with a resolution of 320×240 at 15 Hz. We have also run successful experiments on videos captured with a resolution of 640×480 at a frame rate of 60 Hz. With such a frame rate, the system can capture much faster movements. To allow real-time application, the particle filter will be distributed on a Distributed Vision Cluster, as done in [4] in the case of hand motion tracking.

Figure 1: Example snapshots of a tracked image sequence. Top: Left input image with projection of the calculated configuration of the human model. Bottom: 3D visualization of the calculated configuration.

Based on the algorithms described above, a hnad motion capture system has been developed in [4]. Several adaptations had to be performed, such as the implementation of a z-buffer approach to handle self-occlusions, and the partitioning of the search space by tracking the hand pose and the finger poses independently. The results of the two particle sets are fused at the end of one particle filter run in one particle set before resampling the new particle set.

An effective way to constrain the search space and thus to limit the number of particles is to employ an action model: Given already observed poses, an action model allows to compute the likelihood for a particular new pose. Such an action model is subject of investigation in work package WP3 and will be built-in into the human motion capture in the near future.

We are planning to evaluate the visual tracking system with a video database of seven individuals showing different actions. The data has been captured with four synchronized video cameras, running at 30 Hz. Ground-truth information for the movements is provided through electro-magnetic devices that were attached to the joints of each volunteer. A comparison between the ground-truth data and the visually captured data will allow a) an evaluation of the visual tracking system and b) give us a valuable estimation of the noise our motion models (WP3) will have to deal

with when applied to visual tracking. Furthermore, the marker data will serve as an additional source for the training and evaluation of the action recognition system which is under development. By having several sources of human motion capture data (VICON, magnetic tracking, markerless visual tracking), together with the exchange format defined by the Master Motor Map (Section 3), we will be able to develop and evaluate an action recognition system, which is independent from the data source.

The dataset is available at the Aalborg University under the link http:cvmi.aau.dk/~pradeep/motion (login: “pacoplus”, passwd: “pacoaccess”). For a detailed description, please see Deliverable D3.1.2.

3. Master Motor Map (MMM)

Within the consortium not only one human motion capture system and one action recognition system exist, but several approaches are available, each of those feasible for another case, having its own strengths and drawbacks. Moreover, although the main demonstrator is ARMAR III (see [3]) at the University of Karlsruhe, other partners have to be able to use data and run experiments on their own demonstrator. To overcome the data compatibility problems arising form this circumstance, we have specified the so-called Master Motor Map (MMM) (see [B]).

The MMM is a full specification of a high dimensional but yet simplified kinematics model of a human, having 52 DoF. The names and the sequence of the joints have been chosen according to the H-ANIM 1.1 specification. The MMM adds a detailed specification of the rotation matrices and their orientation with respect to each other. In [B], we have shown that motion capture data from various systems can be converted to the MMM. Among these are the markerless human motion capture system, which has been explained above, and a marker-based motion capture system using the VICON system [5]. Any action recognition system can derive its desired data format from the MMM format, using forward kinematics if needed. Also, we have shown how the MMM format can be transformed to the kinematics of a humanoid robot, using the example of ARMAR III. An application for 3D visualization of the MMM and an archive containing over 400 captured trajectories is available on: http://wwwiaim.ira.uka.de/users/asfour/mmm/.

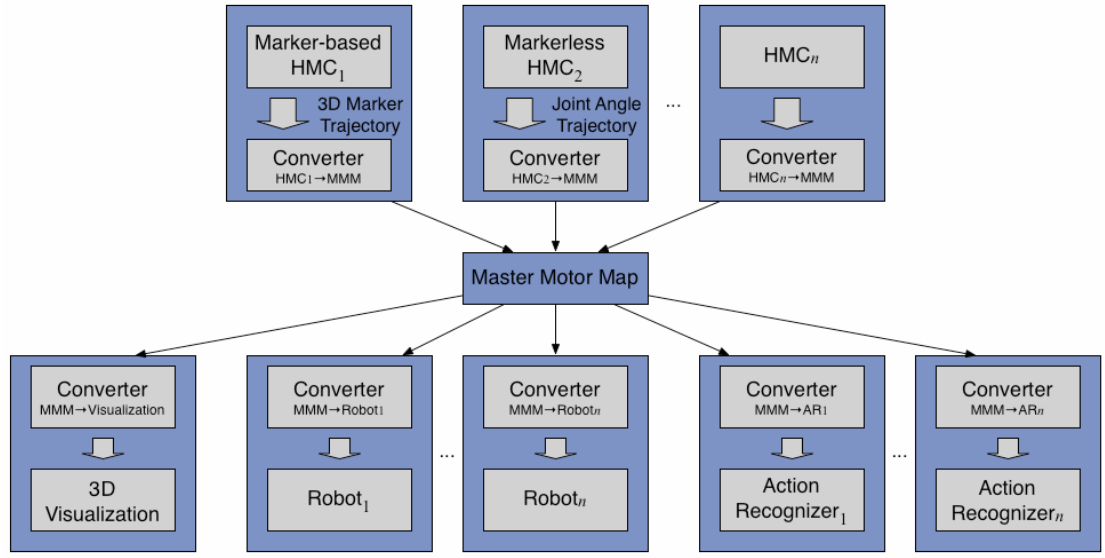


Figure 2: Framework for an unified action representation, recognition and imitation. Each component (HMC, robot, action recognizer, ...) has a converter module, either converting from or to the MMM format.

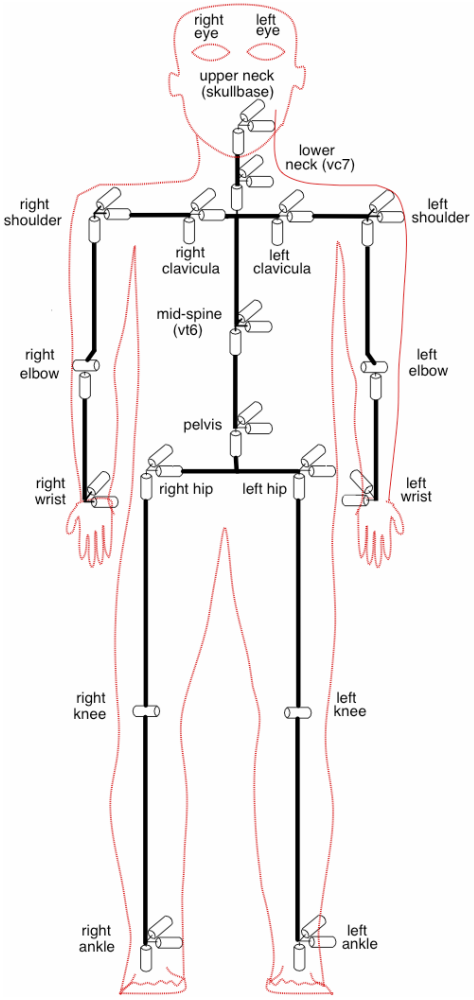


Figure 3: Specification of the kinematics model of the human body (MMM)

4. Imitation learning

Imitation Learning facilitates teaching a robot new tasks and at the same time make the robot move like a human. Imitation learning is basically the concept of having a robot observe a human instructor performing a task and imitating it when needed. Robot learning by imitation, also referred to as *programming by demonstration*, has been dealt with in the literature as a promising way to teach humanoid robots. Several imitation learning systems and architectures based on the perception and analysis of human demonstrations have been proposed ([6]-[12]). In most architectures, the imitation process proceeds through three stages: perception/analysis, recognition, and reproduction [13]. An overview of the basic ideas of imitation learning in robots as well as humans is given by Schaal in [14].

In [C], we presented an HMM-based approach for imitation learning of arm movements in humanoid robots. HMMs are used to generalize movements demonstrated to a robot multiple times. 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 $d = 1,\ldots,D$ key point sequences $K_{d,1}, \ldots, K_{d,n(d)}$, 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 state sequences and selecting only those states that appear in every sequence. Common key points are used for the reproduction of movements on the MMM kinematics model (see Section 2).

The actions that were considered initially in [C] 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. This is an important part in Demo 2 and is investigated in WP3.

5. Coaching

Beyond imitation and the associated observation of human motion, we are interested in other means of transferring human knowledge to a humanoid robot. Not surprisingly, most approaches for creating or modifying behaviours for complex humanoids require specialized knowledge and a large amount of work. Our aim is to provide an alternative, intuitive way to program humanoid behaviour. To do this, we examine human-to-human skill transfer, specifically coaching, and adapt it to the humanoid setting. We enable a real-time scenario where a person, acting as a coach, interactively directs humanoid behaviours to a desired outcome. This tightly coupled interaction between a person and a humanoid allows efficient, directed learning of new behaviours, where behaviour characteristics can be modified on demand. Communication is realized through demonstration and a coaching vocabulary, and changes are effected by transformation functions acting in the behaviour domain. In [D], we introduce a method with potential to improve the time and ease of creating behaviours for complex humanoid robots and describe the use of high-level vocabulary to initiate transformation functions to change the behaviours. A transformation function is typically comprised of a label, which is the coaching command that invokes it, and a set of criteria that serves to define the high level command in terms of low level behavioural criteria. Label and criteria are wrapped together in a function that ultimately effects changes to the appropriate behavioural parameters in accordance with the transformation function's definition.

6. Applying actions to Objects

Most of the robot tasks involve object manipulation. Hence, the robot has to learn how objects should be grasped and manipulated. The proper grasp type depends both on the target object and the current task. As an example, let us assume that the task is to pick up a cup to fill it with coffee. Generating suitable grasps based solely on the object shape provides three possible grasps: a circular sphere grasp, a power wrap grasp, and a two-finger-thumb precision grasp as seen in **Figure 4:** . However, the task knowledge introduces additional constraints because, in order to fill the cup, the opening should not be covered. In this case, only two grasps remain feasible.

Figure 4: Different grasps are possible for picking up a cup (left) and the glove used for measuring human grasps.

6.1 Grasp mapping

The human and robot hand cannot, in general, perform the same types of grasps. Consequently, a mapping between human hand and robot hand is necessary in order to control a robot hand with a human hand. For grasp mapping we have used an Artificial Neural Network(ANN), which can learn the mapping of the human hand space into the robot hand space. To obtain the measurements, we have used the Nest of Birds. It is a magnetic tracker that consists of an electronics unit, a transmitter, and four pose measuring sensors. Nest of Birds calculates the position and orientation of each sensor providing six degrees of freedom. The sensors are mounted on a glove, as illustrated in Figure 4: The centre sensor, mounted on the back side of the glove, serves as a reference sensor. It measures the position and orientation of the hand. The remaining three sensors are mounted on the thumb, index finger, and little finger, and provide fingertip position measurements. A two-layered ANN was used to learn the mapping from the human hand to the robot hand. The input values are the position coordinates of each of the fingers. There are three fingers and their positions are in 3D which gives us nine input values. Note here that the coordinates are provided relative to the reference sensor on the back of the hand. The output layer has as many neurons as the number of DoFs of the robot hand, and each neuron corresponds to a joint angle.

We have evaluated the grasp mapping on three different robot hands in order to study the mapping with respect to various levels of kinematics complexity. We first evaluated how good mapping we could obtain, using just seven training postures: open hand, each individual finger closed, and each half-closed. In the evaluation presented in [16], we have first concentrated only on posture mapping. While this mapping allows the user to form any posture with the robot hand, its accuracy may not be sufficient to grasp objects in an imitation scenario. In order to cope with this problem, we have also evaluated a system where, in the input layer of the ANN, we introduce an additional neuron representing the size of the object. At this point, we assume the object to consist of a basic shape with the form of a cylinder or a cube. The size indicates the object radius for a cylindrical object, or the side of a cubic object. Thus, we need separate networks for grasping cylinders and cubes.

The mapping system has also been used for automatic grasp generation for robotic hands, which is presented in detail in Deliverable D2.1. Here, experience and shape primitives are used in synergy and provide a basis not only for grasp generation but also for a grasp evaluation process when the exact pose of the object is not available. The entire grasp sequence is thoroughly evaluated in a simulated environment, from learning a grasp to actually reaching it, including dynamic simulation of the grasp execution and modelling of corrective movements. Details are given in reference [E].

6.2 Grasp Planning and Visual Object Localization

To enable starting of manipulation and exploration tasks on a humanoid robot with five-fingered hands, we presented in [F] an integrated approach for grasp planning and visual object recognition and localisation. The central idea of this system is the existence of a database with the models of all the objects present in the robot workspace. From this central fact we developed two necessary modules: a visual system able to recognize and localize objects in real-time using stereo vision, and an offline grasp analyzer that provides the most feasible grasps configuration for each object.

The results provided by these modules are stored and used by the control system of the humanoid to decide and execute the grasp of a particular object. The offline grasp analysis system determines the best grasp for the objects by employing a simulation system, together with CAD models of the objects and the hand. The results of this analysis are added to the object database using a description suited for the requirements of the grasp execution modules.

A stereo camera system is used for a real-time object localization using a combination of appearance-based and model-based methods. Given an object, a grasp of that object will be described by the following features (Figure 5): Grasp type, grasp starting point (GSP), approaching direction, and hand orientation.

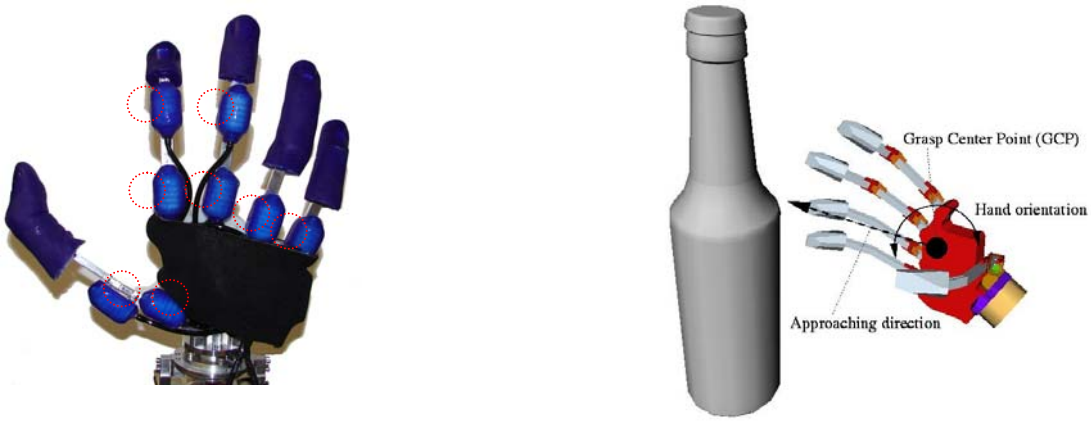
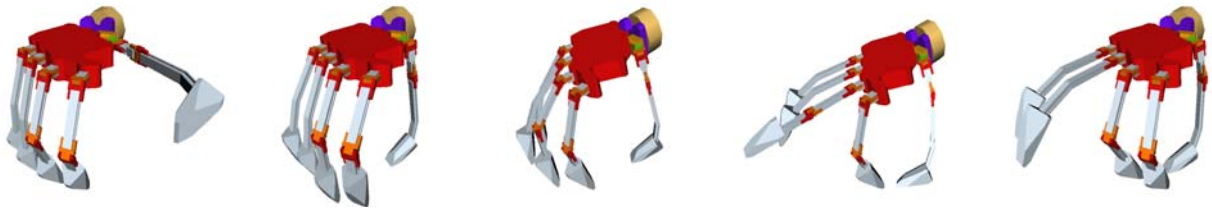


Figure 5: The used five-fingered anthropomorphic hand with eight independent joints (left) and a schematics with the grasp descriptors.

We perform an extensive analysis for each object that consists of testing a wide variety of hand preshapes and approach directions. This analysis is carried out on the simulation environment GraspIt! [17], where each tested grasp is evaluated according to a quality criterion. The resulting best grasps for each object are stored in order to be used during online execution on the robot. We considered five grasp types: precision pinch and tripod, power hook, cylindrical, and spherical. They are depicted in Figure 6.



(a) Hook (b) Cylindrical (c) Spherical (d) Pinch (e) Tripod

Figure 6: Hand preshapes for five grasp types.

We want to emphasize that our approach in [F] describes a first step towards a complete humanoid grasping system. At this stage, the use of object and hand models allows the fast development and testing of multiple

interactive manipulation and grasping skills. In the long-term, it is our purpose to develop grasping and manipulation strategies allowing to deal with unmodelled and unknown objects.

7. Links to other Workpackages

The work on human motion capture presented here relates to the ideas presented in WP3. As the part of WP8, we have mainly concentrated on vision based, markerless human tracking for the goal of action understanding and imitation. In WP3, we concentrate primarily on the definition of action representation as well as modelling and evaluation of different methodologies for action recognition and understanding through motor primitives.

A part of the work on grasping relates partially to WP2. However, in WP8 we are mostly interested in the issue of posture mapping from a human to robot hand, while WP2 concentrates on the modelling and evaluation of closed-loop grasp control and corrective movements.

Attached Papers

[A] P. Azad, A. Ude, T. Asfour, G. Cheng, and R. Dillmann. Stereo-based Markerless 3D Human Motion Capture using Multiple Cues. In *International Workshop on Vision Based Human-Robot Interaction*, Palermo, Italy, 2006.

[B] P. Azad, T. Asfour, and R. Dillmann. Toward an Unified Representation for Imitation of Human Motion on Humanoids. In *International Conference on Robotics and Automation (ICRA)*, Roma, Italy, 2007 (accepted to).

[C] T. Asfour, F. Gyarfas, P. Azad and R. Dillmann. Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006)*, Genoa, Italy, December 2006.

[D] M. Riley, A. Ude, C. G. Atkeson, and G. Cheng. Coaching: An Approach to Efficiently and Intuitively Create Humanoid Robot Behaviors. In *International Conference on Humanoid Robots (Humanoids 2006)*, Genova, Italy, 2006.

[E] J. Tegin, J. Wikander, S. Ekvall, D. Kragic, B. Illev. Experience based Learning and Control of Robotic Grasping In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006) - Workshop 'Towards Cognitive Humanoid Robots*, Genova, Italy, 2006.

[F] A. Morales, T. Asfour, P. Azad, S. Knoop and R. Dillmann. Integrated Grasp Planning and Visual Object Localization For a Humanoid Robot with Five-Fingered Hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

References

[1] 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.

[2] H. Sidenbladh, Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences. *Ph.D. dissertation*, Royal Institute of Technology, Stockholm, Sweden, 2001.

[3] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006)*, Genoa, Italy, December 2006

[4] T. Gumpp, P. Azad, K. Welke, E. Oztop, R. Dillmann, and G. Cheng. Unconstrained Real-time Markerless Hand Tracking for Humanoid Interaction. In *International Conference on Humanoid Robots (Humanoids 2006)*, Genova, Italy, 2006.

[5] Vicon Peak. http://www.vicon.com.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue, Learning by watching: Extracting reusable task knowledge from visual observation of human performance. In *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 799–822, 1994.

[7] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance. In *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[8] A. Billard and R. Siegwart, "Robot learning from demonstration. In *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 65–67, 2004.

[9] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. In *Robotics and autonomous systems*, vol. 47, pp. 69–77, 2004.

[10] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, 2005.

[11] C. G. Atkeson and S. Schaal, "Robot learning from demonstration. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML 1997)*, 1997, pp. 1040–1046.

[12] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. In *Philosophical Transactions of the Royal Society of London: Series B, Biological Science*, vol. 358, no. 1431, pp. 537–547, 2003.

[13] Y. Demiris and G. Hayes. Imitation as a dual-route process featuring predictive learning components: a biologically-plausible computational model. In *Imitation in animals and artifacts*, pp. 327–361, 2002.

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

[15] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005, pp. 2769–2774.

[16] J. M. C. Jorda. Intelligent task-level grasp mapping for robot control'. Master thesis TRITA-CSC-CV 2006:3, CVAP304, ISRN-KTH/CSC/CV--06/03--SE, Sept. 2006 http://www.nada.kth.se/cvap/cvaplop/lop-cvap.html

[17] A. Miller and P. Allen. Graspit!: A versatile simulator for robotic grasping. In *IEEE Robotics an Automation Magazine*, vol. 11, no. 4, pp. 110-122, Dec. 2004.

Image-based Markerless 3D Human Motion Capture using Multiple Cues

Pedram Azad[1], Ales Ude[2], Tamim Asfour[1], Gordon Cheng[3], and Ruediger Dillmann[1]

[1] Institute for Computer Science and Engineering, University of Karlsruhe, Germany
azad|asfour|dillmann@ira.uka.de
[2] Jozef Stefan Institute, Ljubljana, Slowenia
ales.ude@ijs.si
[3] Computational Neuroscience Laboratories, ATR, Kyoto, Japan
gordon@atr.jp

1 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, 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. Therefore, such systems can only provide data for highly supervised offline learning algorithms. In contrast, 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 – 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.

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 a frame rate of 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. 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 25 DOF, or 17 DOF if only modeling the upper body, leading in both cases 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 [5, 13]. The other approach is based on particle filtering and became popular by the work of Deutscher et al. [6]. Recently, we have started to adapt and extend this system for real-time application on a humanoid robot head [3], presenting the newest results in the following. 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 [10]. However, for this system, the goal is not to acquire smooth trajectories but to classify the activities of a human into categories, such as walking, waving, bowing, etc. Other approaches concentrate on deriving as much information as possible from monocular image sequences [15], 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 [14]. Our experience is that it is not possible to build a general 3d human motion capture system, 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.

We use the Bayesian framework *Particle Filtering* to compute the probability distribution of the current configuration, as described in detail in [3]. Particle filtering, also known as the *Condensation Algorithm* in the context of visual tracking, as introduced in [9], has proven to be an applicable and robust technique for contour tracking in general [4] [11] [12], and for human motion capture in particular, as shown in [6] [15].

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 [6, 7]. 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 PIII CPU [7].

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 higher convergence of the probability distribution. 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 with a high smoothness and accuracy for a purely vision based algorithm, without using markers or manual post-processing. The processing rate of our algorithm is 15 Hz on a 3 GHz CPU.

2 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 distribution 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 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. A detailed description of using particle filters for human motion capture can be found in [3].

2.1 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 likelihood that the configuration leading to the set of projected edges is the proper configuration i.e. matching the gradient image the most. The basic

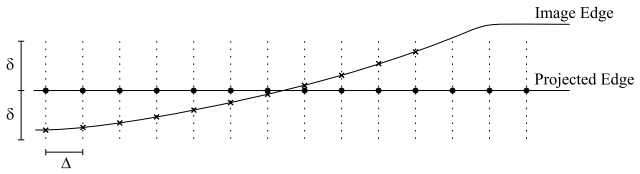


Fig. 1. Illustration of the search of edges

technique is to traverse the projected edges and search at fixed distances Δ for high-contrast features perpendicular to the projected edge within a fixed search distance δ (in each direction) i.e. finding edge pixels in the camera image, as illustrated in figure 1 [9]. For this purpose, usually the camera image is preprocessed to generate an edge image using a gradient based edge detector. The likelihood is calculated on the base of the Sum of Squared Differences (SSD). For convenience of notation, it is assumed that all edges are contained in one contiguous spline with $M = L/\Delta$ discretizations, where L denotes the sum of the length of all projected edges in the current image. The distance at which an edge feature has been found for the mth point is denoted as d_m and μ denotes a constant maximum error which is applied in case no edge feature could be found. The likelihood function can then be formulated as:

$$p(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma^2 M}\sum_{m=1}^{M} f(d_m, \mu)\right\} \tag{1}$$

where $f(\nu, \mu) = \min(\nu^2, \mu^2)$. Another approach 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 [6], rather than performing a search perpendicular to each pixel of the projected edge. By doing this, the computational effort is reduced significantly, even when picking the highest possible discretization of $\Delta = 1$ pixel. Furthermore, one does not have to make the non-trivial decision which gradient pixel to take for each pixel of the projected edge. Assuming that the spread gradient map has been remapped between 0 and 1, the modified likelihood function can be formulated as:

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

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

2.2 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. Both methods also assume 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 [16]. 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 [6]:

$$p_r(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_r^2 M_r}\sum_{m=1}^{M_r}(1-r_m)^2\right\} \tag{3}$$

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.

2.3 Fusion of Multiple Cues

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

$$p_{g,r}(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2}\left(\frac{\sum_{m=1}^{M_g}(1-g_m)^2}{\sigma_g^2 M_g} + \frac{\sum_{m=1}^{M_r}(1-r_m)^2}{\sigma_r^2 M_r}\right)\right\} \tag{4}$$

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 [6]. 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 3.

3 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 (3), we incorporate the result of foreground segmentation in a more efficient way, as will be introduced in Section 3.1. In Section 3.2 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 3.3. The final combined likelihood function is presented in Section 3.4.

3.1 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 3.2, 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 2.2, the only commonly used foreground segmentation technique is background subtraction, which we cannot 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 7. 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. We want to point out that the proposed generation of a filtered edge map does not depend on the segmentation technique.

3.2 Cue Studies and Distance Likelihood Function

In order to understand which are the benefits and drawbacks of each likelihood function and thus getting a feeling of what a likelihood function can do and what not, it is helpful to take a look at the corresponding probability distributions in a simple one-dimensional example. The experiment we use in simulation is tracking a square of fixed size in 2d, which can be simplified furthermore 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 the following, we want to compare 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:

$$p_d(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_d^2}|f(\mathbf{s}) - \mathbf{c}|^2\right\} \tag{5}$$

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 3.3 and 3.4. For the experiments, we used $N = 15$ particles and picked $\sigma_g = \sigma_r = \sqrt{5}$ and $\sigma_d = 0.1$. 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. As one can see in Figure 2, the gradient-based likelihood function p_g produces the narrowest distribution. The probability distributions produced by p_r and p_d are relatively similar; their narrowness can be adjusted by varying σ_r respectively σ_d. The effect of each distribution can be seen in Figure 3. 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. In the figures 4-6, the initial distance from the goal Δx_0 was varied. As expected, p_g leads to the fastest and smoothest convergence for $\Delta x_0 = 5$. $\Delta x_0 = 15$ is already close to the border of the convergence radius for p_g; the particle filter first tends to the wrong direction and then finally converges to the goal position. With $\Delta x_0 = 80$, it is by far impossible for p_g to find the global maximum, it converges to the (wrong) local maximum, matching the right edge of the model with the left

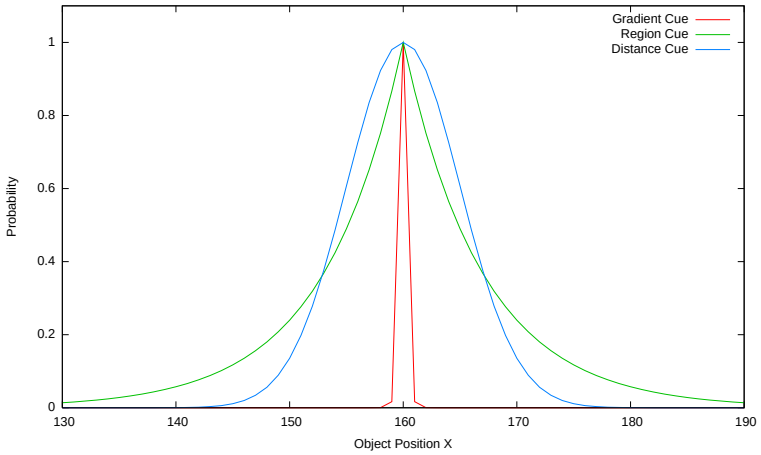


Fig. 2. Comparison of Probablity Distributions

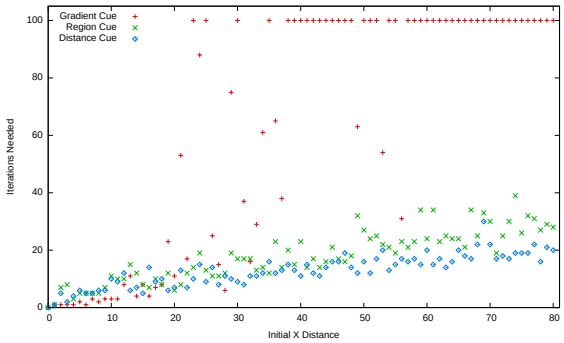


Fig. 3. Comparison of iteration numbers: an iteration number of 100 indicates that the goal was not found

edge of the square in the image. For $\Delta x_0 = 5$ and $\Delta x_0 = 15$, p_r and p_d behave quite similar. However, for $\Delta x_0 = 80$, p_d converges significantly faster, since it has the global view at any time. In contrast, p_r has to approach the goal slowly to reach the area, in which it can converge fast. 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. 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 17 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

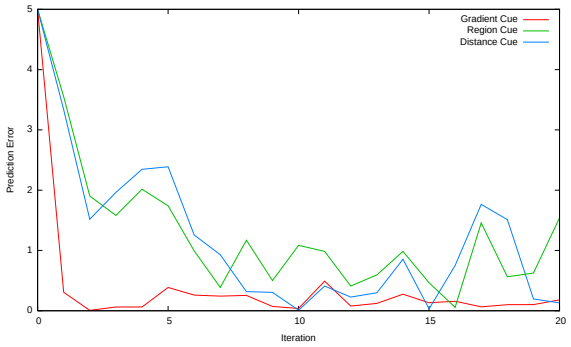


Fig. 4. Comparison of convergence for $\Delta x_0 = 5$

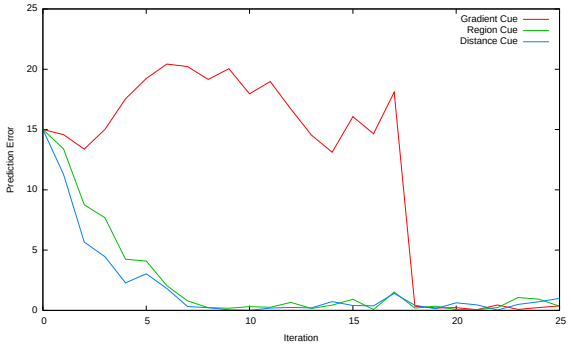


Fig. 5. Comparison of convergence for $\Delta x_0 = 15$

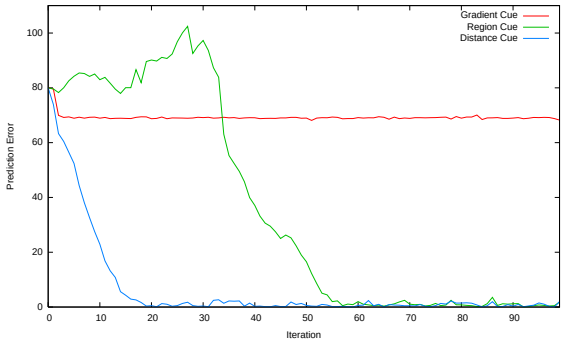


Fig. 6. Comparison of convergence for $\Delta x_0 = 80$

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 in our system in 3d, leading to a significant implicit reduction of the search space.

