Project no.:	**IST-FP6- IP-027657**
Project full title:	**Perception, Action & Cognition through Learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**
Deliverable no.:	**D 2.1**
Title of the deliverable:	**Technical report describing the adaptive algorithms and the designed software**

Contractual Date of Delivery to the CEC:	January 31st, 2007
Actual Date of Delivery to the CEC:	January 30th, 2007
Organisation name of lead contractor for this deliverable:	JSI
Author(s):	Aleš Ude, Damir Omrčen, Tamim Asfour, Danica Kragic, Norbert Krüger, Juan-Andrade Cetto, Babette Dellen
Participant(s):	JSI, UniKarl, KTH, BCCN, AAU, CSIC, UEDIN
Work package contributing to the deliverable:	WP2 (Sensorimotor primitives for learning of OACs)
Nature:	R/D
Version:	1.0
Total number of pages:	22
Start date of project:	1st Feb. 2006 **Duration:** 48 months

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level

PU	Public	
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)	X

Abstract:

This technical report summarizes the algorithms and software developed in WP2 in the first twelve months. It reports on the development and implementation of sensorimotor primitives that can be used to explore the robot's environment, to learn sensory representations of new objects, and to act upon objects (which in this phase are still called "things" in PACO terminology). The following learning approaches and primitive movements are described: oculomotor control and foveation, learning of inverse kinematics and reaching, automatic grasp generation by integrating human demonstration and object shape, modeling of corrective movements using tactile information, and exploratory sensorimotor acts for learning object representations. These primitives provide the means for learning of early object-action complexes. To enable their execution in real-time, we designed a computer architecture that allows us to distribute and synchronize the processing of sensory data on a network of PC computers in real-time. We implemented a real-time visual attention system to demonstrate the capabilities of this system.

Keyword list: Sensorimotor primitives, foveation, grasping, distributed computing

Table of Contents

1 Introduction

The development and emergence of higher-level cognition is only possible on an artificial system with a rich repertoire of sensory and motor capabilities. For this reason humanoid robots are our hardware system of choice for the study of embodied cognition. A humanoid robotic system with the ambition of developing higher-level cognitive categories must be able to actively explore its environment and to interact with people and other agents in its world. A necessary condition for these activities is the existence of low-level sensorimotor processes that provide means for feed-forward and feedback control of the robot's actions based on the continuous stream of incoming percepts. Low-level sensorimotor processes also facilitate the acquisition of new knowledge. Hence one of the first aims of the project is to identify and implement these low-level sensorimotor processes on an artificial system. Our goal is to demonstrate how they can be applied to learn about new objects (or things in PACO terminology), which should finally lead to the generation of early object-action complexes (OACs). The basic paradigm of PACO-PLUS is that OACs are the entities on which cognition develops. By grounding the OACs in physical interactions of the robot with the external world, we provide the basis for the development of shared, mutually grounded symbols.

In this first phase of the project we investigated which low-level sensorimotor primitives are essential for a humanoid robot that will be used for the learning of OACs. This report describes the identified low-level sensorimotor processes, their implementation and our first experiments geared towards the generation of early OACs using the realized sensorimotor primitives.

1.1 Sensorimotor Primitives for Exploration and Interaction

The concept of sensorimotor primitives originates in the work of Arbib [1] who viewed motor primitives as a sequence of actions that accomplish a complete goal-directed behavior. Conceptually, the idea of movement primitives is appealing because it allows us to abstract complex motions as symbols, thus providing the basis for higher-level cognitive processes. This has been demonstrated for example in [12], where motor behaviors execute the appropriate primitives to accomplish a verbally described high-level task.

There is no consensus in the literature about how to encode movement primitives. Proposals include nonlinear dynamic attractor systems that can be flexibly adjusted to represent arbitrarily complex motor behaviors [18], primitive flow fields acquired from motion capture data [10], recurrent neural networks [20], HMMs [3, 8], and representations by force fields [14]. The degree of integration between perception and motor control also varies across these proposals.

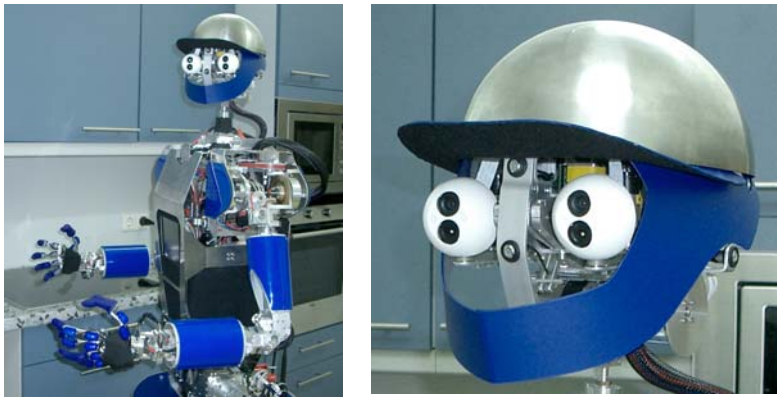


Figure 1. PACO-PLUS humanoid platform ARMAR and its head with foveated vision.