3.3 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 2.3. 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 preprocessing 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 4.

In principal, 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 circumstance 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 the in Figure 4 the prediction result of the likelihood function p_d is noisy within an area of 1-2 pixels in a very simple experiment, to a considerable error of the implicit stereo calculation in the real scenario. The accuracy of stereo triangulation decreases with the distance from the camera in a squared relationship. In order to observe the complete upper body of a human, the subject has to be located at a distance of at least 2-3 meters from the camera head. Thus, a potential error of two or more pixels in each camera image can lead to a significant error of the triangulation result. For this reason, in the proposed system, 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 distribution 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.

3.4 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 3.2 and 3.3. We have experienced that when leaving out the square in Equation (2), 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:

$$p'_g(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_g^2}(1-\frac{1}{M_g}\sum_{m=1}^{M_g} g_m)\right\} \tag{6}$$

For a system capable of 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. As our experimental results show, and as expected by the studies from Section 3.2, by doing this, a relatively small set of particles is sufficient for a successful system. The distance cue drags the peak of the distribution into a subspace in which the hands and the head are located at the true positions. Thus, search space is reduced implicitly, practically leaving the choice in this subspace to the cooperating gradient cue, based on the likelihood function p'_g. In order to formulate the distance cue, first the function $d_i(\mathbf{s}, \mathbf{c})$ is defined as:

$$d_i(\mathbf{s}, \mathbf{c}) := \begin{cases} |f_i(\mathbf{s}) - \mathbf{c}|^2 & : \quad \mathbf{c} \neq \mathbf{0} \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7}$$

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. Furthermore:

$$g(\mathbf{c}) := \begin{cases} 1 & : \quad \mathbf{c} \neq \mathbf{0} \\ 0 & : \quad \text{otherwise} \end{cases} \tag{8}$$

The likelihood function for the distance cue is then formulated as:

$$p'_d(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma_d^2}\frac{d_1(\mathbf{s}, \mathbf{c_1}) + d_2(\mathbf{s}, \mathbf{c_2}) + d_3(\mathbf{s}, \mathbf{c_3})}{g(\mathbf{c_1}) + g(\mathbf{c_2}) + g(\mathbf{c_3})}\right\} \tag{9}$$

where the vector $\mathbf{c_i}$ are computed on the base of the image observations $\mathbf{z}$ using skin color segmentation and stereo triangulation, as explained in Section 3.3. 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 distribution 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:

$$p(\mathbf{z}|\mathbf{s}) \propto \exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_d^2}\sum_{i=1}^{3}\frac{d_i(\mathbf{s}, \mathbf{c_i})}{g(\mathbf{c_i})} + \frac{1}{\sigma_g^2}(1-\frac{1}{M_g}\sum_{m=1}^{M_g} g_m)\right)\right\} \tag{10}$$

4 Image Processing Line

The image processing line is a pipeline, transforming the input image pair into a skin color map and a gradient map, which are then used by the likelihood function presented in Section 3.4. In Figure 7, the image processing line for one image is shown; in the system the pipeline is applied twice: once for the left and once for the right input image. After the input images are smoothed with a 3×3 Gaussian kernel, the HSI image is computed. The HSI image is then filtered twice, once for skin color segmentation and once for foreground segmentation by segmenting the shirt color. A simple 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 applying a 3×3 or 5×5 Gaussian kernel, as done in [6]. Currently, the hands

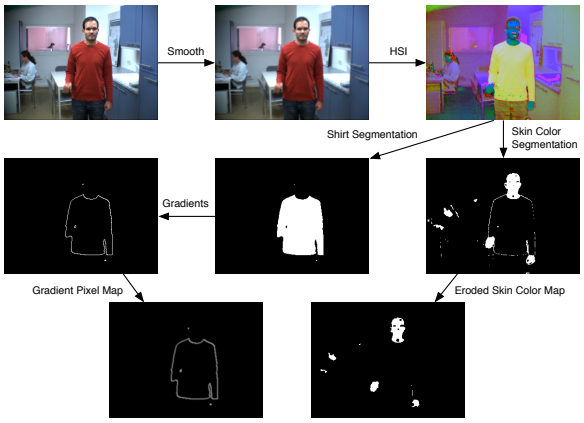


Fig. 7. Visualization of the image processing line

and the head are segmented using a fixed interval color model in HSI color space. The resulting color blobs are matched, 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. As will be discussed in Section 7, we are currently working on implementing a more sophisticated hand-/head-tracking system, which allows to deal with occlusions of skin colored regions.

5 Integrating Vision Toolkit

The complete system has been implemented using the *Integrating Vision Toolkit* (IVT) extensively [2]. With the IVT, the complete image processing

line presented in Section 4 could be implemented in less than 50 lines of code. The IVT provides a clean interface to capture devices of any kind, providing a convenient application for stereo camera calibration based on the OpenCV. For implementing Graphical User Interfaces, QT is integrated optionally, as well as the OpenCV library for image processing routines which are not yet available. The library is implemented in an easy-to-use software architecture, hiding all dependencies behind clean interfaces. The IVT fully supports the operating systems Linux, Mac OS and Windows. The project is available on Sourceforge; the link is included in the References.

6 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 1. 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, if moving more slowly, a processing rate of 15 Hz is sufficient. The relationship between the speed of the movements to be tracked and the frame rate at which the images are captured (and for real-time application processed) is briefly discussed in Section 7. In Figure

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

Table 1. Processing times with $N = 1000$ particles on a 3 GHz CPU

8, six screenshots are shown which show how the system automatically initializes itself. No configuration is told the system; it autonomously finds the only possible configuration matching the observations. Figure 9 shows four screenshots of the same video sequence, showing the performance of the human motion capture system tracking a punch with the left hand. The corresponding video and videos of other sequences can be downloaded from `http://i61www.ira.uka.de/users/azad/videos`.

Fig. 8. Screenshots showing automatic initialization

Fig. 9. Screenshots showing tracking performance

7 Discussion

We have presented an image-based markerless human motion capture system for real-time application. The system is capable of computing very smooth and accurate trajectories in configuration space for such a system. We presented our strategy of multi-cue fusion within the particle filter, and showed the results of studies examining the properties of the cues commonly used and a further distance cue. We showed that by using this distance cue combined with stereo vision, which has not yet been used in markerless human motion capture, we could reduce the size of the search space implicitly. This reduction of search space allows us to capture human motion with a particle filter using as few as 1000 particles with a processing rate of 15 Hz on a 3 GHz CPU. We plan to investigate and implement several improvements of the system:

- Currently, the subsystem for detection of the hands and the head in the images is not powerful enough to deal with occlusions of skin-colored regions in the image. To overcome this problem, we are currently working on implementing a more sophisticated hand and head tracking system, as presented by Argyros et al. [1]. By doing this, we expect the system to be

able to robustly track long and complicated sequences, since it will not be required to try to avoid occlusions between the hands and the hand.

- For any kind of tracking, the effective size of the search space increases exponentially with the potential speed of the movements respectively decreases exponentially with the frame rate at which images are captured. For this reason, the human motion capture systems with the most convincing results use a framerate of 60 Hz or higher, as done by [6]. Commercial marker-based tracking systems use a frame rate of 100 Hz up to 400 Hz and higher, to acquire smooth trajectories. For this reason, we want to perform further tests with the new Dragonfly2 camera, which is capable of providing the same image data as the Dragonfly camera, but at a frame rate of 60 Hz instead of 30 Hz.
- In the theory of particle filters, there exist several methods to decrease the effectively needed number of particles by modification of the standard filtering algorithm. For this purpose, we want to investigate the work on Partitioned Sampling [12] and Annealed Particle Filtering [6].
- We plan to extend the human model by incorporating the legs and feet into the human model. Especially for this purpose, we want to use the benefits of an active head, since with a static head it is hardly possible to have the complete human in the field of vision of the robot at one time step.

To our best knowledge, the proposed system is the first purely image-based markerless human motion capture system designed for a robot head which can track human movements with such accuracy and smoothness, and being suitable for real-time application at the same time. The system does not assume a static camera in any way; future work will also concentrate on running experiments using this benefit of being able to capture human motion while tracking the subject actively.

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).

References

1. A. A. Argyros and M. I.A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *European Conference on Computer Vision (ECCV)*, volume 3, pages 368–379, Prague, Czech Republic, 2004.
2. P. Azad. Integrating Vision Toolkit. `http://ivt.sourceforge.net`.

3. 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.
4. A. Blake and M. Isard. *Active Contours*. Springer, 1998.
5. F. Caillette and T. Howard. Real-time markerless human body tracking with multi-view 3-d voxel reconstruction. In *British Machine Vision Conference*, volume 2, pages 597–606, Kingston, UK, 2004.
6. J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2133, Hilton Head, USA, 2000.
7. 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)*, pages 669–676, Kauai, USA, 2001.
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)*, pages pp. 73–80, San Francisco, USA, 1996.
9. M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
10. 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.
11. J. MacCormick. *Probabilistic models and stochastic algorithms for visual tracking*. PhD thesis, University of Oxford, UK, 2000.
12. J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *European Conference Computer Vision (ECCV)*, pages 3–19, Dublin, Ireland, 2000.
13. I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003.
14. K. Rohr. Human movement analysis based on explicit motion models. *Motion-Based Recognition*, pages 171–198, 1997.
15. H. Sidenbladh. *Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.
16. K. Wong and M. Spetsakis. Motion segmentation and tracking. In *International Conference on Vision Interface*, pages 80–87, Calgary, Canada, 2002.

Toward an Unified Representation for Imitation of Human Motion on Humanoids

Pedram Azad, Tamim Asfour and Rüdiger Dillmann
Institute for Computer Science and Engineering
University of Karlsruhe,
Haid-und-Neu-Strasse 7, 76131 Karlsruhe, Germany
Email: azad|asfour|dillmann@ira.uka.de

***Abstract*— In this paper, we present a framework for perception, visualization, reproduction and recognition of human motion. On the perception side, various human motion capture systems exist, all of them having in common to calculate a sequence of configuration vectors for the human model in the core of the system. These human models may be 2D or 3D kinematic models, or on a lower level, 2D or 3D positions of markers. However, for appropriate visualization in terms of a 3D animation, and for reproduction on an actual robot, the acquired motion must be mapped to the target 3D kinematic model. On the understanding side, various action and activity recognition systems exist, which assume input of different kinds. However, given human motion capture data in terms of a high-dimensional 3D kinematic model, it is possible to transform the configurations into the appropriate representation which is specific to the recognition module. We will propose a complete architecture, allowing the replacement of any perception, visualization, reproduction module, or target platform. In the core of our architecture, we define a reference 3D kinematic model, which we intend to become a common standard in the robotics community, to allow sharing different software modules and having common benchmarks.**

I. Introduction

In the recent past, research on visual perception and understanding of human motion has become of high interest. On the one hand, a large number of various approaches on the perception side exists, resulting in different human motion capture systems, producing output in terms of different models and stored in different formats. On the other hand, many action recognition and activity recognition systems exist, expecting input data specific to their own internal representation. Furthermore, any target platform for the reproduction of human motion, namely 3D models for animation and simulation purposes, and humanoid robots, expects human motion capture data in terms of its own 3D kinematic model. Because of this fact, currently it is not possible to exchange single modules in an overall infrastructure for a humanoid robot, including perception, visualization, reproduction, and recognition of human motion. Furthermore, having common benchmarks is only feasible when a common representation for human motion is shared.

Methods for modeling, generating of human motion and its reproduction on humanoid robots have been proposed in robotics and computer graphics ([1], [2], [3], [4]). These methods involve capturing full body motion of a human performer using human motion capture systems and the transformation of the motion to the kinematics of a humanoid robot or human figures. To answer the question of how to generate and represent motor primitives in imitation learning architectures, several approaches have been proposed. In most related researches, continuous Hidden Markov Models (HMM) are used, where the Viterbi algorithm plays an important role to generate a close motion to the observation. In [5], a stochastic model has been proposed that abstracts the whole body motion as symbols and integrates motion recognition, generation and symbolization of motion patterns. Nakamura presents in [6] a stochastic mimesis model for the imitation of new observed motions without learning the motion and for the online acquisition of motion patterns.

Human motion capture systems can be generally divided into marker-based and markerless systems. Marker-based systems are widely used in the film and animation industry, and in the research field of human motion analysis. One of the most popular marker-based motion capture systems is the VICON system [7], which can track a set of reflective markers that are attached to the person to be tracked. Such systems can acquire a very high accuracy and a high temporal resolution; their output is a set of 3D positions of markers over time in first place. This output data has then to be post-processed manually, and if necessary translated into the configuration space of a given 3D human model by determining rotations on the base of adjacent markers, as done in [8].

Markerless systems are often used in the context of visual perception for humanoid robot systems. Recently, many approaches have been proposed, differing in the number, type, and arrangement of the sensors incorporated, real-time applicability, the ability to perceive 2D or real 3D motion, and the smoothness of the output trajectories. 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 [9], [10]. The other approach is based on particle filtering and became popular by the work of Deutscher et al. [11] for multi-camera systems, and later by [12] for the specific case of monocular image sequences. Recently, we have started to adapt and extend this system for real-time application on a humanoid robot head ([13], [14]). 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 [15].

The output data computed by the mentioned systems is intended to be used for different applications, namely action and activity recognition, and the reproduction of movements on a humanoid robot system. For visualization and evaluation purposes, the data is usually also applied to a 3D human model. The problem is that most human motion capture systems are intended for the mentioned applications, but in practice the transfer rarely happens. The reason is that each system is based on its own representation and therefore the output data is always given in terms of this specific human model with its own data format, which is in general not compatible with systems for action or activity recognition, or the kinematic model of an actual robot system. Since one research group cannot deal with all the mentioned issues, it is crucial to overcome this deficiency to allow for compatibility of any module developed for any of the mentioned purposes. In the following, we propose a framework, defining a reference 3D kinematic human model in its core, and presenting the architecture, with which all modules are connected together. As a practical example, we show how we built in a markerless human motion capture system developed at our institute and the humanoid robot ARMAR III for reproduction of movements.

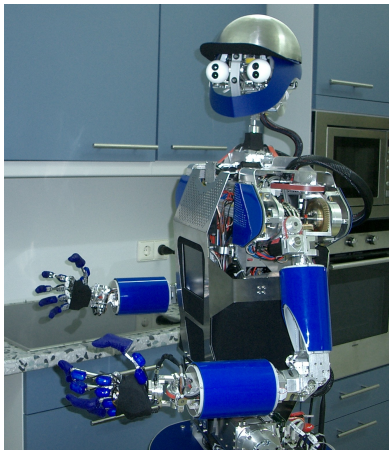


Fig. 1. ARMAR III with sensor head

II. Human Motion Capture

In this section, we want to give a short outline of marker-based and markerless human motion capture systems, in particular concentrating on the data input and output of the systems. In Section III, we show how their output can be mapped to one canonical representation.

A. Markerless Human Motion Capture

As mentioned in Section I, various approaches for markerless human motion capture exist. Here, we want to introduce a system intended for real-time application on an active head of a humanoid robot system, which has been developed at our institute ([13], [14]). The input of the system is a stereo color image sequence, captured with two calibrated Dragonfly cameras built-in into the head of the humanoid robot ARMAR III, which can be seen in Figure 1. The input images are preprocessed, generating output for the gradient cue, the distance cue, and an optional region cue, as described in [14]. The image processing pipeline for this purpose is illustrated in Figure 2.

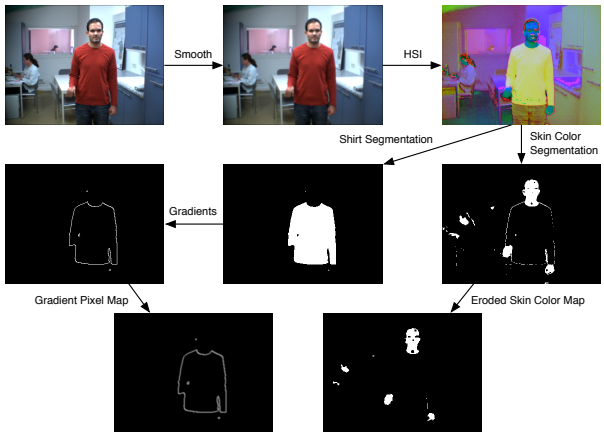


Fig. 2. Visualization of the image processing line

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:

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

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:

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

where $n := \dim(\mathbf{s})$ is the number of DoF of the human model. The transformation $f_i : R^n \rightarrow R^3$ 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 g_m with $m \in \{1,\ldots,M_g\}$ 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.

A detailed description is given in [14]. 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 $\bar{\mathbf{s}}$ over all particles. The format of the output configuration $\bar{\mathbf{s}}$ is:

$$\bar{\mathbf{s}} = (\mathbf{t}_{BT}\ \boldsymbol{\theta}_{BT}\ \boldsymbol{\theta}_{LS}\ \theta_{LE}\ \boldsymbol{\theta}_{RS}\ \theta_{RE})^T,$$

where BT denotes the base transformation, LS and RS the shoulder joints for the left and right arm, and LE and RE the elbow joints. $\mathbf{t}_{BT} \in \mathbb{R}^{1\times3}$ is the base translation, $\boldsymbol{\theta}_{BT}, \boldsymbol{\theta}_{LS}, \boldsymbol{\theta}_{RS} \in \mathbb{R}^{1\times3}$ are rotations given in the Euler convention $R_{X'Z'Y'}(\alpha,\beta,\gamma)$, and $\theta_{RE}, \theta_{RE} \in \mathbb{R}$ are scalar values for the elbow angle α in the Euler convention $R_{X'Z'Y'}(\alpha,0,0)$. The Euler angle conventions can be found in the Appendix B of [16].

B. Marker-based Human Motion Capture

Marker-based human motion capture systems are commercially available and often used in the film and animation industry. The probably most popular one is the VICON system, which consists of a set of infrared cameras, with each having a diode array attached to it. The system offers a convenient calibration routine, which makes it possible to determine 3D positions for the reflective markers that come with the system. Since occlusions can occur, it is necessary to postprocess the output, which can be relatively time-consuming, depending on the number of markers used. In Figure 3, a typical marker setup for the acquisition of human upper body motion is illustrated. Having postprocessed the data, the human motion capture data, which consists of a temporal sequence of 3D marker position sets, can be used for various purposes.

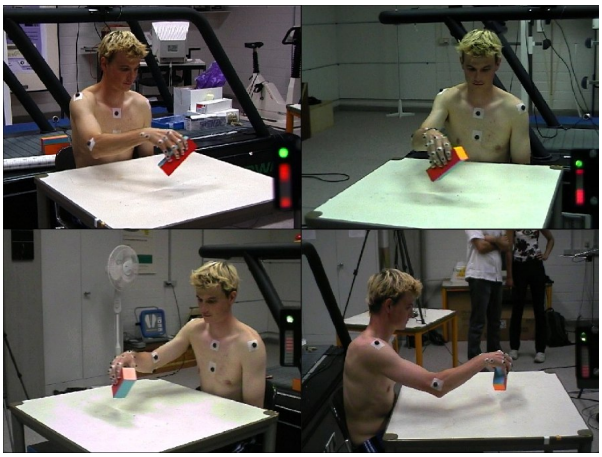


Fig. 3. Illustration of a marker-based human motion capture setup from [8]

For action recognition, it is possible either to use the plain 3D marker positions as input directly, or use trajectories in joint angle space. For the second approach, the trajectories of 3D marker position have to be transformed to joint angle trajectories by deriving rotations from the positions of adjacent markers, as described in [8]. Depending on the target kinematic model of the transformation, the joint angle trajectories can be used for visualization with an articulated 3D human model, or reproduced on an actual humanoid robot system. However, the problem is that usually this approach is a dead-end, since the target kinematic model is determined in advance, and therefore, the final data can be used only for the one desired purpose. Only with a lot of effort, human motion capture data can be shared within the robotics community, since an agreement on a common representation has been missing so far.

III. Master Motor Map

To overcome the deficiencies mentioned above, we propose a reference kinematic model, which we will call the *Master Motor Map* (*MMM*) in the following. The strategy is to define the maximum number of DoF that might be used by any visualization, recognition, or reproduction module, but not more than that. The H-Anim 1.1 specification [17] defines a joint for each vertebra of the spine, which is not suitable for the robotic applications mentioned. Moreover, the H-Anim 1.1 specification does only define relative joint positions in terms of a graph, but not the actual kinematic model including the joint angle conventions for each joint, which is crucial for any robotic application. Therefore, we have defined a subset of the H-Anim 1.1 specification and have specified the joint angle conventions for each joint.

The joints which build the subset of H-Anim 1.1 are *skullbase*, *vc7*, *vt6*, *pelvis*, *HumanoidRoot*, *l_hip*/*r_hip*, *l_knee*/*r_knee*, *l_ankle*/*r_ankle*, *l_sternoclavicular*/*r_sternoclavicular*, *l_shoulder*/*r_shoulder*, *l_elbow*/*r_elbow*, and *l_wrist*/*r_wrist*. The numbers of DoF and the Euler angle conventions are listed in Table I. The kinematic model for the MMM is illustrated in Figure 4.

	DoF	Euler angles
skullbase	3	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
vc7	3	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
vt6	3	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
pelvis	3	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
HumanoidRoot	3	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
l_hip / r_hip	$3+3$	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
l_knee / r_knee	$1+1$	$R_{X'Z'Y'}(\alpha,0,0)$
l_ankle / r_ankle	$3+3$	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
l_sternoclavicular / r_sternoclavicular	$3+3$	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
l_shoulder / r_shoulder	$3+3$	$R_{X'Z'Y'}(\alpha,\beta,\gamma)$
l_elbow / r_elbow	$2+2$	$R_{X'Z'Y'}(\alpha,\beta,0)$
l_wrist / r_wrist	$2+2$	$R_{X'Z'Y'}(\alpha,0,\gamma)$
Total	52	

TABLE I
Number of Degrees of Freedom and Euler angle conventions for the joints of the MMM

The file format is specified as follows. The 52-dimensional configuration vectors are written sequentially to a text file, where each component is a floating point number formatted as readable text. All components are separated by whitespace. After one configuration, an additional floating point value specifies the associated timestamp in milliseconds. Since one configuration contains a fixed number of 53 numbers (including the timestamp), it is not necessary to introduce an explicit end of one configuration. For readability, it is recommended to put a line break after each timestamp instead of a space. The order of the 52 floating point numbers for the configuration vector is:

$$(\mathbf{t}_{RT}\ \boldsymbol{\theta}_{RT}\ \boldsymbol{\theta}_{SB}\ \boldsymbol{\theta}_{VC7}\ \boldsymbol{\theta}_{P}\ \boldsymbol{\theta}_{VT6}\ \boldsymbol{\theta}_{LSC}\ \boldsymbol{\theta}_{LS}\ \boldsymbol{\theta}_{LE}\ \boldsymbol{\theta}_{LW}\ \boldsymbol{\theta}_{RSC}\ \boldsymbol{\theta}_{RS}\ \boldsymbol{\theta}_{RE}\ \boldsymbol{\theta}_{RW}\ \boldsymbol{\theta}_{LH}\ \theta_{LK}\ \boldsymbol{\theta}_{LA}\ \boldsymbol{\theta}_{RH}\ \theta_{RK}\ \boldsymbol{\theta}_{RA})^T$$

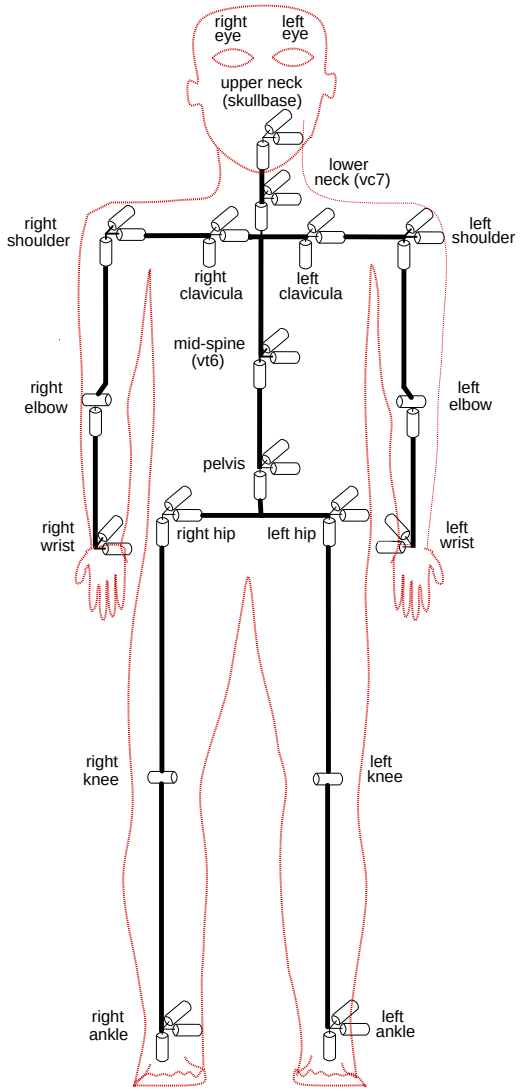


Fig. 4. Illustration of the MMM kinematic model

where RT denotes the root transformation, SB the skull base joint, P the pelvis joint, LSC/RSC the sternoclavicular joints, LS/RS the shoulder joints, LE/RE the elbow joints, LH/RH the hip joints, LK/RK the knee joints, and LA/RA the ankle joints. The length of each vector is given by the number of DoF, given in Table I.

IV. Framework and Converter Modules

In the following, we propose the framework which connects all mentioned modules, namely responsible for data acquisition, visualization, reproduction, and recognition. In the core of the framework is the MMM, as specified in Section III. All perceptive modules have an additionally implemented converter module, which transforms the output data to the MMM. Modules for visualization, reproduction, and recognition, which need motion capture data as input, implement an additional converter module, which transforms the data provided in the MMM format to the required input data format. This framework is illustrated in Figure 5.

The converter modules implement the transformation from one human motion representation to the MMM, or vice versa. In the case of marker-based human motion capture systems, this transformation is computed on the base of adjacent markers, as described in [8]. For all other modules, the converter module has to perform a transformation between two different kinematic models. There are five common basic types of adaptations which can occur in such a transformation:

1) Changing the order of values (all modules).
2) Setting zeroes for joint angles which are not captured (perception modules).
3) Ignoring joint angle values which can or are not to be used (reproduction and recognition modules).
4) Transformations between two different Euler angle conventions for a ball joint (all modules).
5) Adaptations that include more than one joint, if the target module does not offer the corresponding degrees of freedom (reproduction and recognition modules).

We will show three example converter modules, covering all five mentioned cases. One converter module is for mapping the output of our markerless human motion capture system to the MMM, the second for mapping the MMM to the kinematic model for the humanoid robot ARMAR, and the third for transforming 6D task space trajectories to the MMM. Cases 1, 2, and 3 are trivial. Case 4 can be solved by carefully calculating the conversion between two Euler angle conventions, as will be shown. Case 5 can not be generalized; we will show an example for the humanoid robot ARMAR. In the following, the notation $\mathbf{0}_i$ denotes the transposed zero vector of $\mathbb{R}^i$ i.e. $\mathbf{0}_i \in \{0\}^{1\times i}$.

A. Conversion Example 1

Here, we show how the output of our markerless human motion capture system is mapped to the MMM. The conversion covers the cases 1 and 2; case 4 does not occur, since the Euler angle conventions for the shoulder joints are both $R_{X'Z'Y'}(\alpha,\beta,\gamma)$, as is the base rotation for both models. Cases 3 and 5 are not of interest for perception modules. The transformation is formulated as follows:

$$f_1 : \mathbb{R}^{14} \rightarrow \mathbb{R}^{52}$$
$$(\mathbf{t}_{BT}\ \boldsymbol{\theta}_{BT}\ \boldsymbol{\theta}_{LS}\ \theta_{LE}\ \boldsymbol{\theta}_{RS}\ \theta_{RE})^T \rightarrow$$
$$(\mathbf{t}_{BT}\ \boldsymbol{\theta}_{BT}\ \mathbf{0}_{15}\ \boldsymbol{\theta}_{LS}\ \theta_{LE}\ \mathbf{0}_6\ \boldsymbol{\theta}_{RS}\ \theta_{RE}\ \mathbf{0}_{17})^T$$

B. Conversion Example 2

Here, we show how the MMM is mapped to the kinematic model of the humanoid robot ARMAR. The conversion covers the cases 1, 3, 4, and 5. Case 2 is only of interest for perception modules. The first problem is that ARMAR does not have the sternoclavicular joint, which is an example for case 5. One solution is to calculate the effective rotation matrix for the combination of the sternoclavicular and the shoulder joint. The effective rotation for the sternoclavicular and the shoulder joints is then formulated as:

$$\begin{aligned} R_{LS'} &= R_{X'Z'Y'}(\boldsymbol{\theta}_{LSC}) \cdot R_{X'Z'Y'}(\boldsymbol{\theta}_{LS}) \\ R_{RS'} &= R_{X'Z'Y'}(\boldsymbol{\theta}_{RSC}) \cdot R_{X'Z'Y'}(\boldsymbol{\theta}_{RS}), \end{aligned}$$

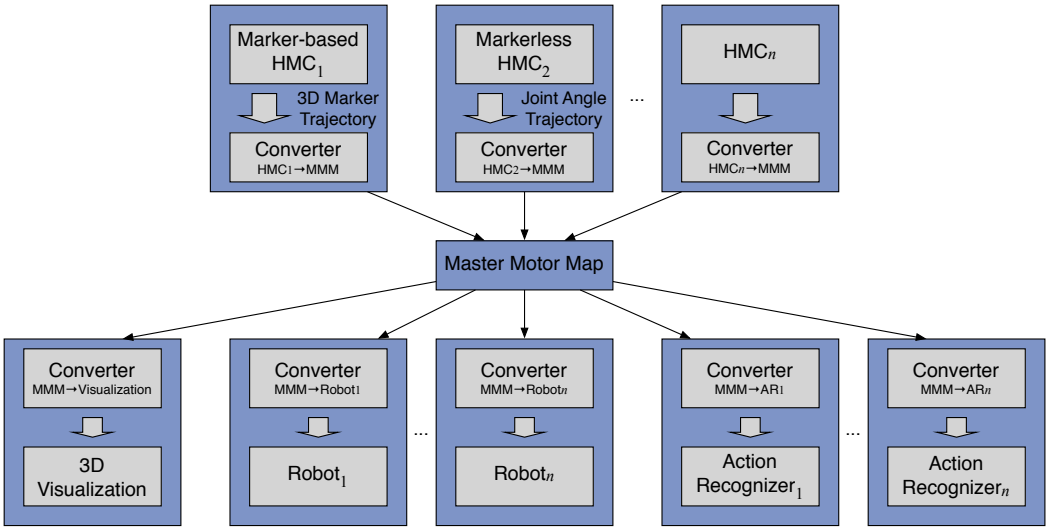


Fig. 5. Illustration of the proposed framework

where the notation $R(\boldsymbol{\theta})$ means $R(\alpha,\beta,\gamma)$ with $(\alpha\ \beta\ \gamma) = \boldsymbol{\theta}$. In the following, c denotes the side, which is necessary because ARMAR III has different coordinate systems for the two sides, in contrast to the MMM:

$$c := \begin{cases} 1 & : \text{ left} \\ -1 & : \text{ right} \end{cases}$$

Then, the kinematics for the shoulder joint of ARMAR III are defined as:

$$\begin{aligned} & R_{S,ARMAR}(c,\alpha,\beta,\gamma) \\ = \ & R_y(c\tfrac{\pi}{6}) \cdot R_x(-\tfrac{\pi}{2}) \cdot R_z(-c\alpha) \cdot R_x(\beta) \cdot R_y(c\gamma) \\ = \ & R_y(c\tfrac{\pi}{6}) \cdot R_x(-\tfrac{\pi}{2}) \cdot R_{Z'X'Y'}(-c\alpha,\beta,c\gamma) \end{aligned}$$

The problem of calculating the transformed rotation can be formulated as finding the angles α,β,γ so that:

$$\begin{aligned} R_{LS'} &= R_{S,ARMAR}(1,\alpha,\beta,\gamma) \\ R_{RS'} &= R_{S,ARMAR}(-1,\alpha,\beta,\gamma) \end{aligned}$$

and furthermore:

$$R_L := R_x(-\frac{\pi}{2})^T \cdot R_y(\frac{\pi}{6})^T \cdot R_{LS'} = R_{Z'X'Y'}(-\alpha,\beta,\gamma)$$
$$R_R := R_x(-\frac{\pi}{2})^T \cdot R_y(-\frac{\pi}{6})^T \cdot R_{RS'} = R_{Z'X'Y'}(\alpha,\beta,-\gamma)$$

In the following, we determine the solution for the left shoulder; the solution for the right shoulder can be determined analogously. First, we need the rotation matrix $R_{Z'X'Y'}(-\alpha,\beta,\gamma)$:

$$\begin{aligned} & R_{Z'X'Y'}(-\alpha,\beta,\gamma) \\ = & \begin{pmatrix} s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha c\beta & -s\alpha s\beta c\gamma + c\alpha s\gamma \\ c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha c\beta & -c\alpha s\beta c\gamma - s\alpha s\gamma \\ -c\beta s\gamma & s\beta & c\beta c\gamma \end{pmatrix} \\ = & \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} = R_L \end{aligned}$$

The two possible solutions can then be calculated by:

$$\alpha = \text{atan2}(\pm r_2, \pm r_5) \quad (1)$$
$$\beta = \text{atan2}(r_8, \pm\sqrt{r_2^2 + r_5^2}) \quad (2)$$
$$\gamma = \text{atan2}(\mp r_7, \pm r_9) \quad (3)$$

These can be disambiguated by taking into account maximum joint angle constraints. To formulate the complete conversion, we define for the left shoulder:

$$g_l : \mathbb{R}^6 \rightarrow \mathbb{R}^3 : \quad (\boldsymbol{\theta}_{LSC}\ \boldsymbol{\theta}_{LS}) \ \rightarrow (\alpha\ \beta\ \gamma)^T,$$

as given in the equations (1)-(3); g_r is defined analogously. The final conversion is then given by:

$$f_2 : \mathbb{R}^{52} \rightarrow \mathbb{R}^{14}$$
$$(\mathbf{t}_{RT}\ \boldsymbol{\theta}_{RT}\ \boldsymbol{\theta}_{SB}\ \boldsymbol{\theta}_{VC7}\ \boldsymbol{\theta}_P\ \boldsymbol{\theta}_{VT6}\ \boldsymbol{\theta}_{LSC}\ \boldsymbol{\theta}_{LS}\ \boldsymbol{\theta}_{LE}\ \boldsymbol{\theta}_{LW}$$
$$\boldsymbol{\theta}_{RSC}\ \boldsymbol{\theta}_{RS}\ \boldsymbol{\theta}_{RE}\ \boldsymbol{\theta}_{RW}\ \boldsymbol{\theta}_{LH}\ \theta_{LK}\ \boldsymbol{\theta}_{LA}\ \boldsymbol{\theta}_{RH}\ \theta_{RK}\ \boldsymbol{\theta}_{RA})^T \rightarrow$$
$$(g_l \begin{pmatrix} \boldsymbol{\theta}_{LSC}^T \\ \boldsymbol{\theta}_{LS}^T \end{pmatrix}^T \boldsymbol{\theta}_{LE}\ \boldsymbol{\theta}_{LW}\ g_r \begin{pmatrix} \boldsymbol{\theta}_{RSC}^T \\ \boldsymbol{\theta}_{RS}^T \end{pmatrix}^T \boldsymbol{\theta}_{RE}\ \boldsymbol{\theta}_{RW})^T$$

C. Conversion Example 3

This example is important for the programming and execution of manipulation tasks, which are specified in terms of object trajectories. Using a magnetic tracking system (Fasttrak, www.polhemus.com), both the position and the orientation of the hand $(x\ y\ z\ \alpha\ \beta\ \gamma)^T$ is tracked in Cartesian space. The mapping to the MMM is provided by a closed-form inverse kinematics algorithm, which computes the transformation for an arm:

$$f_3 : \mathbb{R}^6 \rightarrow \mathbb{R}^7 :$$
$$(x\ y\ z\ \alpha\ \beta\ \gamma)^T \rightarrow\ (\boldsymbol{\theta}_{LS}\ \boldsymbol{\theta}_{LE}\ \boldsymbol{\theta}_{LW})^T$$

In solving the inverse kinematics problem of the arm, the arm redundancy is used for the generation of human-like arm postures. The arm joint angles are reconstructed using a sensorimotor transformation model, which was found in physiological observation of human arm movements [18]. The model maps the Cartesian wrist position to a natural arm posture using a set of representation parameters, which are the upperarm elevation, the forearm elevation, the upperarm yaw, and the forearm yaw. Once these parameters are obtained, the shoulder and elbow joint angles are calculated to match the hand position whereas the forearm and wrist joint angles are calculated to match the hand orientation (see [19]).

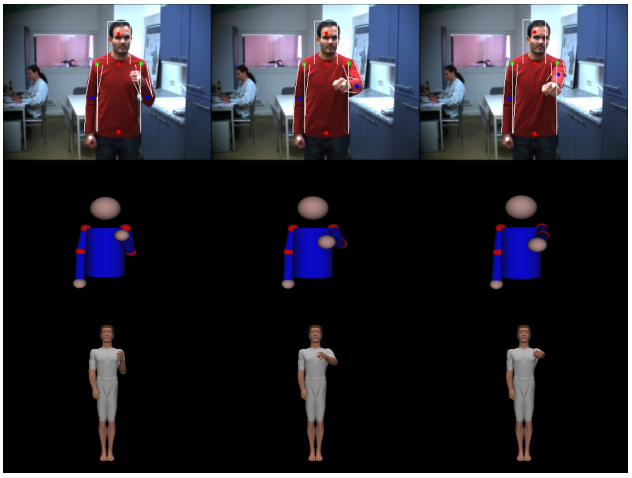


Fig. 6. Example frames. Top: Projected result from the HMC system. Middle: 3D Visualization with the HMC model. Bottom: 3D Visualization of the MMM representation.

V. Conclusion

We have presented a framework for perception, visualization, reproduction, and recognition of 3D human motion. In the core of our framework, we have defined a reference kinematic model – the Master Motor Map. We have showed that our approach performs well in practice by having built in a markerless human motion capture system, a 3D visualization of the MMM, and a module for converting trajectories given in the MMM to the kinematic model of the humanoid robot ARMAR III. Although the proposed framework is intended primarily to query the kinematics of 3D human motion, it can be also augmented with dynamic parameters of the body parts. Currently, we are working on reproducing movements on ARMAR III not only in simulation but on the real robot. Herefore, it is crucial to incorporate a self-collision detection module to avoid configurations which are outside the robot's working space. Furthermore, we are building up a database of movements in the MMM format, consisting of both markerless and marker-based human motion capture data, which will serve as a solid data source for applications related to imitation learning within the EU project PACO-PLUS, and hopefully also within in the entire robotics community.

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).

References

[1] A. Safonova, N. S. Pollard, and J. K. Hodgins, "Optimizing human motion for the control of a humanoid robot." in *Symposium on Adaptive Motion of Animals and Machines (AMAM03)*, Kyoto, Japan, March 2003.

[2] N. S. Pollard and J. K. Hodgins, "Generalizing demonstrated manipulation tasks." in *Workshop on the Algorithmic Foundations of Robotics (WAFR02)*, Nice, France, December 2002.

[3] A. Ude, C. G. Atkeson, and M. Riley, "Programming Full-Body Movements for Humanoid Robots by Observation," *Robotics and Autonomous Systems*, vol. 47, pp. 93–108, 2004.

[4] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Generation of humanoid robot motions with physical constraints using hierarchical b-spline," in *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, 2005, pp. 1869 – 1874.

[5] T. Inamura, Y. Nakamura, and I.Toshima, "Embodied symbol emergence based on mimesis theory," *International Journal of Robotics Research*, vol. 23, no. 4, pp. 363–377, 2004.

[6] L. Dongheui and Y. Nakamura, "Stochastic model of imitating a new observed motion based on the acquired motion primitives," in *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Beijing, China, 2006, pp. 4994 – 5000.

[7] "Vicon Peak," http://www.vicon.com.

[8] T. Beth, I. Boesnach, M. Haimerl, J. Moldenhauer, K. Bös, and V. Wank, "Characteristics in Human Motion – From Acquisition to Analysis," in *International Conference on Humanoid Robots (Humanoids)*, Karlsruhe/München, Germany, 2003.

[9] 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.

[10] 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.

[11] 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.

[12] H. Sidenbladh, "Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences," Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 2001.

[13] 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.

[14] P. Azad, A. Ude, T. Asfour, and R. Dillmann, "A Full Body Human Motion Capture System using Particle Filtering and On-The-Fly Edge Detection," in *International Conference on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

[15] 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.

[16] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[17] H. A. W. Group, "H-anim 1.1 specification," http://h-anim.org.

[18] J. F. Soechting and M. Flanders, "Errors in Pointing are Due to Approximations in Targets in Sensorimotor Transformations," *Journal of Neurophysiology*, vol. 62, no. 2, pp. 595–608, 1989.

[19] T. Asfour and R. Dillmann, "Human-like Motion of a Humanoid Robot Arm Based on Closed-Form Solution of the Inverse Kinematics Problem." in *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA, 27-31 Octorber, 2003.

Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots

Tamim Asfour, Florian Gyarfas, Pedram Azad and Rüdiger Dillmann
University of Karlsruhe
Institute for Computer Science and Engineering (CSE/IAIM)
P.O. Box 6980, D-76128 Karlsruhe, Germany
Email: {asfour,azad,dillmann}@ira.uka.de, florian.gyarfas@alumni.uni-karlsruhe.de

***Abstract*— In this paper, we deal with imitation learning of arm movements in humanoid robots. Hidden Markov Models (HMM) are used to generalize movements demonstrated to a robot multiple times. They are trained with the characteristic features (key points) of each demonstration. Using the same HMM, key points that are common to all demonstrations are identified; only those are considered when reproducing a movement. We also show how HMM can be used to detect temporal dependencies between both arms in dual-arm tasks. We created a model of the human upper body to simulate the reproduction of dual-arm movements and generate natural-looking joint configurations from tracked hand paths. Results are presented and discussed.**

I. Introduction

Humanoid robots are expected to exist and work together with human beings in everyday environments some day. In doing so they need to be able to interact and cooperate with humans. Interaction is facilitated if the robot behaves in a human-like way which implies that his movements look natural. This is not only advantageous for tasks involving direct physical cooperation between humans and robots; even if a robot acts independently his movements should appear familiar and predictable to us humans. In addition to that, it is probably also necessary for a robot to have a human-like appearance to be accepted by society. Given the dynamic character of the environment in which humanoid robots are expected to work, they need to have a high degree of flexibility. They need to be able to adapt to changes in the environment and to learn new tasks continuously, and they are expected to carry out a huge variety of different tasks. This distinguishes them from industrial robots which normally only need to perform a small number of rather primitive tasks in a static environment. It seems impossible to create a humanoid robot with built-in knowledge of all possible states and actions. Therefore, there has to be way of teaching the robot new tasks.

Teaching a robot can be done in a number of ways, for example by means of a robot programming language or a simulation-based graphical programming interface. Another method is “teaching by guiding”, where the instructor operates a robot manipulator while its motion is recorded. The recorded motions are then added to the robot’s action repertoire. Such techniques are well-suited for industrial robots; however, in the domain of humanoid robots, where robots are expected to cooperate with unexperienced users, they suffer from several drawbacks. They are lengthy, complex, inflexible, require special programming skills or costly hardware [1]. That contradicts the purpose of humanoid robots which is to make life easier for us. It is essential that the control of such robots will not be too difficult and time-consuming.

An approach that addresses both issues (human-like motion and easy teaching of new tasks) is *Imitation Learning*: It facilitates teaching a robot new tasks and at the same time make the robot move like a human. Imitation learning is basically the concept of having a robot observe a human instructor performing a task and imitating it when needed. Robot learning by imitation, also referred to as *programming by demonstration* has been dealt with in the literature as a promising way to teach humanoid robots and several imitation learning systems and architectures based on the perception and analysis of human demonstrations have been proposed [2]–[7]. In most architectures, the imitation process proceeds through three stages: perception/analysis, recognition and reproduction [8]. An overview of the basic ideas of imitation learning in robots as well as humans is given by Schaal in [9].

In this paper we focus on a simple form of imitation: A movement is demonstrated to a robot multiple times 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. As described in Section III, we have not yet reproduced movements on a robot but instead simulated the reproduction stage using a software model that we created for this purpose.

Our work was largely inspired by previous work of Calinon and Billard. In [5] and [10], they describe an approach to imitation learning that makes use of Hidden Markov Models (HMM) [11] to learn and reproduce movements demonstrated by a human instructor multiple times (while HMM have become very popular for the *recognition* of gestures or speech, they have not been widely used for the *reproduction* of movements). After the hand path and joint angle trajectories have been perceived, the data is reduced to a subset of critical features in a preprocessing stage using certain criteria (see [10]). Separate HMM (one for the hand path and one for each joint angle trajectory) are trained with those “key points”. The HMM are subsequently used to recognize further demonstrations of the same movement as well as to imitate the demonstrated task.

II. OUR APPROACH

We implemented a similar approach as in [10] for dual-arm movements that also uses HMM to imitate movements shown to a robot multiple times. In our method, however, the data is preprocessed in a slightly different way, and, more importantly, not all states of the resulting HMM are used to reproduce movements. Instead, we try to identify characteristic features that can be observed in all (or many) demonstrations and only consider those when reproducing a movement. We also show how those common features can be used to detect possible temporal interdependencies between both arms in dual-arm tasks. Moreover, we track the orientation of the hand and not just its position, since the correct orientation is essential for carrying out complex tasks. In contrast to [5], we do not, however, try to determine which features of a movement (hand path, joint angles etc.) are most relevant for the correct imitation of a task.

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 (in our case, 7; see Section III). Those HMM are denoted by λ_p, λ_o and λ_j.

A. Preprocessing: Detection of characteristic features

A recorded movement is represented by a set of time-discrete sequences: Per arm there is one sequence $P_{d,1}, P_{d,2}, \ldots, P_{d,l(d)}$ that describes the positions of the TCP over time, one sequence $O_{d,1}, O_{d,2}, \ldots, O_{d,l(d)}$ that describes the orientations of the TCP, and seven more sequences $\theta^j_{d,1}, \theta^j_{d,2}, \ldots, \theta^j_{d,l(d)}$ that each specify the joint angle trajectory of joint j ($l(d)$ denotes the length of demonstration d). We omit the demonstration index d whenever it is not needed to distinguish between demonstrations. P_i and O_i are three-dimensional vectors. As in [10], we detect characteristic features of the perceived movement - key points - in a preprocessing stage and only use those to train the HMM. In doing so, we avoid having a high number of states and facilitate the matching of (or between) multiple demonstrations (see II-C). Those features are a subset of the aforementioned observation sequences. Since we try to detect distinctive features, it should still be possible to reconstruct the original movement well.

We detect key points separately for the position, orientation and joint angles of each arm. We use different criteria than [10] to identify key points, described as follows:

For joint angles, we use the following criterion: Let $\tau_{d,m}$ denote the time stamp of the m-th joint angle key point of demonstration d, i.e. its position in the sequence $\theta^j_{d,1}, \theta^j_{d,2}, \ldots, \theta^j_{d,l(d)}$. Then $\theta_{d,i} = (\theta^1_{d,i}, \ldots, \theta^J_{d,i})$, where J denotes the number of joints, is the m-th key point of the joint angles ($jK_{d,m}$, or just $K_{d,m}$ if the context is clear), $\tau_{d,m} = i$, if and only if

$$\begin{aligned} & \exists j: \quad \dot{\theta}^j_{d,i} = 0, i - \tau_{d,m-1} > \epsilon_1, |\theta^j_{d,i} - \theta^j_{d,\tau_{d,m-1}}| > \epsilon_2 \\ \vee \quad & \exists j: \quad |\theta^j_{d,i} - \theta^j_{d,\tau_{d,m-1}}| \geq \epsilon_3, i - \tau_{d,m-1} > \epsilon_4, \\ & \qquad |\theta^j_{d,n} - \theta^j_{d,\tau_{d,m-1}}| < \epsilon_3 \ \forall n \in [i, \tau_{d,m-1}) \end{aligned}$$

That means that whenever for any j, $\theta^j_{d,i}$ reaches an extremum (i.e. changes direction) or stops changing, sufficient time (ϵ_1) has passed since the creation of the previous key point and the joint angle at i differs from the angle at $\tau_{d,m-1}$ by at least ϵ_2, a key point $K_{d,l}$ is created. The second part of the condition ensures that when an angle that has stayed the same for some amount of time (ϵ_4) starts changing again, a key point is created as well. Figure 1 shows three different candidates for key points that given appropriate thresholds could all satisfy the above criterion. Any point between θ_b and θ_c, however, would not be a key point.

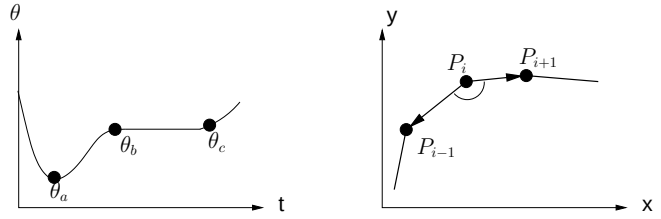


Fig. 1. Potential key points (left) and key point criterion (right)

For the orientation angles of the TCP (key points oK) the same criterion is used, but with possibly different threshold values ($\epsilon_5 - \epsilon_8$).

For TCP positions, we use a slightly different criterion. A point $P_{d,i}$ is a key point $pK_{d,m}$ (or just $K_{d,m}$) in the sequence of points $P_{d,1}, \ldots, P_{d,l(d)}$ if and only if

$$\begin{aligned} & \angle(\vec{P_{d,i}} - \vec{P_{d,i-1}}, \vec{P_{d,i}} - \vec{P_{d,i+1}}) < 180^\circ - \epsilon_9 \\ \vee \quad & ||\vec{P_{d,i}} - \vec{P_{d,i-1}}|| < \epsilon_{10}, i - \tau_{d,m-1} > \epsilon_{11}, \\ & ||\vec{P_i} - \vec{P_{d,\tau_{d,m-1}}}|| > \epsilon_{12} \\ \vee \quad & ||\vec{P_{d,i}} - \vec{P_{d,\tau_{d,m-1}}}|| \geq \epsilon_{13}, i - \tau_{d,m-1} > \epsilon_{14}, \\ & ||\vec{P_{d,n}} - \vec{P_{d,\tau_{d,m-1}}}|| < \epsilon_{15} \ \forall n \in [\tau_{d,m-1}, i) \ , \end{aligned}$$

where $\tau_{d,m}$ denotes the time stamp of the m-th position key point of demonstration d.

So if the angle between the vector that goes from P_i to its predecessor P_{i-1} and the vector that goes from P_i to its successor P_{i+1} is less than $180^\circ - \epsilon_9$ (see Fig. 1), P_i is considered a key point. In practice you would want ϵ_9 to be fairly high so that only sharp corners in the hand path would be detected as key points. Also, as for the joint angle trajectories, if the position remains unchanged for some time or starts changing again, a key point is created as well. Reasonable values for $\epsilon_1, \ldots \epsilon_{15}$ can be determined experimentally.

In [10] hand path key points are created only when there is a change in X, Y or Z direction. However, there can be significant changes in direction in a 3D path that do not necessarily result in a reversed direction of any single

coordinate. Our criterion for the hand path has the advantage that such key points are also detected because it takes the angle into account.

B. HMM structure

As mentioned before, we use multiple HMM for different kinds of observations (joint angles, TCP position, TCP orientation). Each HMM is trained with the key points of all demonstrations using the Baum-Welch algorithm for multiple observations ([12]). Each training sequence consists of the key points of the respective demonstration.

For a given observation sequence, the Viterbi algorithm (see [11]) returns the optimal state sequence of an HMM with respect to that observation sequence, i.e. the sequence of states most likely to generate that observation sequence. Therefore, if used on an HMM with the key points of one of the demonstrations as the observation sequence, the Viterbi algorithm would yield a sequence of states that could be said to correspond (in a probabilistic way) to the key points of the observed movements. Of course, the key point corresponding to a state might not occur in all demonstrations; a state could very well represent a key point of only one demonstration.

Since the states of the HMM roughly correspond to the key points, it seems reasonable to set the number of states equal to the number of key points. However, we use multiple demonstrations to train the various HMM and each demonstration might have different key points; therefore, the HMM should have as many states as there are distinct key points in all demonstrations. We do not know in advance, though, which of the key points of different demonstrations are equivalent, so we have to use an estimate for the number of states. In our experiments, we set the number of states to $\max_d k(d) \cdot 2$, where $k(d)$ denotes the number of key points in demonstration d.

This is of course not a very satisfying solution and we are still working on this problem. The standard HMM architecture that we use does not appear to be perfectly suited for our approach. What we would like to have is a model that had a certain number of main states for important key points that appear in most demonstrations and that allowed for inserting as many states as necessary in between those main states for key points that appear in only one or only a few demonstrations. It should also allow for skipping states that might be key points in most but not all demonstrations. Profile HMM, a special kind of HMM which are commonly used in bioinformatics to align DNA sequences, have those properties [13]. However, they use discrete output variables and we have yet to determine how such an architecture could be used with continuous variables and in the context of our method.

The HMM we use are left-right models [11], i.e. there are no state transitions between two states S_i and S_j if $j <= i$. That is because we want the models to reflect the sequence of key points over time and thus it should not be allowed to go backwards in time.

We use continuous HMM, with the output distribution in each state representing whatever is encoded in the HMM (for example, the TCP position or the angle of a specific joint) at a certain time given by the time stamp of the corresponding key point(s) (if multiple key points correspond to a state we compute the average of their time stamps and associate that average time stamp with the state).