In PACO-PLUS we are interested in the following aspects of sensorimotor primitives: firstly, they provide us with the initial sensorimotor knowledge needed to explore "things" situated in the robot's environment. In this context we are interested to identify which action primitives are needed with the goal of implementing them on the PACO-PLUS hardware. Secondly, by providing means to structure motor knowledge, they supply building blocks that can be used to learn and represent early OACs. The motor knowledge needs to be

parameterized and the appropriate cost functions need to be developed to enable the adaptation of primitives to novel situations and the selection of primitives incorporated into the OACs in a given state.

Figure 2. Simultaneous views from the peripheral and foveal cameras. The high resolution of object image and better localization makes foveal images suitable for image analysis task such as object recognition (right), while the wide field of view from the peripheral camera is suitable for visual search and smooth pursuit (left).

For PACO-PLUS experiments we shall use the humanoid robot ARMAR developed by the University of Karlsruhe. ARMAR is equipped with the foveated vision system that mimics the foveated structure of the human eye. Foveation is implemented using two cameras in each eye; one of the cameras is a narrow-angle foveal camera and the second is a wide-angle camera for peripheral vision. Foveation is useful because, firstly, it enables the robot to monitor and explore its surroundings in images of low resolution, thereby increasing the efficiency of the search process, and secondly, it makes it possible to simultaneously extract additional information - once the area of interest is determined - from higher resolution foveal images that contain more detail (see Figure 2). The exploration of unknown environments can definitely benefit from a vision system with such properties. In addition, ARMAR is equipped with two five-finger hands providing tactile feedback that can support grasping.

Besides motor and sensory capabilities, we also need to consider the problem of real-time processing. On-board processing is best suited for tasks that require tight integration between perception and motor control and small delays. However, on-board processing becomes too limiting when emulating higher-level, computationally expensive cognitive processes. We have therefore developed a cluster architecture that enable us to better explore various levels of cognitive processing on the humanoid system while still providing real-time responses, albeit with somewhat higher latencies.

As described above, our first task was to identify and implement sensorimotor primitives needed to find and explore new objects in the scene. To start learning higher-level concepts, the robot should be able to explore its environment, reach for the objects and manipulate them. On the perceptual side, the robot needs to be able to identify possible regions of interests, discern new objects from the background and acquire sensory representations that can be used for recognition. Starting from a study elucidating the properties of foveated vision systems, we concentrated on the following primitive actions to enable initial exploration needed to find objects and build early object-action complexes:

- searching for objects (visual attention) and saccades,
- inverse kinematics learning and reaching,
- grasping of unknown objects,
- smooth pursuit of grasped objects and tracking,
- explorative movements and the associated sensory processing for the generation of viewpoint independent object representations.

Since grasping, eye movements and arm movements are fundamentally different motor acts, we did not attempt to find a unified motor representation. There may well be that no single representation exists and that different movement primitives are encoded differently. However, we are currently working on developing more unified representations in limited domains such as for example reaching and grasping.

2 Foveation and Lessons for Oculomotor Control

In a foveated vision system, the main task of the eye control system is to place a salient region over the field of view of foveal cameras so that more accurate analysis of regions of interest can be accomplished. Although the focus of the task is to bring an object into the center of the fovea, the control system uses the view from peripheral cameras as the basis for control. Data from peripheral images is more reliable because objects can easily be lost from the foveal views due to the narrowness of the viewfield (see Figure 2).

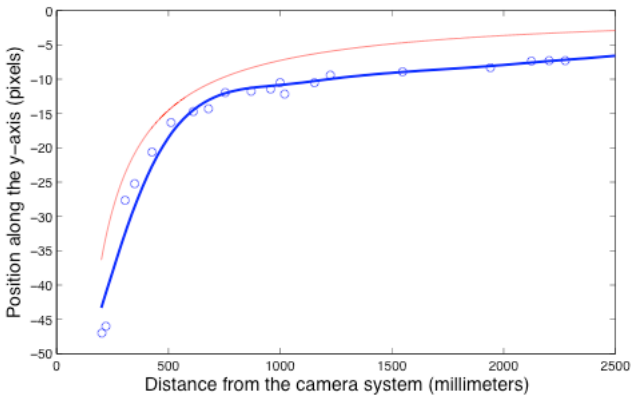


Figure 3. Red curve: y_p^0 with respect to the distance of the object from the camera system as calculated by Eq. (1) (totally aligned, ideal pinhole cameras, R = I, $t_x = t_x = 0$, $t_y = 25$, $\alpha_p = \beta_p = 290.9$). Blue curve: y_p^0 determined experimentally by placing the object manually at various distances so that it projects on the center of the foveal image. At each such configuration we measured the object's position in peripheral image and its distance from the eye (using stereo vision). The blue circles show these measurements.

Two issues need to be considered when analyzing the foveation setup with two cameras:

1. Given a 3-D point that projects onto the center of the foveal image, where will the point be projected onto the peripheral image? This will be the ideal position in the periphery for foveation.
2. If a 3-D point projects onto the peripheral image away from the ideal position described above, how far is the projection of the point from the center of the foveal image?

A thourough analysis of these two problems is provided in Section III of paper [B] attached to this report. The main lessons can be drawn by studying the solutions arising in the case when 1. the origin of the world coordinate system is chosen to be in the projection center of the foveal camera with z-axis equal to its optical axis, and 2. the eye is constructed in such a way that the optical axes of both cameras are parallel and that the projection centers of both cameras are placed at $Z = 0$. In this case we have the following expression for the position (x_p^0, y_p^0) in the peripheral image that results in the projection onto the principal point in the foveal image for a 3-D point at distance Z from the camera,

$$x_p^0 = -\frac{\alpha_p \mathbf{r}_1^{\mathrm{T}} \mathbf{t}_{pf}}{Z}, \quad y_p^0 = -\frac{\beta_p \mathbf{r}_2^{\mathrm{T}} \mathbf{t}_{pf}}{Z}, \tag{1}$$

where $\mathbf{t}_{pf} = [t_x, t_y, 0]$ is the position of the origin of the peripheral coordinate system expressed in the foveal coordinate system and $\mathbf{R}_{pf} = [\ \mathbf{r}_1^{\mathrm{T}}, \mathbf{r}_2^{\mathrm{T}}, \mathbf{r}_3^{\mathrm{T}}\ ]^{\mathrm{T}}$ is the rotation matrix that rotates the basis vectors of the peripheral coordinate system into the basis vectors of the foveal coordinate system. α_p, β_p are the scaling factors of the peripheral camera along both image axis. This tells us that knowing the distance of the object from the eye we can calculate the position in the peripheral camera that corresponds to the cental position of the projection in the foveal camera.

If the projection of a 3-D point at distance Z is displaced from (x_p^0, y_p^0) by (d_x, d_y), then we have the following expression for the displacement (D_x, D_y) of the foveal projection of this point from the principal point in the foveal image

$$D_x \approx \frac{\alpha_f}{\alpha_p} d_x, \quad D_y \approx \frac{\beta_f}{\beta_p} d_y . \tag{2}$$

For example, in one of our foveation setups, the peripheral cameras are equipped with 3mm lenses and with CCD chips of size 6.6 x 4.4 millimeters, while the foveal cameras are equipped with 12mm lenses and with CCD chips of size 3.3 x 2.2 millimeters. The distance between them is about 25mm along the y-axis ($t_x \approx 0$, $t_y \approx 25$). Theoretically, the scaling factors of such cameras are $\alpha_p = \beta_p \approx 290.9$ and $\alpha_f = \beta_f \approx 1306.8$ when the cameras are calibrated for images of size 640 x 480.

Figure 3 shows the variation of y_p^0 with respect to Z under such assumptions and proves that our system indeed exhibits such characteristics. For Z = 1 meter, the ideal position in the periperal image is given by $x_p^0 = 0$ and y_p^0 = -290.9 * 25 / 1000 = -7.3 pixels. For objects further away, y_p^0 tends to zero. From Eq. (1) it follows that the necessary displacement doubles to -14.6 when $Z \approx 498$ mm. Hence, if we fix Z to 1 meter and observe objects more than 0.5m away from the camera, the systematic error in the peripheral images will be less than 7.3 pixels. Eq. (2) tells us that the displacement from the central position in the foveal view will be at most 1306.8 / 290.9 * 7.3 ≈ 32.8 pixels, hence we are still relatively close to the principal point in the foveal image. Note that fixing the distance Z is equivalent to replacing the perspective projection with the orthographic projection in our model.