The output probability is modeled by the density function of a multivariate Gaussian distribution. We do not use mixtures of such density functions - this is crucial if we want to use the HMM to reproduce a generalized movement because we consider the means of those functions to be the generalized positions, orientation angles and joint angles (as explained also in the next two paragraphs); as such, they have to be scalar values.

Before the training takes place, the HMM are initialized as follows: The initial state probabilities π_i are set to equal values that sum to 1. They are not particularly important since they get changed quickly by the Baum-Welch algorithm. We set the initial transition probabilities a_{ij} in such a way that a transition from a state S_i to a state S_j is most likely for $j = i + 1$ and less likely the higher $j - i$ gets. For the sake of simplicity and to make the models more robust, our covariance matrices cov_{ij} of the multivariate Gaussian density functions are diagonal matrices, which means $\text{cov}_{ij} = 0$ for $i \neq j$. The variances (cov_{ii}) are initialized with high values. The means μ_i are all initially set to 0 (unlike in [10] we do not initialize them with the key point values because we do not know in advance which state corresponds to which key point).

C. Common key points

As described earlier, the states of the Hidden Markov Models represent key points of the demonstrations. A key point may, however, only occur in some of the demonstrations. Thus, the question arises which states of the HMM should be used for the reproduction. In [10], the reproduction of a movement is triggered by another demonstration of the same movement. The Viterbi algorithm is then used to determine which states of the HMM match the key points of that demonstration most closely; those states are then used for the reproduction.

In our approach, however, the reproduction of a movement is not necessarily triggered by another demonstration of that same movement. We use only those key points for the reproduction of a movement that are common to all (or almost all) demonstrations and call them “common key points”. The reason for that is that a key point of a *single* demonstration is not necessarily a characteristic feature of the movement, in which case it would be reasonable not to consider it for the reproduction. But how do we know what key point in one demonstration corresponds to some key point in another demonstration? This is a classic matching problem which could for instance be solved by DP matching. However, we actually use our HMM to match key points across demonstrations. We use only those states of the HMM for the reproduction that represent key points which are shared by all (or almost all) demonstrations.

Formal definition: Let D denote the number of demonstrations. A common key point is defined as a 3-tuple:

$$C_j = ((i_{j,1}, \ldots, i_{j,D}), T_j, \nu_j),$$

where $(i_{j,1}, \ldots, i_{j,D})$ denotes the indices of the corresponding key points of the different demonstrations. The D-tupel $T_j = (\tau_{1,i_{j,1}}, \ldots, \tau_{D,i_{j,D}})$ contains the time stamps of those key points, while the n-tuple $\nu_j = (\nu_j^1, \ldots, \nu_j^n)$ describes the values of the trajectory to be reproduced (i.e. joint angles where n=number of joints, Cartesian coordinates where n=3 or orientation angles where n=3 as well) at that common key point. It should be noted that those values are not equal to the values of any key points; they are the means of the HMM's output density function of the state that corresponds to the key points $(K_{1,i_{j,1}}, \ldots, K_{D,i_{j,D}})$.

Detection of common key points: For each demonstration we have a list of key points: $K_{d,1}, \ldots, K_{d,k(d)}$, where d denotes the number of the demonstration, and $k(d)$ stands for the number of key points. By using the Viterbi algorithm on the HMM for each such sequence of key points, we get the sequences of states that correspond best to those key points (more precisely, the sequences of states that would most likely output the key points). Let $S_{d,j}$ denote the state that corresponds to the j-th key point of the d-th demonstration. Common key points can then be detected as follows:

$$\begin{aligned} &K_{1,i_{j,1}}, K_{2,i_{j,2}}, \ldots, K_{D,i_{j,D}} \text{ form a common key point} \\ &C_j = ((i_{j,1}, \ldots, i_{j,D}), T_j, \nu_j) \\ &\Leftrightarrow S_{1,i_{j,1}} = S_{2,i_{j,2}} = \ldots = S_{D,i_{j,D}} \end{aligned}$$

So only those key points of one demonstration that correspond to a state which also corresponds to key points in all other demonstrations are considered common key points. In figure 2 the bold circles are the optimal state sequence returned by the Viterbi algorithm for the key points of each demonstration. For example, S_4 corresponds to the key points $K_{1,4}$, $K_{2,3}$ and $K_{3,2}$. Clearly, only S_1, S_4 and S_5 represent common key points here.

In general, some of the demonstrations are likely to be erroneous, so it seems reasonable to ease the above constraint by saying that it suffices if the majority (say, 4/5) of the demonstrations have a key point that corresponds to the same state.

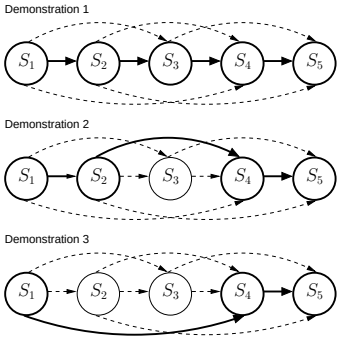


Fig. 2. Identifying common key points - S_1, S_4 and S_5 are the only states representing common key points

Generating a generalized movement with common key points: We can reproduce a generalized movement that is basically the average of all demonstrations and - more importantly - only uses the common key points of all demonstrations, as follows: We consider only those states of the HMM that correspond to common key points; we then take the means of the output density functions (PDFs) of those states (in order) and what we get is a sequence of positions, orientations and joint angles that can be used to reproduce the movement. However, two issues remain to be resolved before the movement can be regenerated from the common key points. First, what time stamps should be associated with each of those common key points? One possibility is to simply take the average of the time stamps T_j of all the key points that constitute a common key point. Second, the positions and orientations are of no use by themselves if we want to reproduce a movement on a robot or a software model of the human body; they have to be transformed to joint angles. That can be done with an inverse kinematics algorithm.

D. Reproduction

The detection of common key points is performed separately for each of the six HMM. Thus, we get sequences of points that are not necessarily in lockstep; they do not have to occur simultaneously. We interpolate between common key points to solve that problem. For example, for each common joint angle key point, we interpolate between the common position and orientation key points closest in time to determine the TCP position and orientation at the same time. That way we obtain a sequence of common key points of which each common key point has values for TCP position, TCP orientation and all joint angles of both arms.

Since the common key points only describe characteristic features of the trajectories and not the whole trajectories, we also have to interpolate between them to obtain actual trajectories that can be used to imitate a movement. Both spline and linear interpolation can be used for that purpose (we have only implemented linear interpolation so far).

So eventually we obtain time-discrete sequences for the TCP position and orientation as well as for all joint angle trajectories that each describe the movement at every point in time (using a certain sampling rate). The sequences of TCP positions and orientation angles are then transformed to joint angles with an inverse kinematics algorithm, so that they can be used for the reproduction.

The resulting joint angle sequences $\hat{\theta}_i^j$ are different from the joint angle sequences θ_i^j that are obtained directly from the joint angle HMM. A weighting factor $\omega \in [0, 1]$ is used to determine the relative influence of $\hat{\theta}_i^j$ and θ_i^j on the joint angles $\tilde{\theta}_i^j$ that are used to generate the final motion to be executed by the robot:

$$\tilde{\theta}_i^j = \omega * \theta_i^j + (1 - \omega) * \hat{\theta}_i^j$$

This weighting factor can be set by the user. Since the joints of the robot are not usually totally equivalent to the joints of the human demonstrator, the human joints have to be mapped to the robot's joints. This can be done in different ways: One can try to approximate the joint angle trajectories of the demonstrated movement as closely as possible, making the imitation look natural. That way the hand path might not be reproduced very well, though. Alternatively, one can try to imitate the hand path as accurately as possible using joint angles determined by an inverse kinematics algorithm which might deviate strongly from the joint angle trajectories perceived in the demonstration. The former is achieved by using a weighting factor of $\omega = 1$ while for the latter one would set ω to 0.

The whole process from perception to reproduction is depicted in figure 3. As in [10], the HMM can of course also be used to recognize (classify) movements. That has not been the focus of our work, though.

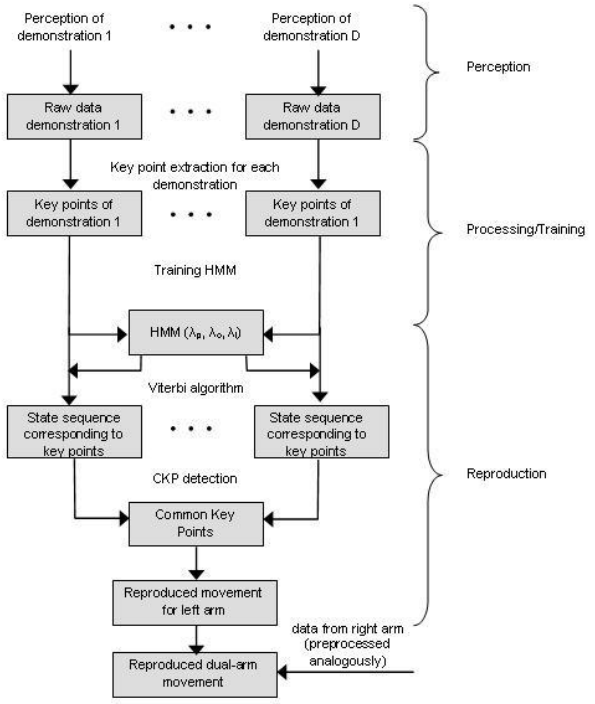


Fig. 3. Overview on the entire approach from perception to reproduction

E. Temporal coordination

When imitating dual-arm movements, often certain constraints have to be satisfied. Dual-arm movements can be uncoordinated or coordinated. No matter what method is chosen for the reproduction, information about coordination between the two arms can be very useful. There are various kinds of coordination; one way to classify two-arm coordination is to distinguish between temporal and spatial coordination. Spatial coordination means that the position and orientation of one hand is at least partially determined by the position and orientation of the other hand. As opposed to that, temporal coordination means that one arm must accomplish a certain subgoal as moving the hand to a specific position before the other arm may continue executing its movement. When imitating a two-arm movement, such dependencies should be detected and reproduced properly. For example, if with your left hand you pour water from a bottle into a glass that you hold in your right hand, you have to move the glass to its correct position before you can start pouring. It would be helpful if one could find out whether an observed temporal relation is just coincidence or whether it constitutes a necessary coordination. Multiple demonstrations allow you to determine the likelihood of some temporal relation being a true coordination.

The HMM approach in conjunction with the common key points introduced in this Section can be used to detect temporal relations between characteristic points of the left hand path and the right hand path that are unlikely to be coincidental.

Let $(\tau_{1,i_{r,1}}, \ \ldots, \ \tau_{D,i_{r,D}})$ be the time stamps of the key points that form a common key point C_r of the right arm's movement and $(\tau_{1,i_{l,1}}, \ \ldots, \ \tau_{D,i_{l,D}})$ be the time stamps of some C_l of the left arm. Then we can conclude that most likely there exists temporal coordination between the point C_r and C_l if in all demonstrations C_r is reached before C_l - or the other way round, i.e., if:

$$\forall k : \left(\tau_{k,i_{r,k}} < \tau_{k,i_{l,k}} \vee \tau_{k,i_{r,k}} > \tau_{k,i_{l,k}}\right)$$

It could also be possible that both arms must reach some points at the same time. That is accounted for by the following additional criterion for temporal coordination (δ is fixed and should be chosen close to zero):

$$\forall k : |\tau_{k,i_{r,k}} - \tau_{k,i_{l,k}}| < \delta$$

Once again, detecting those temporal relations only requires knowledge about common key points which may be acquired without HMM and may thus be implemented independently from the rest of our approach.

III. Human Kinematics Model

The goal of the methods described in this paper is to eventually have robots imitate human arm movements. To achieve this, we need to be able to perceive and analyze demonstrations, transform them to a robot joint representation and reproduce them. To begin with, we have so far only simulate the reproduction on a kinematic model of the human upper body with 18 degrees of freedom (Fig. 4), where each arm is modeled by nine DOFs: two in the inner shoulder joint (sternoclavicluar), three in the outer shoulder joint (glenohumeral), two at the elbow and two at the wrist ([14]).

To make imitation look natural, joint angles must be preserved as well as possible. However, measuring the joint angles of a human demonstration requires a complex tracking system (for example a marker-based system) which is not very usable in everyday life. So instead, we have incorporated two different

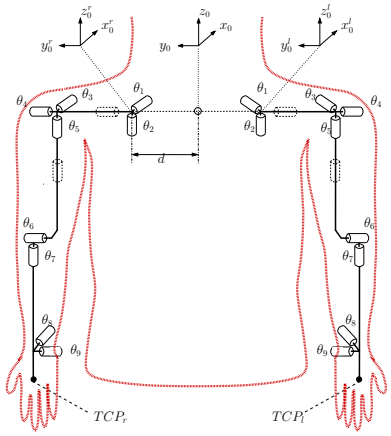


Fig. 4. The kinematics model of both arms

methods for the acquisition of the required data: a method based on a magnetic tracking system, and a purely vision-based markerless human motion capture system.

Using an easy to set up magnetic tracking system (Fastrak, `www.polhemus.com`), the position and orientation of both hands can be tracked. The joint angles can then be reconstructed using an approach based on neurophsychological studies. In [15] and [16], Soechting and Flanders have shown that arm movements are planned in shoulder-centered spherical coordinates and suggest a sensorimotor transformation model that maps the Cartesian wrist position to a natural arm posture using a set of representation parameters, which are the upperarm elevation, the forearm elevation, the upperarm yaw and the forearm yaw, respectively.

In neurophysiology evidence exists that arm and hand postures are independent of each other. This means that one can find the forearm and upper arm posture to match the hand position and then determine the joint angles for the wrist to match the hand orientation. In [17] we proposed a new algorithm which incorporates the physiological observation into a closed-form solution of the inverse kinematics problem to generate natural looking arm postures. For the reconstruction of the arm joint angles, we geometrically derived equations for the arm joint angles $\theta_3, \ldots, \theta_9$ in a closed form. The remaining two shoulder joints θ_1 and θ_2 supported by the arm kinematics model can be set manually by the user. For more details the reader is referred to [18].

The approach described in section II is based on the tracked hand path as well as the seven reconstructed joint angles.

Recently, we have also developed a purely image-based markerless human motion capture system [19], [20]. The input of the system are stereo color 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 preprocessed, generating output for the gradient cue, the distance cue, and an optional region cue, as described in [20]. 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

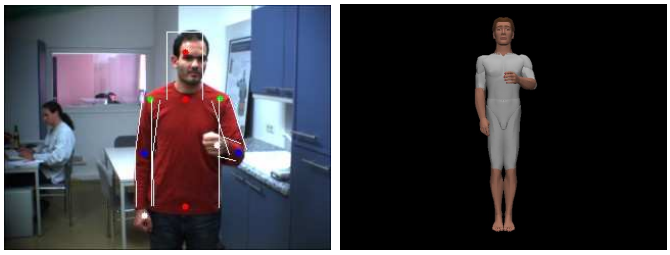


Fig. 5. Illustration of the performance of the markerless human motion capture system. Left: projection of the estimated configuration into the left camera image. Right: 3D visualization of the estimated configuration with an articulated human model.

compute the a-posteriori probabilities is formulated as:

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

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:

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

where $n := \dim(\mathbf{s})$ is the number of DOF of the human model. The transformation $f_i : R^n \rightarrow R^3$ 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 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 [20].

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.

IV. Experiments

We used the kinematic model described in the previous section to conduct several experiments. We tested the preprocessing, generalization and reproduction stages of our approach with three different dual-arm movements:

- pick-and-place task: we picked up a box with both arms, moved it to the right and put it down again
- pouring motion: we poured water from a bottle held by the left hand into a glass that was held by the right hand

- unscrewing motion: we unscrewed the lid of a jar (a complex motion that is different every time it is demonstrated; this did not work too well with our system)

The joint angles were obtained from the position of the hand using the algorithm based on findings in [15] and [16] as described in the previous Section. Our kinematic model was clearly capable of computing natural looking joint angles that way. For our experiments, we used the model to simulate the imitation of movements. We were able to display the simulated movement using a visualization of the human upper body that we created with OpenInventor and combined with our kinematics model. We implemented our own HMM for this system and did not make use of existing HMM toolbox.

Each of the movements mentioned above was demonstrated between five and ten times by the same person. Select training samples as well as the reproduced (generalized) hand path for the pick-and-place task are shown in Figure 6. It can be seen how the generalized trajectory is interpolated linearly between the common key points that were found using all demonstrations.

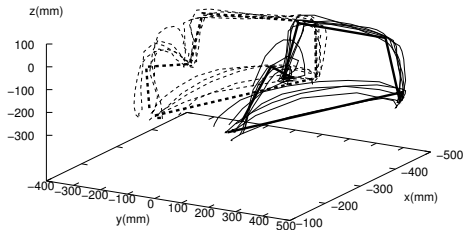


Fig. 6. Pick and place task: Training samples and the generalized trajectory for $\omega = 0$

Figure 7 shows only the reproduced trajectory of the left hand, but for three different ω. As expected, if the reproduction is based on joint angles and not only on the observed hand path ($\omega = 1$ or $\omega = 0.5$), the hand path becomes somewhat erratic. On the other hand, however, the reproduced movement seems more realistic and less artificial in that case which can be explained by the fact that more key points are created for joint angle trajectories than for the hand path (a joint angle key point is generated each time *any* joint changes its direction). If you used spline interpolation between key points instead of linear interpolation, you would probably obtain more realistic looking trajectories for $\omega = 0$ as well.

In Figure 8, the original joint angle trajectory of joint θ_7 (left arm, see also Figure 4) of one randomly selected demonstration (solid line) as well the generalized trajectory of that joint for $\omega = 0$ (dotted line) and $\omega = 1$ (dashed line) are shown. For $\omega = 0$ the spike at about $t = 20s$ is ignored because obviously no key point is detected in the hand path at that point in time. The key points detected in one of the demonstrations are shown in Figure 9 which seems to be what one would expect (characteristic features, i.e. significant changes in direction, have been detected).

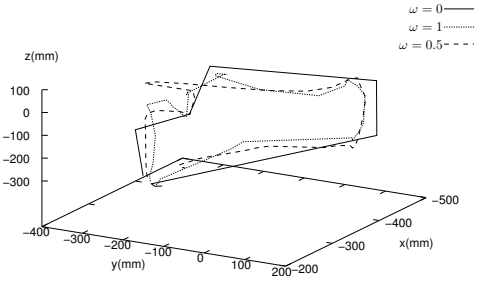


Fig. 7. Generalized trajectory of the left hand for different values of ω

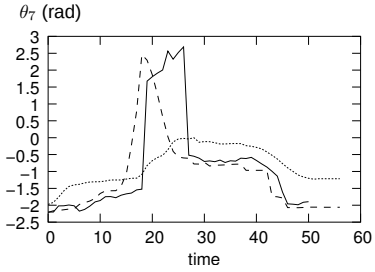


Fig. 8. The generated joint angle trajectory for joint θ_7 of the left arm

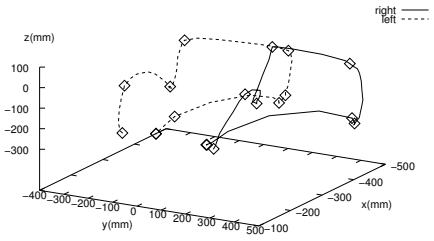


Fig. 9. All detected key points in the pick-and-place demonstration

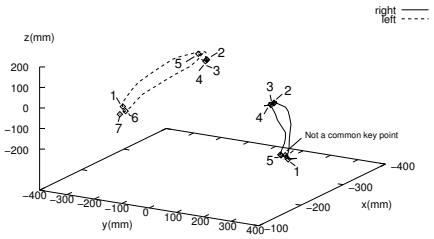


Fig. 10. All detected key points in the pouring motion

Figure 10 shows the key points detected in a training sample of the pouring motion. One of the key points is not a common key point, though, and will thus not be considered for the imitation of the movement.

Table I shows the temporal relations that were identified between the common key points of both arms. The numbers in the precondition column specify for each arm which common key point the other arm's hand must reach before the motion can be continued. Before the left hand is allowed to proceed to the third common key point, the right hand has to reach its second common key point, which is what one would expect.

CKP (right)	pre-condition (left)	CKP (left)	pre-condition (right)
1		1	1
2	1	2	
3		3	2
4	3	4	
5	6	5	
		6	
		7	

TABLE I

TEMPORAL RELATIONS BETWEEN COMMON KEY POINTS OF THE ARMS

Of course, a few other relations that might not be critical for the correct execution of the task were detected as well.

V. DISCUSSION, CONCLUSIONS AND FUTURE WORK

We presented an HMM-based approach for imitation learning of arm movements in humanoid robots. HMM are used to generalize movements demonstrated to a robot multiple times. Common key points in all demonstrations are identified and used for the reproduction of the movements. The results we obtained using a software model specifically created for this purpose show that our approach is an encouraging step in the effort to teach a robot new tasks in a flexible and natural way.

We would like to mention once again the work of Calinon and Billard in [5] and [10] due to the similarity to parts of our work. However, while our work was clearly inspired by their method, only the basic ideas are based on it. We have introduced new aspects such as the detection of common key points and identifying temporal coordination between two arms and use a different preprocessing method. The kinematic models presented in Section III are also part of the contribution of this paper.

So far, we have only simulated generating generalized trajectories, using a kinematic model of the human arms. A natural next step would be the reproduction of movements on the humanoid robot ARMAR-III. That would involve mapping the joints of our kinematic model to those of the robot and will be part of future work. Another aspect of our method that should be improved is how the trajectories are encoded in the HMM. As described in Section II, a generic HMM architecture does not seem to be ideal for our approach.

Also, our system is not invariant to translations or rotations yet, a highly desirable property for any imitation learning system to be used in practice.

Furthermore, it seems essential for the accurate imitation of tasks to take into account objects that are manipulated during the demonstration. Not only do we need to make sure that an imitated movement results in the same manipulation of an object as the demonstrated movement, but considering objects would also be a significant step towards recognizing the goal of a task, which we have not attempted to do yet but which is clearly an important aspect of imitation learning.

ACKNOWLEDGEMENT

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 (SFB 588) funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

REFERENCES

[1] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 799–822, 1994.

[2] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[3] A. Billard and R. Siegwart, "Robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 65–67, 2004.

[4] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng, "Discovering optimal imitation strategies," *Robotics and autonomous systems*, vol. 47, pp. 69–77, 2004.

[5] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005)*, 2005.

[6] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICML 1997)*, 1997, pp. 1040–1046.

[7] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London: Series B, Biological Science*, vol. 358, no. 1431, pp. 537–547, 2003.

[8] Y. Demiris and G. Hayes, "Imitation as a dual-route process featuring predictive learning components: a biologically-plausible computational model," *Imitation in animals and artifacts*, pp. 327–361, 2002.

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

[10] S. Calinon and A. Billard, "Stochastic gesture production and recognition model for a humanoid robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005, pp. 2769–2774.

[11] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, 1989, pp. 257–286.

[12] J. Bilmes, "A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," University of California - Berkeley, Tech. Rep., 1997.

[13] S. R. Eddy, "Profile hidden markov models," *Bioinformatics*, vol. 14, pp. 755–763, 1998.

[14] T. Asfour, "Sensomotorische Bewegungskoordination zur Handlungsausführung eines humanoiden Roboters (german)," Ph.D. dissertation, University of Karlsruhe, Karlsruhe, Germany, 2003.

[15] J. F. Soechting and M. Flanders, "Sensorimotor Representations for Pointing to Targets in Three-Dimensional Space," *Journal of Neurophysiology*, vol. 62, no. 2, pp. 582–594, 1989.

[16] ——, "Errors in Pointing are Due to Approximations in Targets in Sensorimotor Transformations," *Journal of Neurophysiology*, vol. 62, no. 2, pp. 595–608, 1989.

[17] T. Asfour and R. Dillmann, "Human-like Motion of a Humanoid Robot Arm Based on Closed-Form Solution of the Inverse Kinematics Problem." in *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA, 2003.

[18] F. Gyarfas, "Imitation Learning zur modellbasierten Generierung menschenähnlicher Zweiarm-Bewegungen," Master's thesis, University of Karlsruhe, Germany, 2005.

[19] 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.

[20] 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.

Coaching: An Approach to Efficiently and Intuitively Create Humanoid Robot Behaviors

Marcia Riley*, Aleš Ude†‡, Christopher Atkeson†§, and Gordon Cheng†¶
*College of Computing, Georgia Institute of Technology
Atlanta, Georgia 30332–0250, USA, Email: mriley@cc.gatech.edu
†ATR Computational Neuroscience Laboratories, Dept. of Humanoid Robotics and Computational Neuroscience
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
‡Jožef Stefan Institute, Dept. of Automatics, Biocybernetics, and Robotics
Jamova 39, 1000 Ljubljana, Slovenia, Email: ales.ude@ijs.si
§Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA, Email:cga@cmu.edu
¶Japan Science and Technology Agency, ICORP Computational Brain Project
4-1-8 Honcho, Kawaguchi, Saitama, Japan, Email: gordon@atr.jp

***Abstract*— The advances in humanoid robots in recent years have given researchers new opportunities to study and create algorithms for generating humanoid behaviors. Not surprisingly, most approaches for creating or modifying behaviors for complex humanoids require specialized knowledge and a large amount of work. Our aim is to provide an alternative, intuitive way to program humanoid behavior. To do this, we examine human-to-human skill transfer, specifically coaching, and adapt it to the humanoid setting. We enable a real-time scenario where a person, acting as a coach, interactively directs humanoid behavior to a desired outcome. This tightly coupled interaction between a person and a humanoid allows efficient, directed learning of new behaviors, where behavior characteristics can be modified on-demand. Communication is realized through demonstration and a coaching vocabulary, and changes are effected by transformation functions acting in the behavior domain.**

I. Introduction and Related Work

In film and literature we often see people interacting with robots just as they do with other people: for example, they use natural communication such as speech and gesture to direct robots. In this fictional world, even people who are not robot experts can control complex machines including humanoids with ease. However, in today's reality creating behaviors for humanoid robots remains a task for specialists, where communication of behavior details is often time-consuming and takes place largely through the mechanisms of programming.

One way we can begin to address this disconnect between imagined possibilities and current reality is by focusing on paradigms which afford more intuitive methods for creating robot behaviors. Human-to-human skill transfer is an especially interesting model for building robot behaviors, as, besides efficiency, it offers a familiar context to people dealing with humanoids: rather than learning special skills, people can bring their own knowledge from interacting with each other directly into the humanoid domain. In this work we develop an approach to generating robot behavior modeled on a particular type of skill transfer: coaching, where a robot acquires new skills with guidance from a human coach. For this work, we explore the specific behavioral domain of movement. Where possible, we emulate the efficiency of human skill transfer, and because of the familiar, high-level control afforded by coaching we enable non-specialists to participate in creating robot behaviors.

Other robotics researchers inspired by coaching include Nakatani and co-authors [1], who use coaching to aid in balance and walking controller design for a biped robot. Their experiments nicely demonstrate the efficiency gains of introducing intuitive human instruction into the controller design loop, and although their solution is directed toward specialists, the authors encourage creation of adaptable interfaces to allow non-specialists a role in such control. Our approach is more general purpose, targeting trajectory-based movement acquisition and subsequent refinement, and provides mechanisms for novel behavior acquisition and an interface with affordances suitable to specialists and non-specialists alike.

In [2] robot coaching is used in a teaching scheme for a mobile robot where the emphasis is on learning representations for high level tasks rather than on motor skill acquisition. The coaching component, like our system, uses both demonstration and verbal input to direct a robot, although demonstration in [2] is limited to recognizing known primitives, and new behaviors are limited to combinations of these primitives.

In interactive evolutionary computation (IEC), human evaluation is used in optimizations as fitness functions [3], and although especially suited to topics like music retrieval where subjective evaluation is critical, IEC has proven useful in a number of fields including robotics [4]. It differs from coaching, however, in that evaluations usually take the form of selecting preferences from a range of current possibilities, while in coaching specific feedback about how to improve a performance is given.

Motion editors have also been used to create new robot [5] and virtual human behaviors [6], [7], [8]. In [5] Kuroki and colleagues present a motion editor specifically designed for a small biped robot using the graphical tools common in motion editors such as inverse and forward kinematics modes, pose control, pose interpolation, and blending functions. Our

approach differs from this and from most motion editors from the graphics community in the way the user interacts with the robot: our human-robot communication takes the form of a coach's demonstrations and high-level qualitative instructions, while motion editors offer powerful but less intuitive motion editing paradigms requiring more training to master. In addition, our system keeps live robot performance in the loop, allowing for timely evaluation by the coach.