Based on these results we can conclude that it is possible to control the foveated setup based on 2-D information and without using distance information. The maximum error is limited as long as the object is enough far away from the eye. Hence we can avoid calibrating the cameras and we copntrol the eyes using only 2-D feedback.

3 Oculomotor Control and Visual Attention

3.1 Saccades and Smooth pursuit

The main task of the oculomotor system is to find objects of interest (selective attention) and to capture targets on a very narrow fovea. We have implemented saccades for visual attention and smooth pursuit for tracking moving objects. Since we are primarily interested in manipulation tasks and interactions that do not require visual processing during large movements in space, the vestibulo-ocular reflex and optokinetic response for gaze stabilization were not implemented, but can be added in the future if need arises.

Figure 4. Computer cluster at JSI

We developed a head and eye control system that is appropriate to realize both saccadic movements and smooth pursuit and enhances the appearance of the humanoid through mimicking various aspects of human movements: human eyes follow target movements, but without head movements they have a limited range; thus, the robot's control system supports its eye movements through head movements and thus exploits the redundancy of the humanoid head. The details of the control system are explained in the attached paper [B]. The same control system is used to implement both saccadic movement and smooth pursuit. The difference between both type of movements (saccades are very fast movements towards a current focus of attention, whereas smooth pursuit is much more smoother) is realized by increasing the gains and suppressing visual processing when saccading towards a new focus of attention.

3.2 Visual Attention

The visual attention system is described in more detail in paper [24]. It is based on the attention model proposed by Itti, Koch, and Niebur [9], but adds two additional cues: motion and disparity. The main goal of our implementation was to demonstrate the abilities of the distributed architecture we use for the processing

of visual information (see the next section) and to demonstrate how top-down effects can be incorporated into the proposed model, which was initially purely bottom-up. More details can be found in the paper.

3.3 Processing of Sensorimotor Information

Our architecture for sensorimotor processing is based on the processing of visual information in the brain. Visual information is transferred along a number of pathways (e. g. magnocellular pathway, parvocellular-blob pathway, and parvocellular-interblob pathway) and visual processes are executed in well-defined areas of the brain. Visual perception results from interconnections between these partly separate and functionally specialized systems. We designed a system that allows us to distribute computational processes across a network of computers and transfer information from the source to any node in the cluster responsible for a particular visual process. The computational processes can be executed either sequentially or in parallel and we provided means to integrate information from various streams coming at different frame rates and with different latencies. The transfer of information can be both feed-forward (bottom-up processing) and feed-backward (top-down effects). With this framework we gain the capability to explore a greater range of cognitive architectures than previously possible. The effectiveness of the distributed architecture has been demonstrated with the implementation of the visual attention system. We intend to use it to realize various computationally expensive processes such as for example motion capture and movement learning in real-time.

4 Grasping

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. To interact with the environment, the robots have to be able to manipulate both known and unknown objects. Planning object manipulation and grasping is difficult due to the large search space resulting from all possible hand configurations, grasp types, and object properties that occur in regular environments. In our work, robot learning from demonstration, experience and shape primitives are used to provide a successful object grasping strategies. A top-down (experience) and a bottom-up methodology are integrated to develop a more natural grasp learning system. The specific focus here is on 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 perceptual cues but also on the experience that some approach vectors will provide useful tactile cues that finally result in stable grasps. The approach is evaluated using two kinematically different hands, the Barrett hand and the Robonaut hand. A detailed experimental evaluation of the system is presented in paper [C] and the most important results are summarized below.

Figure 5. First row: The real objects. Second row: The modelled objects. Third row: The object primitives used for training.

4.1 Modeling and evaluation of automatic grasp generation by integrating human demonstration experience and object shape primitives

Since it can be difficult to automatically acquire detailed models of complex objects, it is more reasonable to represent objects by their *shape primitives*. The basic shape primitives are e.g. truncated cone, sphere, box, cylinder, as shown in Figure 5. For the grasping process, it is assumed that the objects are placed on a table and the pose is denoted by only three parameters, representing their position and orientation on the table. More details can be found in [4-6].

4.1.1 Grasp Mapping

An off-line learnt grasp mapping procedure maps the human grasps to robot grasps where grasp preshapes are used to limit the large number of possible robot hand configurations. The current grasp recognition system can recognize ten different grasp types, [5]. 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. Details about this can be found in [C]. 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 and initial hand postures are used to grasp an object dependent on the grasp type, Figure 6.