In the next sections we discuss the role of a coach in motor skill acquisition, followed by our adaptation and implementation of useful coaching formalisms comprising our humanoid coaching system, including domain-specific vocabulary, transformation functions, modes of demonstration, and mechanisms for focusing student attention in both time and body space. We then discuss our experiments coaching a robot in catching, and in throwing a ball into a basket. All exchanges occur in a real-time interactive setup that preserves the iterative nature of coaching and the tight coupling among effort, evaluation and guidance.

II. The Role of a Coach in Human Skill Transfer

In building our humanoid coaching system, we first studied human coaching, with particular emphasis on the role of the coach in teaching motor skills. In general, a coach is an expert whose job is to improve the performance of a student. This means providing instructions which are incorporated into the student's learning sessions to produce a successful outcome. Coaching, being a well-established field, offers us a number of formalisms for teaching new skills. These include acquiring new motor knowledge; focusing attention on relevant task features to improve learning of critical task aspects; assigning priorities among goals; giving specific feedback to improve the performance; giving a strategy for correction; and helping to iteratively define the characteristics of a successful outcome. These coaching methods imply a tightly coupled interaction between coach and student where close observation of student performance is followed by feedback or further instructions from the coach.

The role and usefulness of an expert to guide a student has been well-studied in humans. Performance and learning varies with the form of the supplied information, its amount and its timing. Frequent ways instructors give information are by showing videotapes of a person performing the task, directly demonstrating the task, physically guiding a person through a task, and providing verbal instructions. With the right guidance at the right time the student can adjust behavior both during and after a learning session until the desired motion or state is attained.

Students use live or video demonstration to observe strategies, spatial or temporal information, and as a reference of correctness for their own attempts at the behavior [9]. Some researchers have shown that mistakes may be more instrumental in facilitating learning than perfect performances, which by themselves are not giving the type of information the learner needs.

Several studies, however, found that showing videotapes alone, which is similar to direct demonstration, often did not improve motor learning [9]. It was postulated that too much information is available, particularly for complex tasks, and the viewer does not know which details are important to the outcome. In one study, showing a videotape by itself was even shown to hinder learning. On the other hand, as early as 1952, verbal instructions were shown to have a lasting effect on learning and performance, although verbal instructions are more useful when used in conjunction with other input, particularly demonstration [9].

Verbal instructions can communicate information including focus, specific stance, or strategies for error correction. Some verbal information takes the form of specific kinematic feedback, such as "bend your knees". Besides patterns of coordination, kinematic feedback can also be position, velocity, and acceleration information. Expert instructors play a valuable role in being able to observe, identify and correct kinematic errors by giving verbal descriptions to the student. The usefulness of kinematic information is supported by studies giving evidence of kinematic trajectory plans in the parietal cortex [10]; the presence of inverse dynamics models in the cerebellum [11]; and motor equivalence where different limbs are shown to produce kinematically similar patterns, despite having such different dynamical properties [12], [13].

In the next sections we discuss implementation of coaching components pulled from these ideas and tied together by an interface used in directing humanoid behaviors.

III. The Humanoid Robot Coaching System

A. Overview

In our humanoid coaching system the coach, much like a dance instructor or sports coach, wishes to change a given behavior to suit a particular end. In order to achieve this, the coach and humanoid must be able to communicate. The interface shown here facilitates and coordinates this communication. Embedded in it are access points for the different capabilities of the system which incorporate:

- vocabulary;
- a set of transformation functions;
- the ability to demonstrate a desired behavior, either through performance or by physically guiding the robot;
- the ability to focus on specific parts of a behavior for refinement (body and time segmentation);
- the ability to clarify instructions or resolve ambiguities through a student-coach dialogue.

Each capability is derived from an aspect of human coaching. The vocabulary, for instance, reflects verbal instructions coaches commonly use to give instructions. These commands center around kinematic descriptions of motion, such as *higher* and *bend*, used often when teaching motor skills. Movements are changed by transformation functions (TFs) articulated by this high-level vocabulary which manipulate appropriate behavioral parameters to achieve a specific outcome (see Section III-B for details). New movement acquisition is based

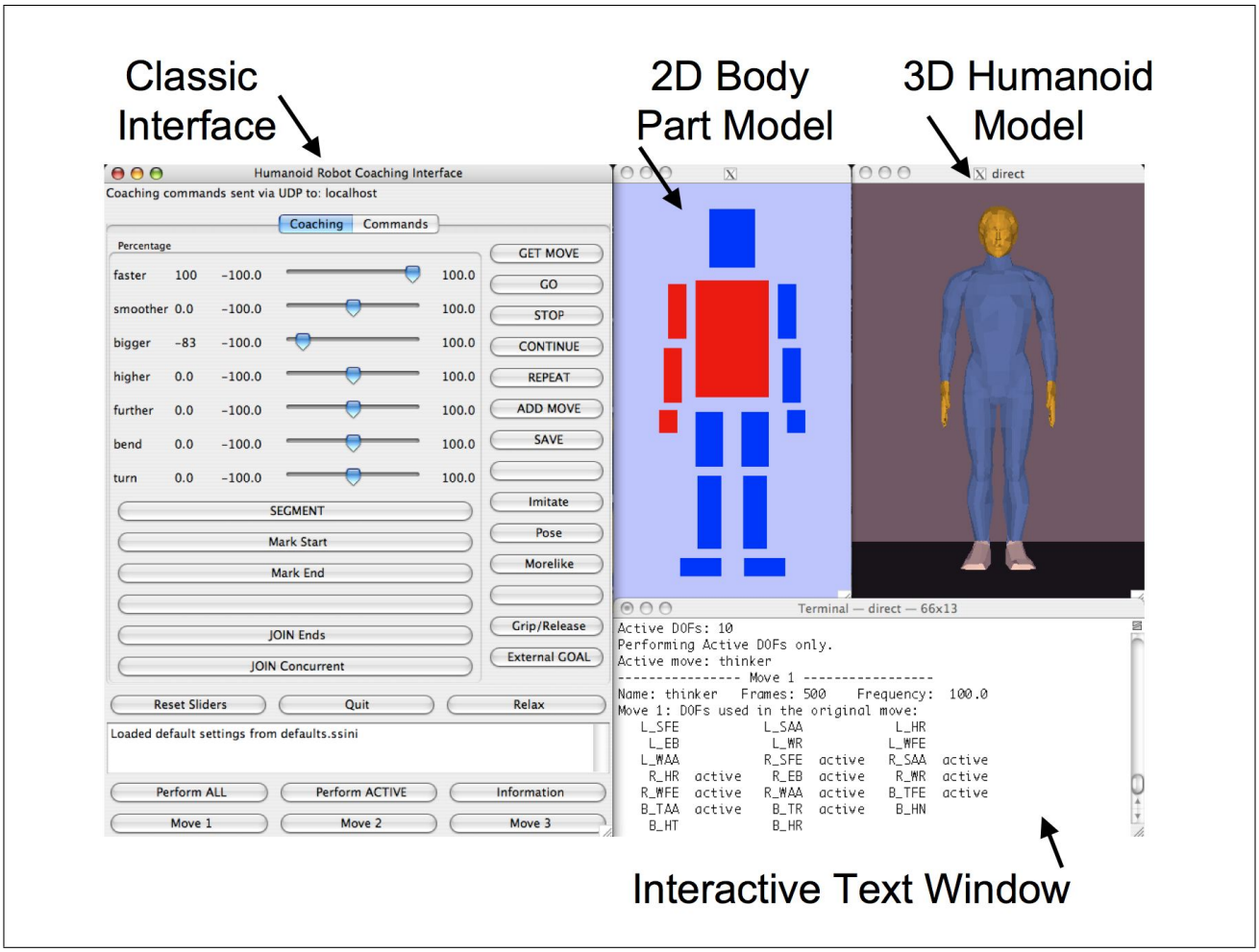


Fig. 1. The four modules of the humanoid robot coaching interface.

on two widely-used methods: demonstration and guiding. Focusing on behavioral features relevant to success as defined by the coach is achieved by selecting specific parts of a movement, such as arm or leg motion (segmenting in body space) for coaching. Attention can also be focused on certain sections of a movement (segmenting in time) by breaking it into sub-movements. Composition of partial movements into a complex movement is easily accomplished by joining segmented sub-movements. Lastly, during human coaching, students are free to ask for clarification when misunderstandings arise. We emulate this by giving the robot the ability to initiate a dialogue with the coach to ask for further instructions when faced with ambiguous or unclear situations.

The interface itself is shown in Fig. 1 and is comprised of modules representing the different functionalities. They are:

- A classic interface comprised mainly of buttons and sliders labeled with various coaching commands making up the explicit coaching vocabulary.
- A simple 2D representation of a robot body allowing the coach to easily focus changes on any part(s) of the body.
- A 3D graphics window which allows visualization of movements on a 3D humanoid to allow quick, intuitive segmentation, and real-time 3D visualization of color markers used in vision-based demonstrations.
- An interactive text-based window to facilitate student-initiated dialogue between coach and student, and to provide current state information to the coach on demand.

Information transfer is initiated by using the vocabulary on the classic interface. We use this type of interface for many higher-level (”verbal”) instructions in order to avoid the pitfalls of speech processing, such as the need for speaker-specific training, although the system has also been successfully tested with speech recognition software.

B. Transformation Functions

At the heart of the system lie transformation functions, which form the essential mechanism for bringing about changes in robot behavior. A TF is typically comprised of a label, which is the coaching command that invokes it, and a set of criteria that serves to define the high level command in terms of low level behavioral criteria. Label and criteria are wrapped together in a function that ultimately effects changes to the appropriate behavioral parameters in accordance with the TF’s definition.

C. The Role of World and Self Knowledge

To set the criteria for TFs, the system needs access to certain types of knowledge relevant to the behavior domain. For the movement domain the robot needs an understanding of the relationship between its body and the world. In people, body and world knowledge for movement is gained from childhood on, beginning when children explore the space around them with seemingly random gestures. In our system we seek a minimal knowledge representation that affords the robot the same type of understanding.

We designate world and body (self) reference frames with a known correspondence, each comprised of a 3D Cartesian system where the axes correspond to left, up and front. At any time the robot is able to map its own local orientation to the world reference frame. A TF is defined as relative to either the world or body frame. For example, the notion of "front" and "back" embedded in the *further* TF is always relative to the robot body frame, so the current robot body orientation is used no matter where the robot is in the world, while *higher* is always relative to the world frame. Taken together, the TFs begin to define a type of domain-specific dictionary of behavioral knowledge.

Body knowledge in the humanoid coaching system is also represented in the form of kinematic chains whose connectivity is known to the robot. In our system, the interdependencies of the human skeleton are represented as 6 hierarchically dependent kinematic chains. By exploring the relationship of the robot body joints to the appropriate Cartesian reference frame, the robot can determine which joints may be useful in effecting change for a specified direction. For example, the robot may find that a higher arm movement could be accomplished by extending the arm front and up (shoulder flexion/extension) or to the side and up (abduction/adduction), or some combination of the two. Additionally, knowing its body connectivity, a robot may suggest using the torso to effect changes in an arm posture. In determining which changes to make, the robot engages in a dialog with the coach (see the appendix) resulting in the final set of relevant DOFs used to effect the change. During this exchange, the robot can demonstrate the effect of the candidate DOFs to provide immediate feedback to the coach.

DOF exploration starts with the body parts selected by clicking in the 2D window, which graphically represents body part vocabulary (right arm, head, etc.) in a simplified robot shape. Body part(s) are highlighted (in red) when active, and each part corresponds to a set of candidate DOFs that are considered in effecting subsequent changes. This selection process works in conjunction with the *Perform ACTIVE* and *Perform ALL* options on the classic interface which direct the robot to perform changes using only the selected DOFs, or with all DOFs involved in the movement. With this mechanism, the coach has the option of seeing the effect of partial changes on the entire movement while refining specific pieces.

To determine appropriate DOFs, the robot makes use of forward kinematics where each joint change is related to a change in the 3D positions of virtual points attached to the relevant body part. Our robot is comprised of revolute joints modeled with twists [14] as in our previous work [15], [16], [17]. Each candidate joint is moved by respectively increasing and decreasing its value, and the change in 3D point position attached to the body part moved by the joint is then compared to criteria for the TF, where the position of a point after rotation is given by

$$\boldsymbol{P_{t+1}} = \boldsymbol{g}(\boldsymbol{R}, \boldsymbol{d}) \cdot \boldsymbol{exp}(\hat{\boldsymbol{\omega}}\boldsymbol{\theta}) \cdot \boldsymbol{P_t} \tag{1}$$

where $\boldsymbol{P_t}$ and $\boldsymbol{P_{t+1}}$ are the initial and final 3D positions respectively of a point attached to the body part given in the body coordinate system, $\boldsymbol{g}(\boldsymbol{R}, \boldsymbol{d})$ is the homogeneous matrix representing the body orientation and position in the world coordinate frame, and $\boldsymbol{exp}(\hat{\boldsymbol{\omega}}\boldsymbol{\theta})$ is the exponential that maps a rotation of angle θ radians about $\boldsymbol{\omega}$, the unit vector in the direction of the joint axis, to the corresponding rotation matrix. (Note that for the special case of pure rotation, the exponential coordinates of rotation, θ and ω, suffice in the place of the twist coordinates, and the exponential mapping can be efficiently calculated by Rodrigues' formula.)

When both rotational directions match the TF criteria, the solution prefers to continue in the current direction of motion, but the final decision is left to the coach.

For world-based criteria like *higher*, it is important to test DOFs with respect to the robot's world position and orientation since changes therein can affect the solution set of DOFs. (Consider making a higher arm movement lying down versus standing, for example.)

D. Initial Behavior Acquisition

Another important use of domain knowledge is found in imitation, where the coach demonstrates movements that can be understood and reproduced by the robot during interactive coaching sessions. It is not surprising that imitation plays a key role in coaching motor skills, as it is a successful and fundamental strategy used for human learning [18], and has inspired much work in the robotics and virtual human communities [19], [20], [21].

To solve direct imitation, the robot already has crucial information: its position and orientation with respect to the world reference frame, and an understanding of its own body configuration.

Our approach, described in [15], [16], [17] relates the coach's kinematics to the robot's kinematics automatically, and acquires the motions in the robot joint space by matching the position of markers in Cartesian world space attached to the coach's body to the motion of corresponding virtual markers attached to the robot body and measured in body space.

In the past we have used a commercial optical motion capture system with active markers and trailing wires to track points on the body, but for coaching we usually use our own less intrusive (wireless) color tracking system, which tracks color blobs attached to clothing (Fig. 2). During coaching, imitation occurs in real time or immediately following a

demonstration, and the solution is constrained to the robot joint limits.

In the coaching system, the *Imitate* command is used with the 3D window to allow real-time display of 3D vision markers attached to the coach, and to visualize solution markers as the transition from Cartesian space to joint angles is calculated. This is important in ensuring good tracking information is maintained, a reasonable solution ensues, and problems such as occlusion can be quickly identified and monitored.

Another common method of seeding behaviors is physically guiding a robot through a motion. This is invoked with the *Pose* command and is accomplished by lowering gains on the robot and directly capturing joint angles while the coach physically guides the robot through a motion.

The last demonstration-based command, *Morelike*, is intended to make a movement similar to the movement being shown. This is achieved by performing a weighted average on joint angles for each DOF used in the demonstration and in the current movement to drive them toward the demonstration.

E. Descriptions of Transformation Functions

Due to space constraints we present only brief descriptions of the remaining transformation functions, omitting most of the mathematical details. TFs were implemented using tools from various areas including digital signal processing, spline analysis, approximation theory, and computer vision.

We chose Cartesian and joint angle space to express movement information because they reflect common spaces for describing movements in human coaching, and lend themselves easily to change within this paradigm. Movements, M, are represented either by a sequence of points P_t in time, splines or radial basis functions, and transformation functions act on these representations.

At the top left of the classic interface, we find motion descriptors and associated sliders, which control the magnitude of the desired changes bounded by the robot's capabilities. *faster* changes the frequency of the movement under consideration, where robot velocity capabilities limit desired frequencies if necessary. *smoother* requires less sharp changes in position with respect to time. This is achieved using a moving average filter which smooths a curve in joint space representing the active motion segment (See Fig. 3). The slider value influences the filter window size. *bigger* corresponds to an increase in amplitude of the movement range measured in joint space and is achieved using a global scaling algorithm [22]. *higher* causes an increase along the vertical axis of the world Cartesian system, and is accomplished by moving the maximum (or minimum) of the current trajectory toward the robot's maximum joint position with a blending function. *further* directs the motion either further left or right, or front or back with respect to the robot body. *bend* bends a part of the body (e.g.,elbow, knee or waist) by increasing the appropriate joint angle over the movement segment under consideration. *turn* orients the body (here, the torso and head) right or left relative to body space, or toward an object in its surroundings.

Next we consider the time segmentation commands *SEGMENT*, *JOIN Ends*, and *JOIN Concurrent* that allow the coach to split a movement into sub-movements or join two movements together. The coach can visualize a movement in the 3D humanoid window to quickly select the beginning and end of a segment using the *SEGMENT*, *Mark Start* and *Mark End* buttons. Once a movement segment is identified, instructions from the coach will affect only this segment until segmentation is turned off.

In the case of *JOIN Ends*, the end of one movement is joined to the beginning of the second movement. When the two joined movements have different frequencies, relative frequencies are preserved by re-sampling the slower segment represented by splines at the higher frequency. *JOIN Concurrent* aligns the start of two segments and merges them into one. This action is intended to join movements with different DOFs (legs plus arms, for example), allowing the coach to create complex movements from simpler ones. The buttons *Move 1*, *Move 2* and *Move 3* allow the coach to switch between movements and select movements to be joined.

When movement segments are joined care is taken to smoothly blend the end and start of adjacent segments to avoid sharp discontinuities in the motion. In all cases the robot's joint limits (position and velocity) act as constraints during modifications, and joint velocities and accelerations are computed by finite differencing after position changes.

Also on the interface are the object interaction commands *Grip/Release* and *External Goal*. The first allows the coach to tell the robot when to grip or release objects in its hand, while the second tells the robot that the current behavior is associated with an external object found in its environs.

The remaining commands are meta-commands which control the flow of the overall coaching session (*GET MOVE*, *GO*, *STOP*, etc.); or housekeeping commands such as *Relax*, which resets the robot posture to reasonable values.

IV. Experiments and Results

Our previous work showed the feasibility of using real-time full-body imitation for movement acquisition [15], [16], [17]. Here we discuss our work on coaching the robot to throw and catch a ball where our student is a 30 DOF humanoid robot [23] shown in Fig. 5. The gross movement for throwing was acquired from direct demonstration using computer vision (see Fig. 2). The original trajectory acquired from the vision data, shown in Fig. 3, was too noisy for the robot to properly execute. So our coaching sequence was as follows:

- acquire a set of throwing movements using real-time demonstration;
- select one of the movements and use *SEGMENT* to extract the relevant part of the trajectory for the desired throw;
- smooth the movement several times, each time acting on the previous results with *smoother* (Fig. 3).

With an acceptable throwing movement, we could now focus on coaching the robot to throw the ball toward the basket. To do this we

- increase the velocity and acceleration with *faster*

Fig. 2. The initial throwing behavior was captured and processed in real time using color markers attached to the body and computer vision techniques.

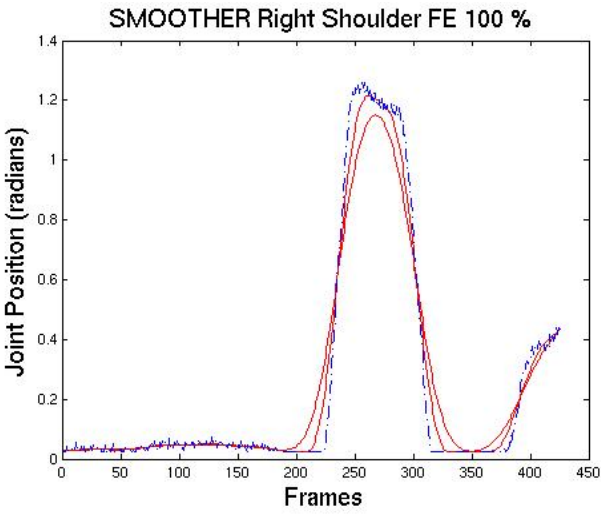


Fig. 3. Original (dashed, noisy line) and modified trajectories for the right shoulder flexion/extension DOF showing modification by two iterations of the *smoother* transformation function implemented with a moving average filter.

- change the course of the trajectory with *higher* (Fig. 4) to extend the length of the throw,
- use *release* to specify the exact timing for the release.

During the coaching session, the robot demonstrated how *higher* can be accomplished using a variety of DOFs, and let the coach select the appropriate DOFs (shoulder and elbow flexion/extension) to make the new movement. After each refinement, we (the coaches) watched the robot to evaluate its performance, and then gave successive instructions based on what we saw. Throwing at this point was much improved, but still not satisfactory. This led us to constrain the body space for the movement from DOFs originally used in the movement to the DOFs most relevant for successful robot throwing until throwing was successful.

We then moved the basket, and again coached the robot until it could throw successfully to the new location. In the second coaching sequence, *further* was instrumental in directing the movement toward the robot's right, particularly for the robot torso, as the new target was further to the right. It is important to point out that the acquisition of this behavior was accomplished without any programming and without the input of accurate parameters like velocities and accelerations. The initial trajectories were acquired by observation and then modified using qualitative higher-level instructions. Fig. 5 shows a sequence of postures from a coached throwing movement.

In our catching experiments, we used coaching to improve the performance for an existing catching behavior [24]. In this case we used the transformation function *higher* to change the height where the robot catches the ball. This parameter had an effect on the time it took to catch the ball, with lower catches affording more time to plan and execute an intercept motion. *GO* was used to specify when to begin prediction of the ball's flight. For different types of ball trajectories, different parameters led to successful catching. Our system supports

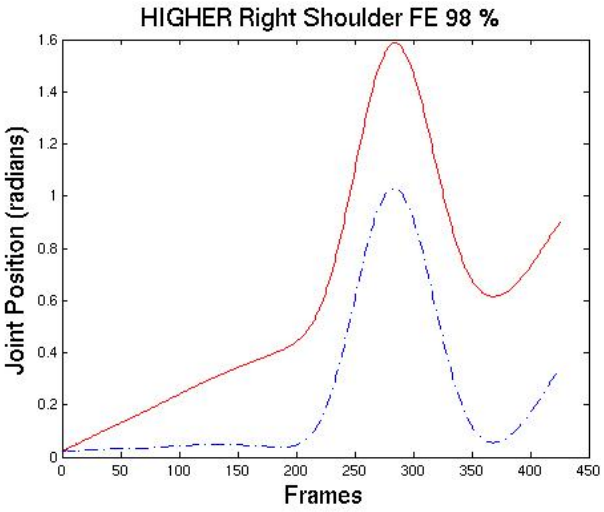


Fig. 4. Original (dashed line) and modified (solid) trajectories showing modification by the *higher* transformation function after using *smoother*.

permanently associating the relevant behavior parameters to the movement primitives and thus expanding the knowledge base of the robot.

V. Conclusions and Future Directions

The presented system explores a new way to intuitively create behaviors for complex humanoid robots. Currently, much time is spent by specialists in creating each new behavior. Our intent is to introduce other methods with the potential to improve the time and ease of creating behaviors. Efficiency is often facilitated by intuitive solutions, as they are easy to understand and require less training to use. As we examined strategies people use to acquire new skills, we were inspired by coaching's proven merits in accelerating human skill acquisition. In addition, and perhaps because of

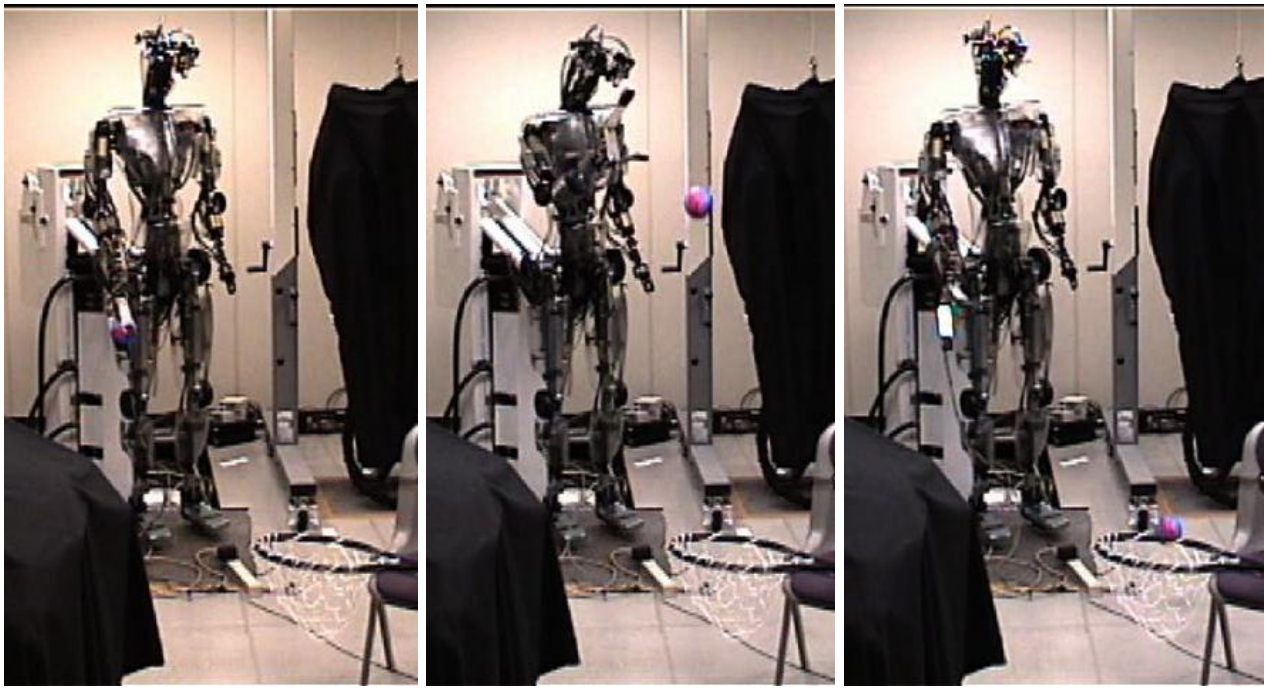


Fig. 5. Postures from a sequence of coached throwing movements.

its success in accelerating learning, coaching is a paradigm familiar on some level to most people. It is a special case of a more general teacher-student relationship that we meet from our infancy forward.

Because of this, our coaching system offers 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. This familiarity allowed us to create a "walk up and use" type of system, where, unlike many motion editing systems, little previous training is needed, and, unlike most current robot control schemes, non-specialists can participate in implementing complex robot behaviors such as throwing a ball in a basket. In doing so we do not obviate the need for specialists to create low-level algorithms for robot control. Instead, we look at the potential role of introducing the advantages of interactive high-level instruction and interactive goal specification used often by people in improving the overall efficiency of creating new robot behaviors. Our approach brings a collaborative nature of problem solving to the domain, where the intent is for widespread availability, ease of use, and the ensuing behavioral flexibility and customization these methods make possible.

Consistent with these goals, we wish to develop new methods for adding transformation functions to the system. The functions described here represent examples of domain-specific transactions related to the language of motion, but are not meant to be an exhaustive list. At present, more transformations can be added as needed by traditional programming methods. However, it would be more suitable and interesting to develop a mechanism for learning new transformations and attaching them to a particular label without the need for such programming. We will work on this in the future.

VI. Appendix

The following exchange shows an excerpt from an interaction between the robot and coach during a *higher* command. The position of a virtual point on the upper arm at its current position and after a positive and negative rotation from the current position is shown. An increase in the second (y) dimension corresponds to an increase along the vertical world axis, the criteria for *higher*. The main points of the robot's communication to the coach are shown in bold. The coach's responses are shown in italics. The robot first checks all active DOFs (those corresponding to body parts selected in the 2D window, here the left upper arm), and then checks any connected parts (here the torso) whether they are active or not to suggest additional possibilities to the coach.

HIGHER requested.
....checking right shoulder

Potential candidates to help with UP for this part:

DOF	Status:
...shoulder flexion/extension	Active (rsfe)
...shoulder abduction/adduction	Active (rsaa)
...shoulder rotation	Active (rshr)

I could also check:

....torso rotation	Not Active (btr)

....torso abduction/adduction Not Active (btaa)
....torso flexion/extension Not Active (btfe)

Cartesian frame changes:
x y z
testing dof shoulder flexion/extension (rsfe)
-10.296700 4.763384 2.019819 (starting position)
-10.296700 11.492188 5.085141 (positive rotation)
-10.296700 3.356163 -1.992737 (negative rotation)

testing dof shoulder adduction/abduction (rsaa)
-11.976195 4.334291 -2.170500
-10.514429 3.448190 -2.170500
-14.950410 12.791493 -2.170500

testing dof shoulder rotation (rhr)
-10.179647 3.349600 -1.507159
-9.144885 3.349600 -4.726161
-8.644364 3.349600 0.327433

Up: Checking displacement for: y
rsfe winner: y displacement: 6.7288
rsaa winner: -y displacement: 8.4572
rhr NO winner: displacement: 0.0000

Can change by using shoulder flex/ext.
Use it?(yes or no)?
Coach: *yes*
Can change by using shoulder abd/add.
Use it?(yes or no)?
Coach: *yes*

Finished with right shoulder. Testing torso next...

ACKNOWLEDGMENT

This material is based upon work supported in part by the DARPA Learning Locomotion Program, the National Science Foundation under NSF Grants CNS-0224419, DGE-0333420, and ECS-0325383 and the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

REFERENCES

[1] M. Nakatani, K. Suzuki, and S. Hashimoto, "Subjective-evaluation oriented teaching scheme for a biped humanoid robot," in *Proc. IEEE-RAS Conference on Humanoid Robotics*, September/October 2003.

[2] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proc. Second Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, July 2003.

[3] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no.9, pp. 1275–1296, 2001.

[4] S. Kamohara, H. Takagi, and T. Takeda, "Control rule acquisition for an arm wrestling robot," in *IEEE International Conference on System, Man, Cybernetics*, October 1997, pp. 4227–4231 Vol.5.

[5] Y. Kuroki, B. Blank, T. Mikami, P. Mayeux, A. Miyamoto, R. Playter, K. Nagasaka, M. Raibert, M. Nagano, and J. Yamaguchi, "Motion creating system for a small biped entertainment robot," in *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems*, October 2003, pp. 1394–1399.

[6] J. Lee and S. Shin, "A hierarchical approach to interactive motion editing for human-like figures," in *Proceedings of ACM SIGGRAPH 99*, 1999, pp. 39–48.

[7] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data," in *Proceedings of ACM SIGGRAPH 2002*, 2002.

[8] M. Gleicher, "Comparative analysis of constraint-based motion editing methods," *Graphical Models*, vol. 63, no. 2, pp. 107–134, 2001.

[9] R. Schmidt and T. Lee, *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, 3rd edition, 1999.

[10] J. Kalaska, *What parameters of reaching are encoded by discharges of cortical cells?* John Wiley & Sons, 1991.

[11] N. Schweighofer, J. Spoelstra, M. Arbib, and M. Kawato, "Role of the cerebellum in reaching movements in humans: Ii. a neural model of the intermediate cerebellum," *European Journal of Neuroscience*, vol. 10, pp. 95–105, 1998.

[12] J. S. Kelso, Ed., *Human Motor Behavior: An Introduction*. Lawrence Erlbaum Associates, Inc., 1982.

[13] N. Bernstein, *The control and regulation of movements*. London: Pergamon Press, 1967.

[14] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, New York: CRC Press, 1994.

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

[16] M. Riley, A. Ude, K. Wade, and C. Atkeson, "Enabling real-time full-body imitation: A natural way of transferring human movement to humanoids," in *Proc. IEEE Int. Conf. Robotics and Automation*, Taipei, Taiwan, September 2003, pp. 2368–2374.

[17] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, pp. 93–108, 2004.

[18] J.Piaget, *Play, Dreams and Imitation in Childhood*. New York: W. W. Norton, 1945, translated 1962.

[19] C. Breazeal and B. Scassellati, "Robots that imitate humans," *TRENDS in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[20] M. Mataric, "Getting humanoids to move and imitate," *IEEE Journal of Intelligent Systems*, vol. 15, no. 4, pp. 18–24, July/August 2000.

[21] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, pp. 233–242, 1999.

[22] N. Pollard, J. K. Hodgins, M. Riley, and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," in *IEEE-RAS Conference on Robotics and Automation*, May 2002, pp. 1390–1397.

[23] C. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, and M. Kawato, "Using humanoid robots to study human behavior," *IEEE Journal of Intelligent Systems*, vol. 15, no. 4, pp. 46–56, July/August 2000.

[24] M. Riley and C. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Autonomous Robots*, vol. 12, pp. 119–128, 2002.

Experience based Learning and Control of Robotic Grasping

Johan Tegin and Jan Wikander
Mechatronics Laboratory
Machine Design
KTH, Stockholm, Sweden
E-mail: johant, jan@md.kth.se

Staffan Ekvall and Danica Kragic
Computational Vision and Active
Perception Laboratory
KTH, Stockholm, Sweden
E-mail: ekvall, danik@nada.kth.se

Boyko Iliev
Biologically Inspired Systems Lab.
Applied Autonomous Sensor Systems
Örebro University, Örebro, Sweden
E-mail: boyko.iliev@tech.oru.se

***Abstract*— In this paper a method for automatic grasp generation for robotic hands is presented. Experience and shape primitives are used in synergy and provide a basis not only for grasp generation but also for a grasp evaluation process when the exact pose of the object is not available. The problem is studied in a Programming by Demonstration scenario where the system first recognizes the human induced grasp and the object it is applied to. Based on these, a suitable grasping scheme is chosen for the robot so that it can perform a successful grasp. In this work, the entire grasp sequence is thoroughly evaluated in a simulated environment, from learning a grasp to actually reaching it, including dynamic simulation of the grasp execution with a focus on grasping objects whose pose is not perfectly known. We also discuss the necessary requirements for evaluating this approach in a real setting.**

I. Introduction

One of the main challenges in the field of robotics is to make robots ubiquitous. To intelligently interact with the world, one of the key abilities that robots need to have is to manipulate objects. Typical environments in which robots will be deployed, such as a house or an office, are dynamic and it is very difficult to equip robots with an ultimate and general grasp planning capability. Planning a grasp is difficult due to the large search space resulting from all possible hand configurations, grasp types, and object properties that occur in regular environments. Another important question is how to equip robots with capabilities of gathering and interpreting the necessary information for novel tasks through interaction with the environment in combination with minimal prior knowledge.

In relation to grasping, some recent approaches propose the use of prehensile postures where object features and experience is used to aid the selection of the pre-grasp posture and grasp scheme. Such an approach significantly decreases the size of the search space. This paper presents a method for grasp generation for robotic hands where programming by demonstration, experience and shape primitives are used to provide a successful grasp. A top-down (experience) and a bottom-up methodology are integrated to develop a more natural grasp learning system. It is important to note that the bottom-up methodology can be seen as semi-autonomous grasping control. The proposed method is shown to work for choosing the grasp approach vector, but can also be used to choose other grasp control parameters that affect the fingers' closing sequence, controller switching, reactions to tactile sensor inputs et cetera. The methods in this paper are applicable to numerous grasping related problems but the focus here is on one of the main challenges – choosing the object approach vector, which is dependent both on the object shape and pose as well as the grasp type. Using the proposed method, the approach vector is chosen not only based on perceptional cues but also on experience that some approach vectors will provide useful tactile cues that finally result in stable grasps. Moreover, a methodology for developing and evaluating grasp schemes is presented where the focus lies on obtaining stable grasps under imperfect vision.

Our longterm research is related to the design of the Programming by Demonstration systems, [1], [2], where the user teaches the robot new tasks by simply demonstrating them. The robot can first imitate human behaviour and then improve through continuous interaction with the environment. This approach borrows some ideas from the field of teleoperation, that provides a means of direct transmission of dexterity from the human operator. Most of the work in this field focuses however on low-level support such as haptic and graphical feedback and deals with problems such as time delays, [3]. For instruction systems that involve object grasping and manipulation, visual and haptic feedback are necessary. The robot has to be instructed *what* and *how* to manipulate, [4]. If the kinematics of robot arm/hand system is the same as for the human, a one-to-one mapping approach may be considered. This is, however, never the case. The problems arising are not only related to the mapping between different kinematic chains for the arm/hand systems but also to the quality of the object pose estimation delivered by the vision system. Hence, the methods presented here should be considered in a Programming by Demonstration setting where the system can recognize the human induced grasp and the object it is applied to. Based on these, a suitable grasping scheme is chosen for the robot so that it can perform a successful grasp. Our previous results related to these problems have been presented in [5], [6], [7] and [8].

In this work, the entire grasp sequence is thoroughly evaluated in a simulated environment, from learning a grasp to actually reaching it, including dynamic simulation of the grasp execution. We also discuss the necessary requirements for evaluating this approach in a real setting. It should be noted here that the problems arising are not only related to the mapping between different kinematic chains for the arm/hand systems but also to the quality of the object pose estimation delivered by the vision system.

The contributions of the work presented in this paper are:

- A suitable grasp is related to object pose and shape and not only a set of points generated along its outer contour. This means that we do not assume that the initial hand position is such that only planar grasps can be executed as proposed in [9]. In addition, grasps relying only on a set of contact points may be impossible to generate on-line since the available sensory feedback may not be able to estimate the exactly same points on the object's surface once the pose of the object is changed.
- The choice of the suitable grasp is based on the *experience*, i.e. it is learned from the human by defining the set of most likely hand preshapes with respect to the specific object. A similar idea was investigated in [10] but only one robotic hand and four grasp preshapes were considered. We evaluate both Barrett [11] and Robonaut hand, [12]. Since grasp preshapes are generated based on recognition of human grasps it makes them more natural. This is, of course, of interest for humanoid robots where the current trend is to resemble human behaviour as closely as possible.
- Finally, we evaluate the quality of different grasp types with respect to inaccuracies in pose estimation. This is an important issue that commonly occurs in robotic systems. The reasons may be that the calibration of the vision system or hand-eye system is not exact or that a detailed model of the object is not available. We evaluate how big pose estimation error different grasp types can handle.

This paper is organized as follows. In Section II we shortly review the related work and in Section III a description of the the whole system is given. In Section IV, our grasp mapping strategy is presented in more detail followed by the adopted control approach Section V. Planning and grasp quality is discussed in Section VI and the results of the conducted experimental evaluation are given in Section VII. We summarize the paper in Section VIII.

II. Related Work

Considering specifically object manipulation tasks, the work on automatic grasp synthesis and planning is of significant relevance, [10], [13], [9], [14]. The main issue here is the automatic generation of stable grasps assuming that the model of the hand is known and that certain assumptions about the shape of the object can be made. Example of assumptions may be that the full and exact pose of the object is known in combination with its (approximate) shape, [10]. Another common assumption is that the outer contour of the object can be extracted and a planar grasp applied, [9]. The work on contact-level grasps synthesis concentrates mainly on finding a fixed number of contact locations with no regard to hand geometry, [15], [16].

Taking into account both the hand kinematics as well as some *a-priori* knowledge about the feasible grasps has been acknowledged as a more flexible and natural approach towards automatic grasp planning [17], [10]. In [17], a method for adapting a given prototype grasp of one object to another object, such that the quality of the new grasp would be at least 75% of the quality of the original one was developed. It has to be, however, pointed out that this process required a parallel algorithm running on supercomputer to be computed efficiently. The method proposed in [10] presents a system for automatic grasp planning for a Barrett hand by modelling an object as a set of shape primitives, such as spheres, cylinders, cones and boxes in a combination with a set of rules to generate a set of grasp starting positions and pregrasp shapes.

With respect to dynamic grasping and manipulation control, previously presented results include catching a ball or playing the piano using the robotic DLR Hand [18]. Exchanging a light bulb has been shown using the Utah/MIT hand [19]. High speed grasping has also been demonstrated in [20]. In terms of grasping systems, relevant ideas have been presented in [21].

III. System Description

In this paper, robotic grasping sequences are performed combining a learning by demonstration framework with semi-autonomous grasping. Let us start by a short motivation for the system design. Consider a human and a robot each standing in front of a table, on which a set of objects are placed, Fig. 1. A specific task is then demonstrated to the robot. That task may be moving (*pick up/move/put down*) an object. The robot recognizes which object has been moved and where using visual feedback. The magnetic trackers on the human hand, provide information that enables the robot to recognize the grasp type used. The robot should then reproduce or imitate the action induced by the human, [5]. Recent work has also evaluated how tasks can be learnt based on multiple demonstrations, [6].

In this paper, we design and evaluate a system for automatic grasp generation and fine control, that can be used in the above scenario. The approach is evaluated in simulation using two kinematically different hands, the Barrett hand and the Robonaut hand. Using the Barrett hand as an example, a methodology for designing a grasp controller for corrective movements is outlined. In addition, it is shown how dynamic simulation can be used for building grasp experience and for the evaluation of grasp performance.

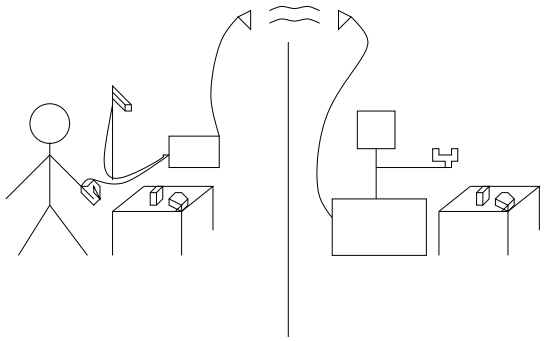


Fig. 1. Left: A human demonstrates object manipulation tasks to the robot. A camera and data glove equipped with magnetic trackers provide sensory input for task recognition. Right: The robot uses this information to reproduce the demonstrated task using its own frame of reference.

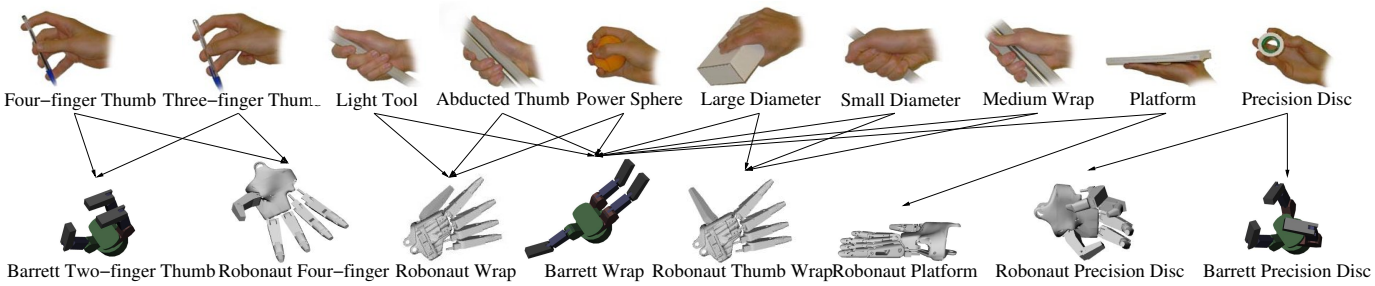


Fig. 2. Initial robot hand postures for different grasp types.

We shortly review the components currently used in our system:

1) Object Recognition and Pose Estimation
   Estimation of the objects' poses before and after an action enables the system to identify *which* object has been moved *where*. For object recognition and pose estimation, Receptive Field Co-occurrence Histograms is used [7], [22]. In this study, it is assumed that the objects are resting on a table. The pose can hence be represented by three parameters (x, y and ϕ).
2) Grasp Recognition
   A glove with magnetic trackers provides hand postures to to a grasp recognition system [8]. The position of the hand is used to segment the grasp task.
3) Grasp Mapping
   An off-line learnt grasp mapping procedure maps the human grasps to robot grasps as presented in Section IV.
4) Grasp Planning
   The robot selects a suitable grasp controller. The object will be approached from the direction that maximized the probability of reaching a successful grasp. This is presented in more detail in Section VI.
5) Grasp Execution
   A semi-autonomous grasp controller is used to control the hand from the planned approach position until a force closure grasp is reached, Section V.

The evaluation of the system proposed in this work is performed using a modified and extended version of the robot grasp simulator GraspIt! [23] to allow for repetitive experiments and statistical evaluation. We strongly believe that the results of the experimental evaluation facilitate further development of robot grasping systems.

IV. Grasp Mapping

It has been argued that grasp preshapes can be used to limit the large number of possible robot hand configurations. This is motivated by the fact that, when planning a grasp, humans unconsciously simplify the grasp choice by choosing from a limited set of prehensile postures appropriate for the object and task at hand [24]. Related to robotics, Cutkosky [25] classified human grasps needed in a manufacturing environment and evaluated how the task and object geometry affect the choice of grasp. The work on virtual fingers generalized the existing grasp taxonomies, [26]. Based on the above work and described in our previous work [8], the current grasp recognition system can recognize ten different grasp types. Due to the different kinematics between the robot and human hand, the grasp demonstrated by the human has to be first mapped to the robot. For this purpose, the mapping scheme showed in Table I was defined.

Human Grasp	Barrett Grasp	Robonaut Grasp
Large Diameter	Barrett Wrap	Robo. Thumb Wrap
Small Diameter	Barrett Wrap	Robo. Thumb Wrap
Medium Wrap	Barrett Wrap	Robo. Thumb Wrap
Abducted Thumb	Barrett Wrap	Robonaut Wrap
Light Tool	Barrett Wrap	Robonaut Wrap
Four-finger Thumb	Barrett Two-finger Thumb	Robo. Four-finger
Three-finger Thumb	Barrett Two-finger Thumb	Robo. Four-finger
Power Sphere	Barrett Wrap	Robonaut Wrap
Precision Disc	Barrett Precision Disc	Robo. Precision Disc
Platform	Barrett Wrap	Robonaut Platform

TABLE I

THE MAPPING OF HUMAN GRASPS TO ROBOT GRASP CONTROLLERS. THE LEFT COLUMN IS A SELECTION OF HUMAN GRASPS FROM CUTKOSKY'S GRASP HIERARCHY.

It has to be noted here that the robot grasp types do not refer only to hand postures, but to grasp execution schemes. Such a scheme includes the initial position, the *approach vector*, the robot hand closing sequence, controllers for corrective movements, etc. Hence, different strategies are used to grasp an object dependent on the grasp type. Fig. 2 illustrates the initial hand postures for each of the controllers.

V. Grasp Control

There are two basic grasp controllers in the system: Power Grasp and Precision Grasp. There are eight variations of these, three for the Barrett hand and five for the Robonaut hand. The difference lies in the initial grasping position and the finger control during closure. In this paper, all eight variations are evaluated. As the the dynamics of the grasping process is essential in deciding if a stable grasp was reached, the Barrett Wrap grasp was simulated using rigid body dynamics and the control scheme outlined in Section V-A.

- **Power Grasp**
  First, the initial hand posture is set according to the grasp type recognized from the human demonstrator. The hand then approaches the object until contact is detected upon which all fingers close until contact. Depending on the grasp type, the joint angle speed may be different for each joint, causing for example the thumb to close more slowly.
- **Precision Grasp**
  This controller is similar to the Power Grasp, but with an added dimension. Once a contact is detected, the hand retracts a predefined distance and then close all fingers simultaneously. This allows the robot to better combine tactile sensing with computer vision, as we previously demonstrated in [27].

The grasp approach vector is defined relative to the object's pose and center. Other object shapes may require evaluation of several approach vectors, e.g. the object top and bottom, or one or more for each object feature.

A. Control Scheme

As previously mentioned, with the goal of making robots ubiquitous, complete knowledge of the world cannot be expected. In addition, limited accuracy in computer vision or effects such that objects or the hand itself may occlude vision, requires a grasp control algorithm able to handle such uncertainty. Here we show how to derive a low level controller that is able to cope with some of these problems. The controller is designed to cope with uncertainties and corrective movements, but it does not communicate with higher level controllers. Hence, the grasp control has no support for moving the wrist or detecting object motion from vision. From start to completion of the grasp, the grasp controller is autonomous.

The grasping sequence can be seen as comprised of two phases; first closing the fingers until contact and then maintaining the contact while applying proper contact forces. It is also important to implement a contact displacement controller so that the object position after finger contact can be controlled. In other words, using position control we can also apply local corrective movements. The need for such movements can be exemplified by the Barrett hand where the grasp is often of higher quality when all fingers have approximately the same closing angle rather than when the object is far from the palm center. This behaviour can be seen in the example task shown in Fig. 4. Here, the Barrett hand is modelled as rigid bodies where the two joint angles of each finger have a fixed relation. Control is performed by applying torque joint. Hence position control requires D-control or friction modelling. This and all other control is performed in Matlab, see Fig. 3.

Before the contact, the velocity of each finger is individually controlled. The contact is then detected by deriving the acceleration from the joint encoders. While the reference values for position and force start to change, the velocity controller is smoothly switched off. A feed-forward loop compensates for gravity. Alternative controllers have been investigated in e.g. [28].

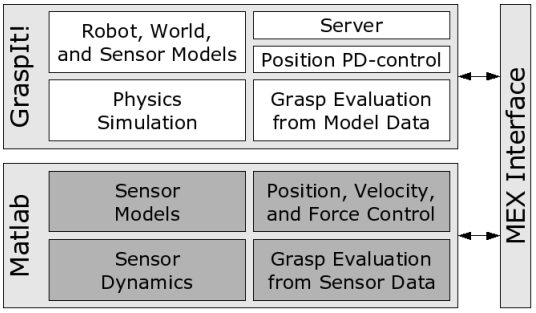


Fig. 3. Software layout for the simulation environment.

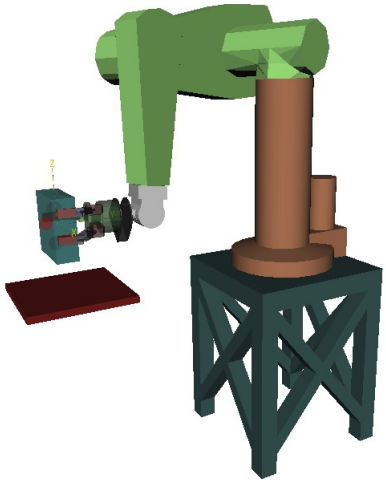


Fig. 5. The box is grasped and lifted by the Barrett hand mounted on a Puma arm.

B. Control Design

To enable a more intuitive formulation of the controller – as opposed to decentralized control of reference trajectories and torques – a control design is used that allows the controller to be specified in a more direct way, as presented in our previous work [29]. To exemplify the design process we use the Barrett hand. The angle between the two fingers on the one side, the spread, and the closure of each finger can be controlled by setting the joint torques. Accordingly, the hand has four degrees of freedom. The basis for the controller is a linear transform T relating the original joint angles q to new control variables x, see Fig. 6. The transform is

$$x = Tq. \tag{1}$$

It is approximated that joint angle corresponds to finger position. (The controller is designed as if the hand was a parallel jaw gripper.) The closing force is controlled using tactile force sensor data while joint encoder data is used to control the finger positions. For now, spread is not controlled.

To control the total grasp force, a variable is defined to control the hand closure:

$$x_2 = \frac{q_2 + q_3}{2} + q_4. \tag{2}$$

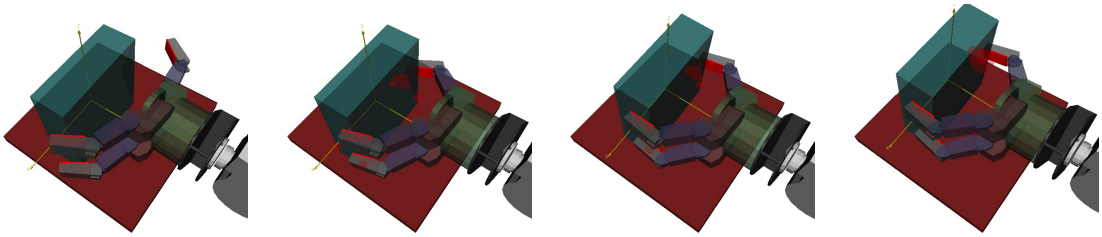


Fig. 4. Execution of a sample task where *corrective movements* are used to center the object.

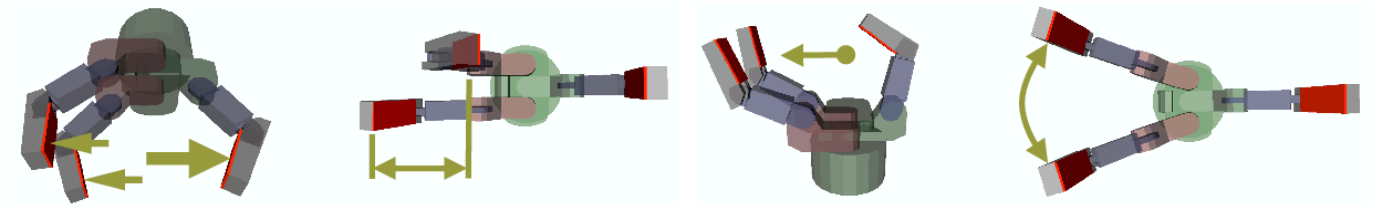


Fig. 6. Grasp controllers: total grasp force, stability, centering, and spread.

To control centering, the next variable is defined as the difference between the average joint angle of the two fingers on the one side and the single finger on the other side:

$$x_3 = \frac{q_2 + q_3}{2} - q_4. \tag{3}$$

Stability is added to the grasp by trying to keep the angles q_2 and q_3 equal. A control variable that is the difference in joint angle between the two fingers on the one side is defined:

$$x_4 = q_2 - q_3. \tag{4}$$

Controlling the force, centering and stability according to the above and Fig. 6, the transform becomes:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 1/2 & 1/2 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}. \tag{5}$$

The control forces f are computed using a P-controller $f = De$ where D contains controller gains and e is an error vector with force and position errors. The joint torques F are computed as

$$F = T^T f = T^T De. \tag{6}$$

Tactile sensors, see Section V-C are used to control the grasp force (x_2) and joint encoders to control the position (x_3) and "stability" (x_4). The error e is computed using the desired [des] and actual [act] variable values as

$$\begin{aligned} e &= \begin{bmatrix} e_1 & e_2 & e_3 & e_4 \end{bmatrix}^T \\ e_1 &= 0 \\ e_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} e_f \\ e_3 &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} e_x \\ e_4 &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} e_x \\ e_f &= f_{des} - f_{act} = f_{des} - T^{-T} F_{act} \\ e_x &= x_{des} - x_{act} = x_{des} - T q_{act}. \end{aligned} \tag{7}$$

To focus on the displacement control, we use

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & k_p & 0 & 0 \\ 0 & 0 & 5k_p & 0 \\ 0 & 0 & 0 & k_p \end{bmatrix}. \tag{8}$$

C. Tactile Sensors

Most robots are equipped with sensors that measure joint positions, but only tactile sensors are able to provide measurements at the exact point of contact. In the current system, it is assumed that three distributed extrinsic tactile sensors capable of detecting the normal force only were mounted to the distal links, see Fig. 7. This type of touch sensors are available at a low cost and are easy to mount to an existing robot hand as we have shown in our previous work [27]. Considerations on different tactile sensors are put in to perspective in [30]. More general overviews of sensors for grasping include [31], [32].

VI. Grasp Planning

The grasp planner assumes that an approximate model of each object considered for grasping is available. Since it can be difficult to automatically acquire detailed models of complex shapes, it is more reasonable to assume that it will be possible to extract *shape primitives* using computer vision

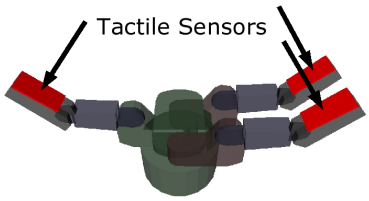


Fig. 7. The placement of the tactile sensors.

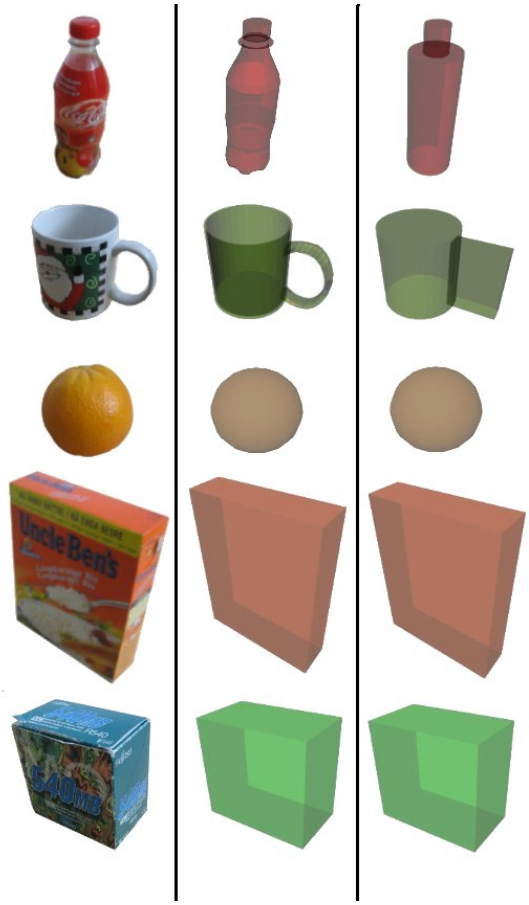


Fig. 8. Left: The real objects. Center: The modelled objects. Right: The object primitives used for training.

or laser technology. Each object can be represented by its appearance (textural properties) for visual recognition and its object shape, or shape primitives, for grasp planning. The basic shape primitives are e.g. truncated cone, sphere, box, cylinder etc. Recent progress presented in [33] shows a promising method for retrieving shape primitives using vision, although the method currently is restricted to objects with uniform color. To evaluate an object representation using primitives, primitive representations were derived, see Fig. 8.

The planning is performed using a simple search technique where many different approach vectors are tested on the object. The training can be performed on either the primitive object model or the full object model, and in the experiments we have evaluated both methods. A more detailed model will in general result in higher grasp quality on the real object.

For power grasps, three parameters (θ, ϕ, ψ) are varied describing the approach direction and hand rotation. For precision grasps, a fourth parameter d, that describes the retract distance when contact is detected, is added. The number of evaluated values for the variables are θ=9, ϕ=17, ψ=9, d=6. For the precision grasps the search space was hence 8262 grasps which required about an hour of training using kinematic simulation. For the power grasp simulations, 1377 approach vectors were evaluated. The 5 s long grasping sequence is dynamically simulated in 120 s (Intel P4, 2.5 GHz, Linux). The quality measures for each grasp is stored in a *grasp experience* database.

A. Grasp Quality Measures

To evaluate grasps, the 6-D convex hull spanned by the forces and torques that the grasp can resist is analyzed using GraspIt! [34]. The ϵ-L1 quality measure is the smallest maximum wrench the grasp can resist and is used for power grasps. For precision grasps, a grasp quality measure based on the volume of the convex hull was used, volume-L1. These grasp quality measures obviously require full knowledge of the world, and can thus only be used in simulation.

B. Grasp Retrieval

At run-time, the robot retrieves the approach vector that result in the highest quality grasp from the grasp experience database. As the highest quality grasp is not necessarily the most robust with respect to position and model errors, the grasp should be chosen taking also those parameters into account, see Section VII-B. Because of robot kinematic constraints and possible non-free paths toward the object, all approach directions are not suitable at task execution time. Thus, the robot searches the database only for directions that are applicable in the current situation. In a Programming by Demonstration scenario, the mapping from human to robot grasp is one-to-one. But if the robot acts autonomously, i.e. *explores* the environment and performs grasp on unknown objects, the grasp type is not defined and the best grasp can be chosen from among all possible grasps.

VII. Experimental Evaluation

This section provides experiments that demonstrate i) grasps performed by the robot hand given the current state of the environment and the *grasp experience* database, and ii) experiments that show how errors in pose estimation affect the success of the final grasping result.

The five objects shown in Fig. 8 were modelled and added to the GraspIt! simulator. The real objects were placed on a table, Fig. 9 (left). A camera monitors the world state which consist of five objects placed at arbitrary positions. The figure on the right shows the results of object recognition and pose estimation process - the objects are placed at the same positions in the simulator as they are in the world.

Fig. 9. Left: The human moves the rice box. The system recognizes what object has been moved and which grasp is used. Right: The robot grasps the same object using the mapped version of the recognized grasp.

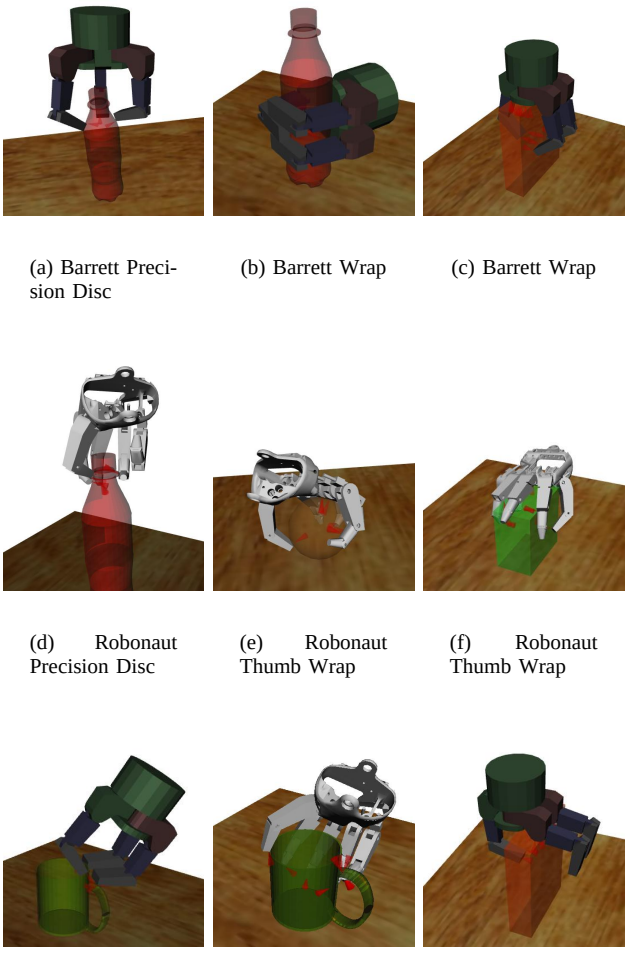
The human teacher, wearing a data-glove with magnetic trackers, moves one object. The move is recognized by the vision system and so is the grasp the teacher used. This information is used to generate a suitable robot grasp (grasp mapping) that controls the movement of the robot hand in the simulator.

A. Control

Fig. 10 shows a few examples of the best grasps obtained during kinematic simulation when the robot is free to choose any approach direction. Fig. 10 (i) shows an example of a failed grasp, due to a simulated error in pose estimation.

Grasping the rice box was dynamically simulated using the controller from Sections V-A and V-B. Of the 1377 worlds, 1035 were automatically discarded because the hand interfered with the table upon which the box is placed while approaching the object, or that the object was obviously out of reach. The remaining 342 initial robot hand positions were evaluated and resulted in 171 force closure grasps, 170 failed grasp attempts, and one simulation error. The top three hand initial positions and the resulting grasps are shown in Fig. 11.

Some sample data from the third best simulation, Fig. 11 c) and f), is shown in Fig. 12. The desired grasping force is set to 5 N. A low-pass filter is used for the tactile sensor signal.



(a) Barrett Precision Disc (b) Barrett Wrap (c) Barrett Wrap

(d) Robonaut Precision Disc (e) Robonaut Thumb Wrap (f) Robonaut Thumb Wrap

(g) Barrett 2-finger Thumb (h) Robonaut 4-finger Thumb (i) Failed Barrett Wrap

Fig. 10. Examples of grasp executions for various grasp types and objects. (a)-(h) shows successful grasps, while (i) shows a failed grasp due to a simulated error in pose estimation. The contact friction cones are plotted in red.

B. Introducing Error in Pose Estimation

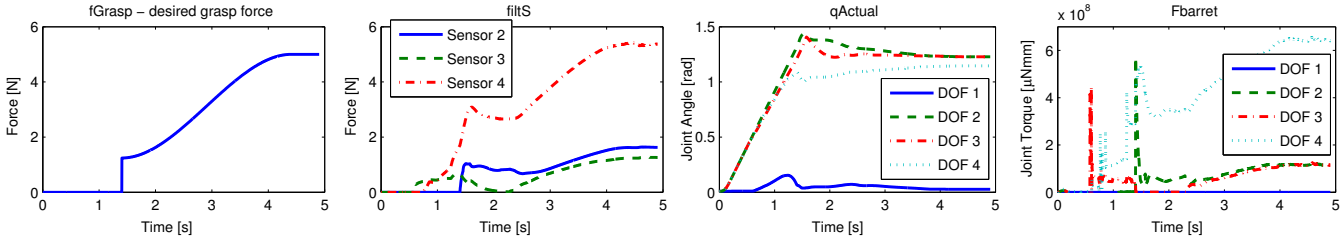
To evaluate the performance under imperfect pose estimation, we have simulated errors in pose estimation by providing an object pose with an offset. As pointed out in [35], the robustness of a grasp to positioning the end-effector has not been widely addressed in the literature.

In the experiment, the target object was placed on the table and the robot performed 50 grasps using different approaches. The robot hand position was between each grasped translated a certain distance in a random direction. As a result, the robot interpreted the situation as if the object (and possibly table) was in another position than that for which the grasp was planned. This was repeated for five different vector lengths: 0, 1, 2, 3, and 4 cm. In total, the robot grasped the object 250 times from a total of 201 different positions.

Fig. 13 - 17 show the grasp success rates for various grasps and objects, under increasing error in position estimation. The hand is moved along the approach vector until contact and the grasp scheme is initialized. A grasp is considered successful if it results in force-closure. As expected, power grasps are more robust to position errors than precision grasps. The precision grasps target details of an object, e.g., the bottle cap or the ear of the mug. Thus, the grasps are much more sensitive to position inaccuracies. It is interesting to see that the dynamic simulation and the controller previously outlined yields significantly better results than that from purely kinematic simulation. This is a motivation for continuing the investigations on the dynamics of the grasp formation process.

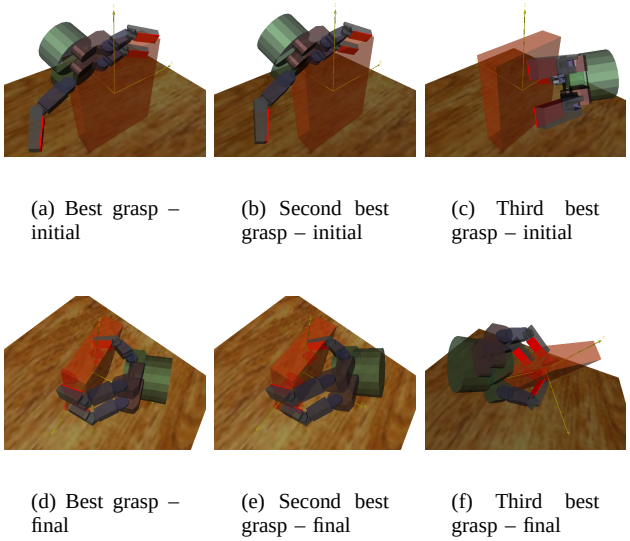
It is clear that the Barrett hand is more robust than the Robonaut hand, likely due to its long fingers. The exception is the grasping of the mug, Fig. 14, where the Robonaut Four-finger Thumb grasp is the best.

The bottle and the mug have been trained both using a



(a) Desired grasp force (b) Tactile sensors - filtered (c) Joint angles (d) Joint torques

Fig. 12. Data logged from the grasp simulation in Fig. 11 c) and f). The first 1.3 seconds the fingers close under force control. The force at that time is used as the start value for the force controller that ramps the grasp force to 5 N. The joint angle values show that the joint angles are getting closer to equal as time goes by. The controller output shows some undesirable peaks induced by collisions between the fingers and the object.



(a) Best grasp – initial (b) Second best grasp – initial (c) Third best grasp – initial

(d) Best grasp – final (e) Second best grasp – final (f) Third best grasp – final

Fig. 11. The top three approach positions and the final grasps for the rice box. These results show that it is important to consider the dynamics when designing grasp execution schemes and for analyzing the grasp formation process. In several simulations the fingers stop after contacting the box as they should, but when the grasping force is increased, the box slides on the low friction proximal links until it comes in contact with the high friction tactile sensors.

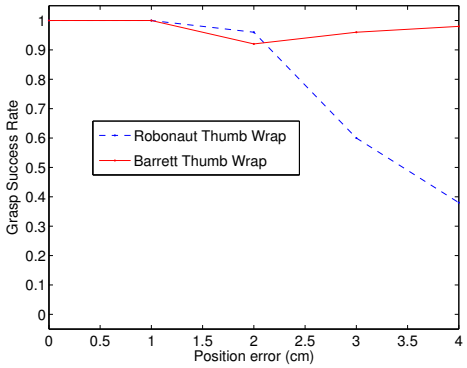


Fig. 15. Grasping the orange.

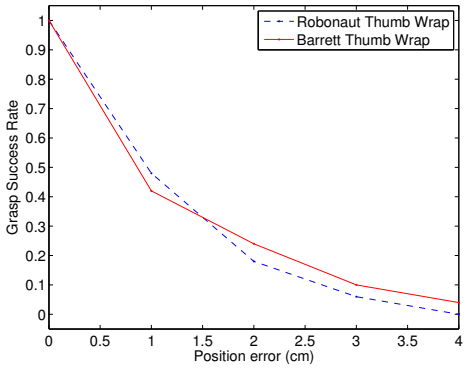


Fig. 16. Grasping the zip disc box.

primitive model and using the real model (see Fig. 8). Training on the primitive model does not decrease the grasp success rate much, especially not for the bottle. However, the primitive model of the mug is, unlike the real mug, not hollow, which causes problems for some of the precision grasps trained on the primitive.

We have also evaluated how an error in rotation estimate affects the result. For each object and grasp type, we tested how much the object could be rotated before the grasp failed. As expected, for symmetric objects like the orange and the bottle this type of error has no effect. However, for the other objects we found that the difference in rotation error tolerance is large. Table II shows the rotation tolerance for various objects and grasp types. For two of the Robonaut grasps on the mug, the rotation is not a problem, with a perfect success rate. For one of the Barrett grasps on the mug, the rotation estimation is absolutely crucial and cannot withstand a small rotation inaccuracy. Thus, this type of grasp should be avoided for this object.

C. Discussion

The success rate of the presented system depends on the performance of four subparts: i) object recognition, ii) grasp recognition, iii) pose estimation of the grasped object, and

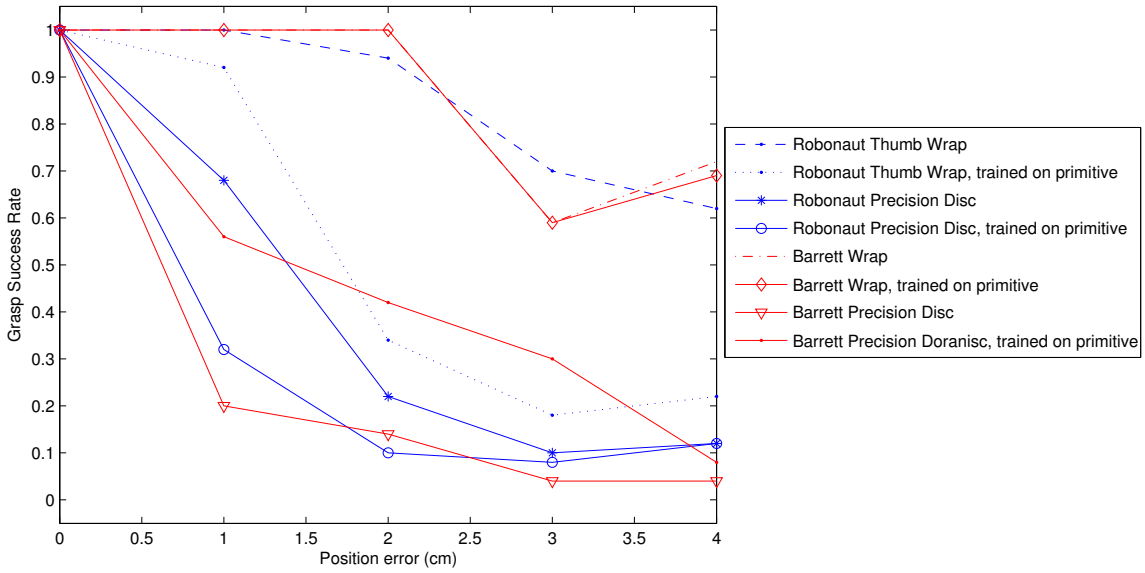


Fig. 13. Grasping the bottle.

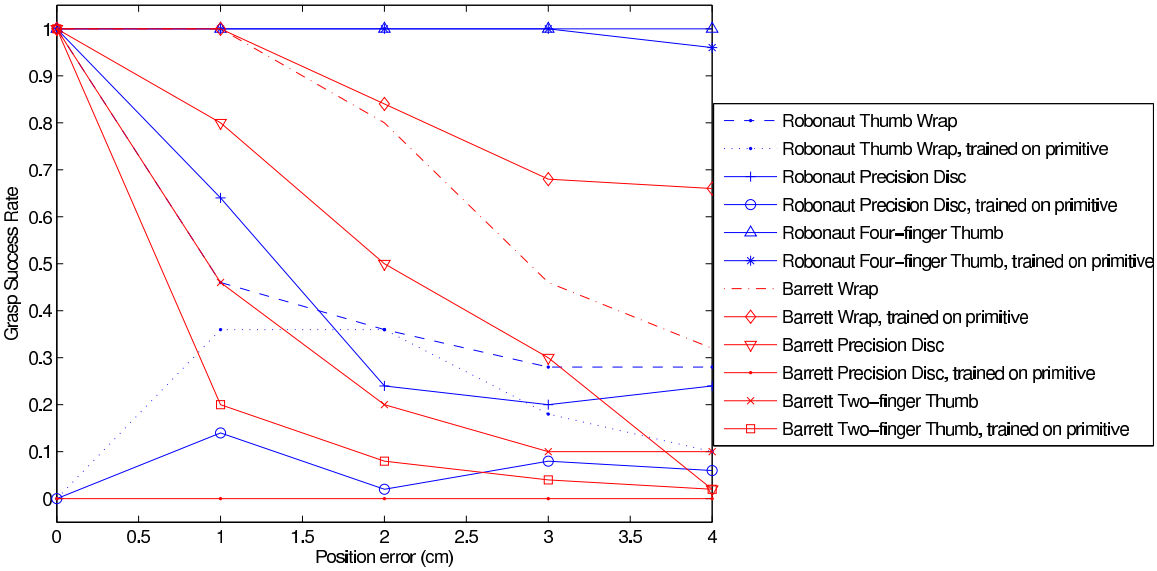


Fig. 14. Grasping the mug.

iv) grasp execution. As demonstrated in previous papers, [7], [8], the object recognition rate for only five objects is around 100 %, and the grasp recognition ratio is about 96 % for ten grasp types. Therefore, the performance in a static environment may be considered close to perfect with respect to the first steps. As the object pose and possibly the object model is not perfectly known, some errors were introduced that indicate the needed precision in the pose estimation given a certain grasp execution scheme. Initial results suggest that for certain tasks stable grasping is possible even when the object's position is not perfectly known.

If a high quality dynamic physical modelling is essential, for example when grasping compliant objects or for advanced contact models, other simulation tools may be more suitable, see e.g. [36]. But since grasping often can be performed rather slowly, and as model errors for mass properties, sensors, friction, and in actuator and gear models are often quite large, second order dynamic effects can be ignored in the control design and instead considered as small disturbances [37].

VIII. Conclusions

Methods for generating robot grasps based on object models, shape primitives and/or human demonstration have been presented and evaluated. While many factors are important, the focus lies on one of the main challenges in automatic grasping;

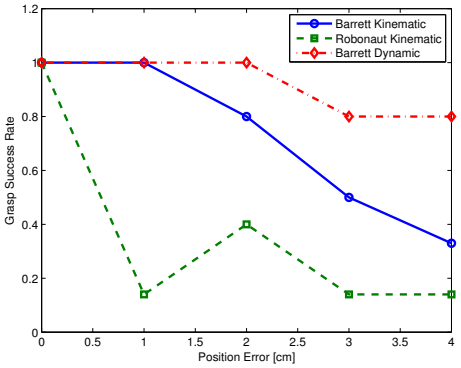


Fig. 17. Grasp success rates for different controllers and simulations. The dynamic grasp is the one from Fig. 11 c) and f). (Due to some problems with the simulator, a limited number of samples were used in the evaluation of dynamic grasping. For the 0, 1, 2, 3, and 4 cm random displacement, the number of trials were 50, 14, 18, 18, and 12 respectively (instead of 50). Still, these samples were truly random and we believe that the number of trials is high enough to draw conclusions.)

Object and Grasp Type:	Rot. Err. Tolerance (degrees):
Mug, Robonaut Precision Disc	4
Mug, Robonaut Thumb Wrap	180
Mug, Robonaut Four Finger Thumb	180
Zip Disc Box, Robonaut Thumb Wrap	17
Rice Box, Robonaut Thumb Wrap	2
Zip Disc Box, Barrett Wrap	3
Rice Box, Barrett Wrap	17
Mug, Barrett Wrap	12
Mug, Barrett Precision Disc	0
Mug, Barrett Two Finger Thumb	6

TABLE II

THE ROTATION ERROR TOLERANCE FOR DIFFERENT OBJECTS AND GRASP TYPES.

the choice of approach vector which depend on the object as well as on the grasp type. Using the proposed methods, the approach vector is chosen not only based on perceptional cues, but on experience that some approach vectors will provide useful tactile cues that result in stable grasps. Moreover, a methodology for developing and evaluating grasping schemes has been presented. Focus lies on obtaining stable grasps under imperfect vision, something that has not been thoroughly investigated in the literature.

Simulating results was necessary for generating insight into the problem and for performing the statistical evaluation for the grasp experience, since i) the world must be reset after each grasp attempt, and ii) computing the grasp quality measure requires perfect world knowledge.

The proposed strategies have been demonstrated in combination with tactile feedback and hybrid force/position control of a robot hand. The functionality of the proposed framework for grasp scheme design has been shown by successfully reaching force closure grasps using a Barrett hand and dynamic simulation.

Future work include further grasp execution scheme development and implementation. Furthermore, to ensure truly secure grasping outside the simulator, the grasping scheme must also comprise a grasp quality evaluation method that does not use information available in simulation only. Preferably such a measure would also depend upon the task at hand.

The grasp experience database contains not only a record of success rates for different grasp controllers but also the object-hand relations during an experiment. In this way, we can specify under what conditions the learnt grasp strategy can be reproduced in new trials.

The results of the experimental evaluation performed in a simulated environment suggest that the outlined approach and tools can be of great use in robotic grasping, from learning by demonstration to robust object manipulation.

ACKNOWLEDGEMENT

This work was supported by EU through the project PACO-PLUS, FP6-2004-IST-4-27657, by the Knowledge Foundation through AASS at Örebro University, and by the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems at KTH.

REFERENCES

[1] H. Friedrich, R. Dillmann, and O. Rogalla, "Interactive robot programming based on human demonstration and advice," in *Christensen et al (eds.):Sensor Based Intelligent Robots, LNAI1724*, pp. 96–119, 1999.

[2] J. Aleotti, S. Caselli, and M. Reggiani, "Leveraging on a virtual environment for robot programming by demonstration," in *Robotics and Autonomous Systems, Special issue: Robot Learning from Demonstration*, vol. 47, pp. 153–161, 2004.

[3] M. Massimino and T. Sheridan, "Variable force and visual feedback effects on teleoperator man/machine performance," in *Proc. of NASA Conference on Space Telerobotics*, 1989.

[4] 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.

[5] S. Ekvall and D. Kragic, "Integrating object and grasp recognition for dynamic scene interpretation," in *IEEE International Conference on Advanced Robotics, ICAR'05*, 2005.

[6] S. Ekvall and D. Kragic, "Learning task models from multiple human demonstration," in *IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, 2006.

[7] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *IEEE/RSJ IROS*, 2005.

[8] S. Ekvall and D. Kragic, "Grasp recognition for programming by demonstration," in *IEEE/RSJ IROS*, 2005.

[9] A. Morales, E. Chinellato, A. H. Fagg, and A. del Pobil, "Using experience for assessing grasp reliability," *International Journal of Humanoid Robotics*, vol. 1, no. 4, pp. 671–691, 2004.

[10] A. T. Miller, S. Knoop, and H. I. C. P.K. Allen, "Automatic grasp planning using shape primitives," in *In Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1824–1829, 2003.

[11] W. Townsend, "The barretthand grasper – programmably flexible part handling and assembly," *Industrial Robot: An International Journal*, vol. 27, no. 3, pp. 181–188, 2000.

[12] C. Lovchik and M. Diftler, "The Robonaut hand: a dexterous robot hand for space," in *Robotics and Automation, IEEE International Conference on*, vol. 2, pp. 907–912, 1999.

[13] N. S. Pollard, "Closure and quality equivalence for efficient synthesis of grasps from examples," *International Journal of Robotic Research*, vol. 23, no. 6, pp. 595–613, 2004.

[14] R. Platt Jr, A. H. Fagg, and R. A. Grupen, "Extending fingertip grasping to whole body grasping," in *Proc. of the Intl. Conference on Robotics and Automation*, 2003.

[15] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'00*, pp. 348–353, 2000.

[16] D. Ding, Y.-H. Liu, and S. Wang, "Computing 3-d optimal formclosure grasps," in *In Proc. of the 2000 IEEE International Conference on Robotics and Automation*, pp. 3573 – 3578, 2000.

[17] N. S. Pollard, "Parallel methods for synthesizing whole-hand grasps from generalized prototypes," 1994.

[18] C. Borst, M. Fischer, S. Haidacher, H. Liu, and G. Hirzinger, "DLR hand II: Experiments and experiences with an antropomorphic hand," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 702–707, Sept. 2003.

[19] M. Jägersand, *On-line Estimation of Visual-Motor Models for Robot Control and Visual Simulation*. PhD thesis, Univ. of Rochester, 1997.

[20] A. Namiki, Y. Imai, M. Ishikawa, and M. Kaneko, "Development of a high-speed multifingered hand system and its application to catching," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 2666–2671, Oct. 2003.

[21] I. Horswill, *Behavior-Based Robotics, Behavior Design*. Technical report CS 395, Northwestern University, 2000.

[22] S. Ekvall, D.Kragic, and F. Hoffmann, "Object recognition and pose estimation using color cooccurrence histograms and geometric modeling," *Image and Vision Computing*, vol. 23, pp. 943–955, 2005.

[23] A. T. Miller and P. Allen, "Graspit!: A versatile simulator for grasping analysis," in *Proceedings of the of the 2000 ASME International Mechanical Engineering Congress and Exposition*, 2000.

[24] J. Napier, "The prehensile movements of the human hand," in *Journal of Bone and Joint Surgery*, vol. 38B(4), pp. 902–913, 1956.

[25] M. Cutkosky, "On grasp choice, grasp models and the desing of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[26] T. Iberall, "Human prehension and dextrous robot hands," *The Int. J. of Robotics Research*, vol. 16, no. 3, 1997.

[27] D. Kragic, S. Crinier, D. Brunn, and H. I. Christensen, "Vision and tactile sensing for real world tasks," *Proceedings IEEE International Conference on Robotics and Automation, ICRA'03*, vol. 1, pp. 1545–1550, September 2003.

[28] R. Volpe and P. Khosla, "A theorethical and experimental investigation of explicit force control strategies for manipulators," *IEEE Trans. Automatic Control*, vol. 38, pp. 1634–1650, Nov. 1993.

[29] J. Tegin and J. Wikander, "A framework for grasp simulation and control in domestic environments," in *IFAC-Symp. on Mechatronic Syst.*, (Heidelberg, Germany), Sept. 2006.

[30] R. D. Howe, "Tactile sensing and control of robotic manipulation," *Advanced Robotics*, vol. 8, no. 3, pp. 245–261, 1994.

[31] M. H. Lee and H. R. Nicholls, "Tactile sensing for mechatronics - a state of the art survey," *Mechatronics*, vol. 9, pp. 1–31, Oct. 1999.

[32] J. Tegin and J. Wikander, "Tactile sensing in intelligent robotic manipulation – a review," *Industrial Robot*, vol. 32, no. 1, pp. 64–70, 2004.

[33] F. Bley, V. Schmirgel, and K. Kraiss, "Mobile manipulation based on generic object knowledge," in *IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, 2006.

[34] A. Miller and P. Allen, "Examples of 3D grasp quality computations," in *Proceedings of the of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1240–1246, 1999.

[35] A. Morales, P. J. Sanz, A. P. del Pobil, and A. H. Fagg, "Vision-based three-finger grasp synthesis constrained by hand geometry," *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 494–512, 2006.

[36] G. Ferretti, G. Magnani, P. Rocco, and L. Viganò, "Modelling and simulation of a gripper with dymola," *Mathematical and Computer Modelling of Dynamical Systems*, 2006.

[37] D. Prattichizzo and A. Bicchi, "Dynamic analysis of mobility and graspability of general manipulation systems," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 2, pp. 241–257, 1998.

Integrated Grasp Planning and Visual Object Localization For a Humanoid Robot with Five-Fingered Hands

Antonio Morales
Department of Computer Science and Engineering
University Jaume I
Campus Riu Sec, E-12071 Castellón, Spain
morales@uji.es

Tamim Asfour, Pedram Azad,
Steffen Knoop and Rüdiger Dillmann
Institute of Computer Science and Engineering
University of Karlsruhe
Haid-und-Neu-Straße 7, 76131 Karlsruhe, Germany
{*asfour,azad,knoop,dillmann*}*@ira.uka.de*

***Abstract*— In this paper we present a framework for grasp planning with a humanoid robot arm and a five-fingered hand. The aim is to provide the humanoid robot with the ability of grasping objects that appear in a kitchen environment. Our approach is based on the use of an object model database that contains the description of all the objects that can appear in the robot workspace. This database is completed with two modules that make use of this object representation: An exhaustive offline grasp analysis system and a real-time stereo vision system. The offline grasp analysis system determines the best grasp for the objects by employing a simulation system, together with CAD models of the objects and the five-fingered hand. The results of this analysis are added to the object database using a description suited to the requirements of the grasp execution modules. A stereo camera system is used for a real-time object localization using a combination of appearance-based and model-based methods. The different components are integrated in a controller architecture to achieve manipulation task goals for the humanoid robot.**

I. Introduction

The attention of the robotics community has been drawn more and more to humanoid robots in the last years. Their design, building and applications addresses many interesting research challenges: biped walking, human-robot interaction, autonomy, interaction with unstructured and unknown environments, and many others. Among them, the development of manipulation skills is of utmost importance and one of the most complex.

One of the main challenges that humanoid developers have to face when considering manipulation issues is the design of robot hands and arms. In the case of hands for humanoids their design is guided by the need of a great versatility, which means a large number of fingers and degrees of freedom, the reduced size and the human-like appearance. A constant issue has been to design human-size light arm/hand systems either focusing on a pure mechanical approach [1] or taking some anthropomorphic and biological inspiration [2]. A recent work, *Domo*, has focused on the design of compliant and reliable humanoid arms able to run for days in a secure way for humans [3]. The limited size of robot hands complicates the dispositions of the joint actuators. The solution usually comes from the use of novel actuation systems, pneumatic or fluidic [4] or cable driven [5].

In order to deal with manipulation tasks in human-centered environments, an intensive use of sensor information, particularly visual and tactile, within closed control loops is indispensable. Visual information has been used mainly to identify and apprehend the pose and shapes of objects [6], [7]. Especially relevant for dexterous manipulation tactile information has been used to reach stable grasps through finger gating or for controlling whole body grasping [8]. Several control architectures were proposed for manipulation tasks. Their main goals are to coordinate a set of behaviors implied by manipulation [9], to introduce learning in the sensor motor coordination [10], and to get inspiration from biology findings [2].

The work presented in this paper is part of long term German Humanoid Project, which goal is to develop a robot aimed to assist humans in tasks of everyday life [11]. To reach these goals, many complex abilities and characteristics are included: Humanoid shape, multimodality and the ability to cooperate with humans and learn. In the aspect of manipulation it includes the ability to learn from demonstration and to use high level cognitive models of objects and tasks.

In this paper we present an integrated approach for grasp planning. The central idea of this system is the existence of a database with the models of all the objects present in the robot workspace. From this central fact we develop two necessary modules: a visual system able to locate and recognize the objects (Sec. III), and an offline grasp analyzer that provides the most feasible grasps configuration for each object (Sec. IV). The results provided by these modules are stored and used by the control system of the humanoid to decide and execute the grasp of a particular object. We emphasize that this paper describes a first step towards a complete humanoid grasping system. At this stage the use of object and hand models allows the fast development and test of multiple interactive manipulation skills. In the long-term it is desirable, and is our purpose to develop grasping and manipulation strategies able to deal with unmodelled and unknown objects.

II. System overview

Since the robot has to work in an environment mostly designed for humans, the approach of the whole project has been to build a anthropomorphic arm/hand system that allows to imitate the way humans perform these activities. ARMAR, our humanoid robot, has 23 mechanical degrees-of-freedom (DOF). From the kinematics control point of view, the robot consists of five subsystems: Head, left arm, right arm, torso and a mobile platform [12]. The head has 2 DOFs arranged as

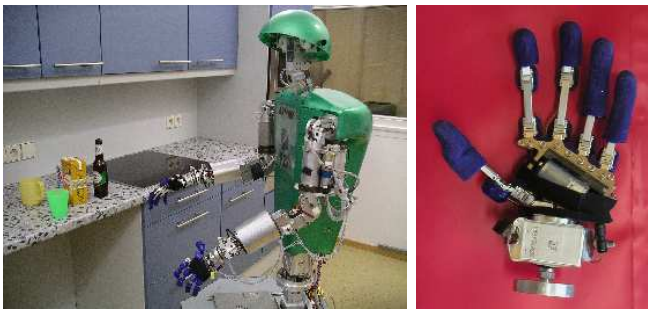


Fig. 1. The humanoid robot ARMAR with the 5-finger hands

pan and tilt and is equipped with a stereo camera system and a stereo microphone system. Each of the arms has 7 DOFs and is equipped with 6 DOFs force torque sensors on the wrist. Each hand has five fingers and 11 DOFs (3 for the thumb and 2 for the other four fingers) driven by fluidic actuators [4].

A functional description of the grasp planning system described in this paper is depicted in figure 2. It consists of the next parts:

- The **global model database**. It is the core of our approach. It contains not only the CAD models of all the objects, but also stores a set of feasible grasps for each object. Moreover, this database is the interface between the different modules of the system.
- The offline **grasp analyzer** that uses the model of the objects and of the hand to compute on a simulation environment a set of stable grasps (see Sec. III). The results produced by this analysis are stored in the grasps database to be used by the other modules.
- A **online visual procedure to identify objects in stereo images** by matching the features of a pair of images with the 3D prebuilt models of such objects. After recognizing the target object it determines its location and pose. This information is necessary to reach the object. This module is described in detail in section IV.
- Once an object has been localized in the work-scene, a grasp for that object is then selected from the set of precomputed stable grasps. This is instanced to a particular arm/hand configuration that takes into account the particular pose and reachability conditions of the object. This results in an approaching position and orientation. A path planner reaches that specified grasp location and orientation. Finally, the grasp is executed. These modules are not described in this paper since they are still under development.

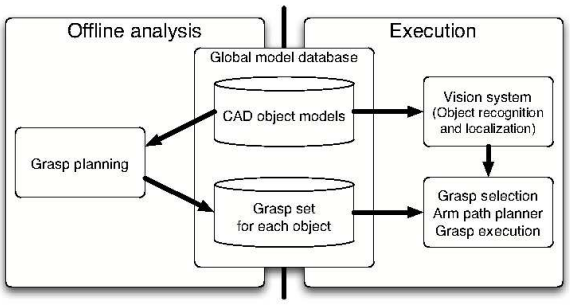


Fig. 2. Overview of the system.

III. Offline grasp analysis

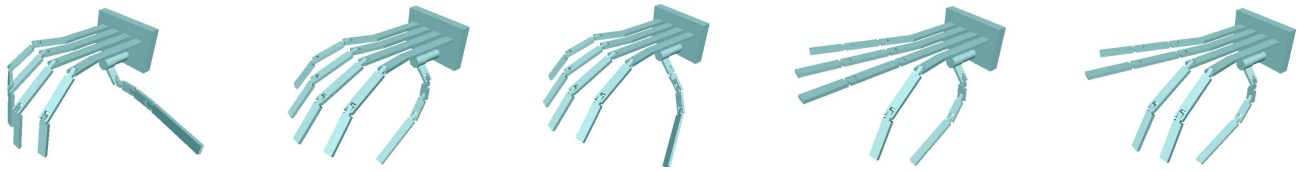
In most of the works devoted to grasp synthesis, grasps are described as sets of contact points on the object surface where forces/torques are exerted. However, this representation of grasps presents several disadvantages when considering their execution in human-centered environments. These problems arise from the inaccuracy and uncertainty about the information of the object. Since we have models of the shapes of the objects this uncertainty comes mainly from the location of the object and inaccuracy in the positions of the mobile humanoid. Usually, the contact-based grasp description requires the system be able to reach precisely the contact points and exert precise forces.

It is possible to include inaccuracy in the force/torque models, but this paper faces this problem from a different approach. In our approach grasps are described in a qualitative and knowledge-based fashion. Given an object, a grasp of that object will be described by the following features (see Fig. 4):

- **Grasp type:** A qualitative description of the grasp to be performed (see Fig. 3). The type of the grasp has practical consequences since it determines the grasp execution control, i.e.: the hand preshape posture, the control strategy of the hand, which fingers are used in the grasp, the way the hand approaches the objects and how the contact information of the tactile sensors is interpreted.
- **Grasp starting point (GSP):** For approaching the object, the hand is positioned at a distant point near it.
- **Approaching direction:** Once the hand is positioned in the GSP it approaches the object following this direction. The **approaching line** is defined by the GSP and the approaching direction.
- **Hand orientation:** the hand can rotate around the approaching direction. The rotation angle is a relevant parameter to define grasp configuration.

It is important to note that all directions are given with respect to an object centered coordinate system. The real approach directions result from matching of this relative description with the localized object pose in the workspace of the robot.

A main advantage of this grasp representation is its practical application. A grasp can be easily executed from the informa-



(a) Hook (b) Cylindrical (c) Spherical (d) Pinch (e) Tripod

Fig. 3. Hand preshapes for the five types of grasp.

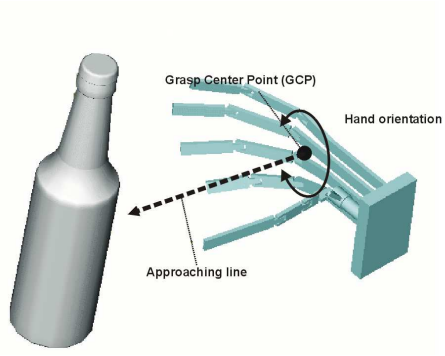


Fig. 4. Schematics with the grasp descriptors

tion contained in its description, and is better suited for the use with execution modules like arm path planning. Moreover this representation is more robust to inaccuracies since it only describes starting conditions and not final conditions like a description based in contacts points.

It is important to notice too, that this approach involves the existence of an execution module able to reach a stable grasp from the given initial conditions. This module will require the uses of sensor information, tactile or visual, to complete the grip. This module is out of the scope of this paper.

A. Grasp types

Cutkosky describes 16 different grasping hand postures for human hands [13]. The taxonomy described by Cutkosky is very complete and present many grasps that could be hardly executed by our anthropomorphic five-fingered hand attending to its mechanical limitations. Hence, we have made a selection of the most representative grasps that can be executed with our hand. These are three power grasps: hook, cylindrical and spherical and prismatic; and two precision grasps: pinch and tripod (see Fig 3). For this selection we have considered that the only finger with abduction/adduction mobility is the thumb, being thus the only one able to change its opposition with respect to the other fingers.

- **Hook grasp:** In this grasp the hand opposes the gravity. All fingers, but the thumb, form a hook that would enclose a cylindrical shaped object. The palm might exert force opposing the fingers. The thumb does not participate in any case.
- **Cylindrical grasp:** All fingers close around a cylindrical object. The thumb opposes completely the other four fingers.
- **Spherical grasp:** All fingers close around a ball-shaped object. The thumb is disposed in a way that it maximizes the area covered by the fingers.
- **Pinch grasp:** The grasp is characterized by the opposition of the thumb and index finger tips. The rest of the fingers do not participate. This is appropriate to grasp thin objects.
- **Tripod grasp:** In this case the grasp is conformed by the opposition of the thumb fingertip against the index and middle finger tips. This grasp is useful to grasp small objects.

Precision grasps only imply contacts on the finger tips, while power grasps use contacts on the whole hand surface, finger tips, phalanxes, and palm. This difference is relevant for the design of the execution controller. Roughly, for the execution of a power grasp the hand approaches the object until it makes contact, and then closes the fingers. However, in the case of precision grasps, the fingers have to close at a certain distance so that only the finger tips make contact with the object.

An important aspect when considering an anthropomorphic hand is how to relate the hand with respect the grasp starting point (GSP) and the approaching direction. For this we define for the hand the grasp center point (GCP). It is a virtual point that has to be defined for every hand and that is used as reference for the execution of a given grasp. Figure 4 depicts the parameterization of a grasp. The GCP is aligned with the GSP of the grasp. Then the hand is oriented and preshaped according to the descriptors of the grasp. Finally, it moves along the approaching line.

B. Methodology of the analysis

An important characteristic in our system is that there exists a 3D CAD model for every objects that appears in the workspace. This allows for extensive offline analysis of the different possibilities to grasp an object, instead of focusing on fast online approaches. To accomplish this we have also built a computer 3D model of the hand.

We perform an extensive analysis for each object that consists of testing a wide variety of hand preshapes and approach directions. This analysis is carried out on a simulation environment where every tested grasp is evaluated according to a quality criterion. The resulting best grasps for each object are stored in order to be used during online execution of the robot.

We use GraspIt! [14] as grasping simulation environment It has some very convenient properties for our purposes such as the inclusion of contact models and collision detection algorithms, and the ability to import, use and define object and robot models.

Our approach to compute stable grasps on 3D objects is inspired by a previous work by Miller et al. using GraspIt! [15]. The offline analysis follows four steps to find the grasps for a given object:

1) The shape of the object model is approximated by a set of basic shape primitives (boxes, cylinders, spheres and cones). There are many ways to obtain these primitive approach. GraspIt! doesn't provide any procedure to produce them. We assume that the primitive description of the objects is part of the model of an object.
2) A set of candidate grasps is generated automatically for every primitive shape of the object description. A grasp candidate consists of a hand type, a grasp starting point, an approach direction and a hand orientation. For every primitive there exists a set of predefined grasp types and approaching directions [15].
3) Each grasp candidate is tested within the simulation environment. The hand is placed in the grasp starting point and oriented according to the approaching direction and hand orientation. Then, the hand is preshaped depending on the grasp type.
   The approach phase is different for power and precision grasps. For power grasps, the hand moves opened along the approach direction until it touches the object. Then, it closes and the quality of the grasp is evaluated. If the quality is under certain threshold then the hand opens, backs a step amount and closes again. This sequence is repeated until a maximum stability measurement is reached.
   However, in the case of precision grasps, a different test is designed: 1) the hand is preshaped at the grasp starting point , 2) it closes and the grasp is evaluated if there exist a contact with object 3) it opens again and moves a step forward, 4) steps 2 and 3 are repeated until it reaches a maximum stability or a maximum number of steps is reached. Following this procedure we ensure that the first contacts with the object are made with the fingertips.
   The final position of the hand and the quality obtained is stored.
4) Finally, all final grasps that are over the minimum threshold are sorted and stored.

Part of this procedure is available in the source code of GraspIt! [14]. However it is designed exclusively for the Barrett Hand [16]. We have redesigned it to adapt it to our hand model.

As a metric for evaluating the quality of a grasp we use the magnitude of the largest worst-case disturbance wrench that can be resisted by a grasp of unit-strength. This metric is described in detail by Ferrari and Canny [17].

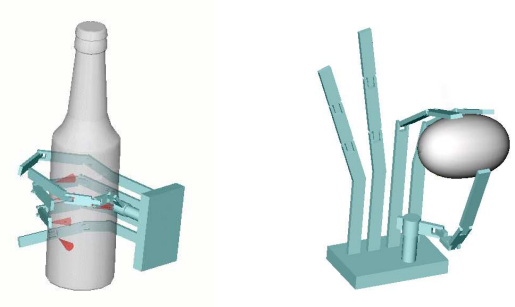


Fig. 5. Two examples of grasps produces by the grasp planning

Finally two examples of the grasps obtained for a beer bottle and an egg are shown in Fig. 5.

C. Grasp database

All stable grasps computed for every object are stored in a database in order to be used by execution modules. Every grasp stored includes the grasp type, the grasp starting point, hand orientation, approaching direction and the quality measure obtained from the simulation. This value is used by the other modules to select the best grasp for a given object.

IV. Object recognition and localization

In general, any component of a vision system in a humanoid robot for application in a realistic scenario has to fulfill a minimum number of requirements. In the particular context of the grasping system presented in this paper, the main requirements are these.

1) The component has to deal with a potentially moving robot and robot head: The difficulty caused by this is that the problem of segmenting objects can not be solved by simple background subtraction. The robot has to be able to recognize and localize objects in an arbitrary scene when approaching the scene in an arbitrary way.
2) Recognition of objects has to be invariant to 3D rotation and translation: It must not matter in which rotation and translation the objects are placed in the scene.
3) Objects have to be localized in 6D (location + orientation) with respect to a 3D rigid model in the world coordinate system: It is not sufficient to fit the object model to the image, but it is crucial that the calculated 3D pose is sufficiently accurate in the world coordinate system. In particular, the assumption that depth can be recovered from scaling with sufficient accuracy in practice is questionable.
4) Computations have to be performed in real-time: For realistic application, the analysis of a scene and accurate localization of the objects of interest in this scene should take place at frame rate in the optimal case, and should not take more than one second.

A. The Limits of State-Of-The-Art Model-Based Systems

Most model-based object tracking algorithms are based on relatively simple CAD wire models of objects, as the example

llustrated in Figure 6. Using such models, the starting and end points of lines can be projected very efficiently into the image plane, allowing real-time tracking of objects with relatively low computational effort. However, the limits of such systems are clearly the shapes they can deal with. Most real-word objects, such as cups, plates and bottles, can not be represented in this manner. The crux becomes clear when taking a look at an object with a complex shape, as it is the case for the can illustrated in Figure 7.

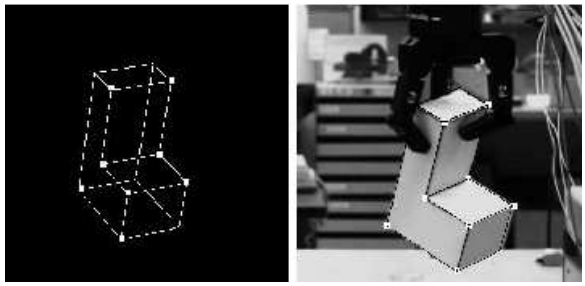


Fig. 6. Illustration of an object modeled by a wire model from [6]

The only practical way to represent such an object as a 3D model is to approximate its shape by a relatively high number of polygons. To calculate the projection of such a model into the image plane practically the same computations a rendering engine would do, have to performed. But not only the significantly higher computational cost makes common model-based approaches not feasible, also from a conceptual point of view the algorithms can not be extended for complex shapes: Objects which can be represented by straight lines and even planes have the property that each edge of the object is represented by a straight line in the model, which are then used for matching. As soon as an object also has curved surfaces this is not the case anymore: the edges of the polygons do not correspond to potentially visible edges. In [18], we show that a purely model-based approach for arbitrary 3D object models would take more than five minutes for the analysis of one potential region, having a database of three objects.

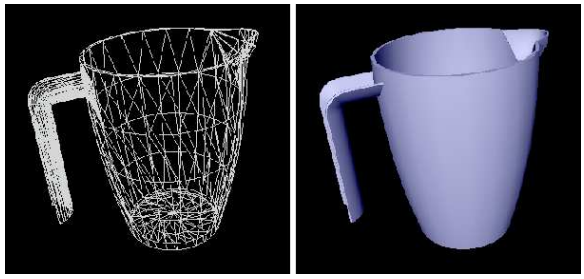


Fig. 7. Illustration of a 3D model of a can

B. Our Approach

Our approach combines the benefits of model-based and *global* appearance-bases methods [19] for object recognition and localization. Recently, *local* appearance-based methods using texture features have become very popular [20]–[23]. However, these methods are only applicable for sufficiently textured objects, which is often not the case for the objects of interest for our intended application [18].

In [18], we present a system which can build object representations for appearance-based recognition and localization automatically, given a 3D model of the object. An initial estimate for the position of the object is determined through stereo vision, while an initial estimate for the orientation is determined by retrieving the rotation the recognized view was produced with. Then, a number of correction calculations are performed for accurate localization, which is explained in detail in [18]. An outline of the overall algorithm is given by the following steps:

1) Perform color segmentation in both images.
2) Determine color blobs with a connected components algorithm.
3) Match the blobs in the left and right image on the base of their properties and the epipolar geometry.
4) For each matched blob:
5) Calculate initial estimate for the position by stereo triangulation.
6) Determine the best matching view by calculating the Nearest Neighbor in the PCA eigenspace.
7) Determine initial estimate for the orientation by retrieving the stored rotation for the recognized match.
8) Apply pose correction formulate as presented in [18].
9) Verify validity by comparing the size of the blob to the expected size, determined on the base of the calculated pose and the object model.

Fig. 8. Illustration of the color segmentation result for the colors red and green

As we show in [18], our system is very robust and is able to recognize and localize the objects in our test environment accurately and reliably in real-time. Recognition and localization for one potential region takes approximately 5 ms on a 3 GHz CPU, with a database of five objects: a cup, a cup with a handle, a measuring cup, a plate, and a small bowl. An exemplary segmentation result is shown in Figure 8; the result of a full scene analysis is visualized in Figure 9.

V. Discussion and Conclusion

At this point it is important to mention the work of Kragic et al. [6] due to the similarity in some aspects to our work. They present a visual tracking system also able to recognize objects. Once an object is recognized the model and pose of it is sent to GraspIt!. A human operator uses GraspIt! visualization and

Fig. 9. Recognition and localization result for an exemplary scene. Left: left input image. Right: 3D visualization of the result.

analysis tools to determine a stable grasp with the Barrett Hand. Later the grasp is executed.

On the visual part the main difference is that we are able to deal with arbitrarily complex shaped objects, while Kargic et al. are limited to planar-faced objects. Another main difference to our approach is that we compute grasps automatically and offline, without a human operator. The addition of these features, five-fingered hands, automatic grasping synthesis, and realistic shaped objects in a realistic environment (but with simplified texture/colours) makes our approach more complete and autonomous.

To conclude, in this paper we have presented an integrated approach that includes an offline grasp planning system with an visual object identification system. The integration of these two modules relies on the use of an appropriate object and grasp representation database that is also described.

However, the work presented here is only a part of a larger manipulation system. Some modules are still required, in order to execute any of the grasps computed. First, in any situation several grasp candidates are possible, but only one can be executed. A module that selects one taking into account the task and the execution conditions is necessary. Once a grasp is selected, an arm motion planner is necessary to move the hand to the pregrasping location according to the grasp description and the object pose. And finally, a module that executes the grasp using tactile and visual feedback has to be developed too.

ACKNOWLEDGMENT

The work described in this paper was partially conducted within the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft) and the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission. Support for the first author is provided in part by the spanish government under project DPI2001-3801 and DPI2004-01920; by the Generalitat Valenciana under projects inf01-27, GV01-244, CTIDIA/2002/195, GV05/137; and by the Fundació Caixa-Castelló under project P1-1B2005-28, P1- 1A2003-10.

REFERENCES

[1] A. Konno, K. Nagashima, R. Furukawa, K. Nishiwaki, T. Oda, M. Inaba, and H. Inoue, "Development of a humanoid robot *saika*," in *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, Grenoble, France, Sept. 1997, pp. 805–810.

[2] C. Laschi, P. Dario, M. Carrozza, E. Guglielmelli, G. Teti, D. Taddeucci, F. Leoni, B. Massa, M. Zecca, and R. Lazzarini, "Grasping and manipulation in humanoid robotics," in *IEEE International Conference on Humanoid Robots (Humanoids 2003)*, Karlsruhe, Germany, Oct. 2003.

[3] A. Edsinger-Gonzalez and J. Webber, "Domo: a force sensing humanoid robot for manipulation research," in *IEEE International Conference on Humanoid Robots*, Santa Monica, California, Nov. 2004, in CD.

[4] S. Schulz, C. Pylatiuk, A. Kargov, R. Oberle, and G. Bretthauer, "Progress in the development of anthropomorphic fluidic hands for a humanoid robot," in *IEEE-RAS/RSJ Inernational Conference on Humanoid Robots (Humanoids 2004)*, Santa Monica, California, Nov. 2004.

[5] W. Bluethmann, R. Ambrose, M.Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Autonomous Robots*, vol. 14, no. 2–3, pp. 179 – 197, Mar. 2003.

[6] D. Kragic, A. Miller, and P. Allen, "Real-time tracking meets online grasp planning," in *IEEE International Conference on Robotics and Automation*, Seoul, Republic of Korea, May 2001, pp. 2460–2465.

[7] G. Taylor and L. Kleeman, "Integration of robust visual perception and control for a domestic humanoid robot," in *IEEE International Conference on Robotics and Automation*, Sendai, Japan, Sept. 2004, pp. 1010–1015.

[8] R. Platt Jr., A. Fagg, and R. Grupen, "Extending fingertip grasping to whole body grasping," in *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sept. 2003, pp. 2677–2682.

[9] R. Platt Jr., O. Brock, A. Fagg, D. Karupiah, M. Rosenstein, J. Coelho, M. Huber, J. Piater, D. Wheeler, and R. Grupen, "A framework for humanoid control and intelligence," in *IEEE International Conference on Humanoid Robots (Humanoids 2003)*, Karlsruhe, Germany, Oct. 2003.

[10] M. Cambron and R. Peters, "Learning sensory motor coordination for grasping by a humanoid robot," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, Nashville, Tennessee, Oct. 2000, pp. 6–13.

[11] R. Becher, P. Steinhaus, and R. Dillmann, "ARMAR II - a learning and cooperative multimodal humanoid robot system," *"'International Journal of Humanoid Robotics"'*, vol. 1, no. 1, pp. 143–155, 2004.

[12] T. Asfour, K. Berns, and R. Dillmann, "The humanoid robot ARMAR: Design and control." in *The 1st IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2000)*, MIT, Boston, USA, 7-8 September, 2000.

[13] M. Cutkosky, "On grasp choice, grasp models and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[14] A. Miller and P. Allen, "Graspit!: A versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, Dec. 2004.

[15] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *IEEE International Conference on Robotics and Automation*, Taipe, Taiwan, Septmeber 2003.

[16] Barrett Technology Inc., http://www.barrett.com/.

[17] C. Ferrari and J. Canny, "Planning optimal grasps," in *IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 2290–2295.

[18] P. Azad, T. Asfour, and R. Dillmann, "Combining appearance-based and model-based methods for real-time object recognition and 6d-localization," in *International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[19] S. Nayar, S. Nene, and H. Murase, "Real-time 100 object recognition system," in *International Conference on Robotics and Automation (ICRA)*, vol. 3, Minneapolis, USA, 1996, pp. 2321–2325.

[20] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision (ICCV)*, Corfu, Greece, 1999, pp. 1150–1517.

[21] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, Tokyo, Japan, 2003, pp. 93–102.

[22] E. Murphy-Chutorian and J. Triesch, "Shared features for scalable appearance-based object recognition," in *IEEE Workshop on Applications of Computer Vision*, Breckenridge, USA, 2005.

[23] S. Obdrzalek and J. Matas, "Object recognition using local affine frames on distinguished regions," in *British Machine Vision Conference (BMVC)*, vol. 1, Cardiff, UK, 2002, pp. 113–122.