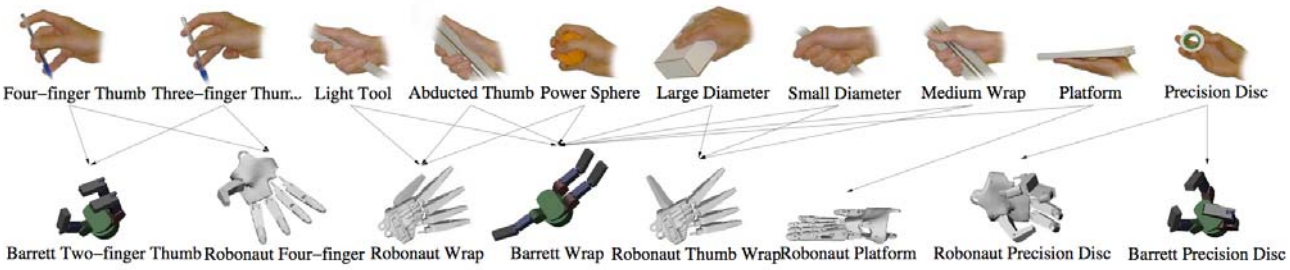


Figure 6. Initial robot hand postures for different grasp types.

4.1.2 Grasp Control

Two basic grasp controllers in the system are developed, one for power and one for precision grasps. 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.

- **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 visual input, as we previously demonstrated in [11].

The grasp approach vector is defined relative to the object's pose. 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.

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 $\theta = 9$, $\phi = 17$, $\psi = 9$, $d = 6$. 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 quality measures for each grasp are stored in a *grasp experience* database. To evaluate grasps, the 6-D convex hull spanned by the forces and torques that the grasp can resist is analyzed using GraspIt! [13].

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. 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.

4.1.3 Introducing Error in Pose Estimation

To evaluate the performance under imperfect pose estimation, we have simulated errors in pose estimation by adding an offset to the object pose. 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 grasp 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.

Figure 7 and Figure 8 show the grasp success rates for various grasps and two 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 a 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 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, Figure 8, where the Robonaut Four-finger Thumb grasp is the best.

The bottle and the mug have been trained both using a primitive model and using the real model (see Figure 5). 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.

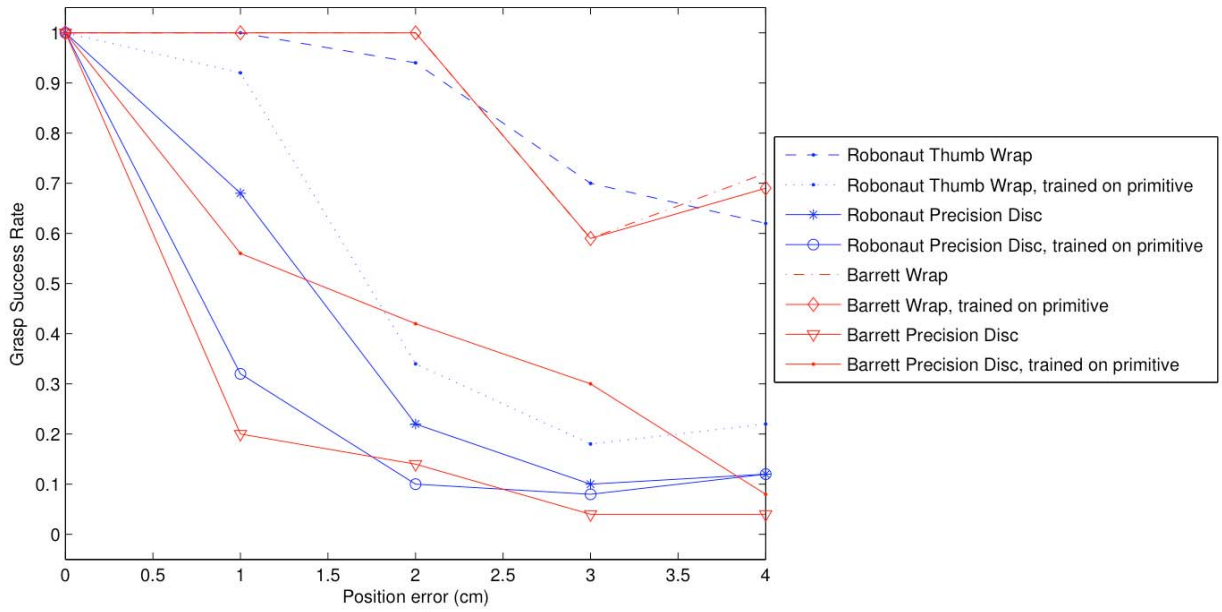


Figure 7. Grasping the bottle.

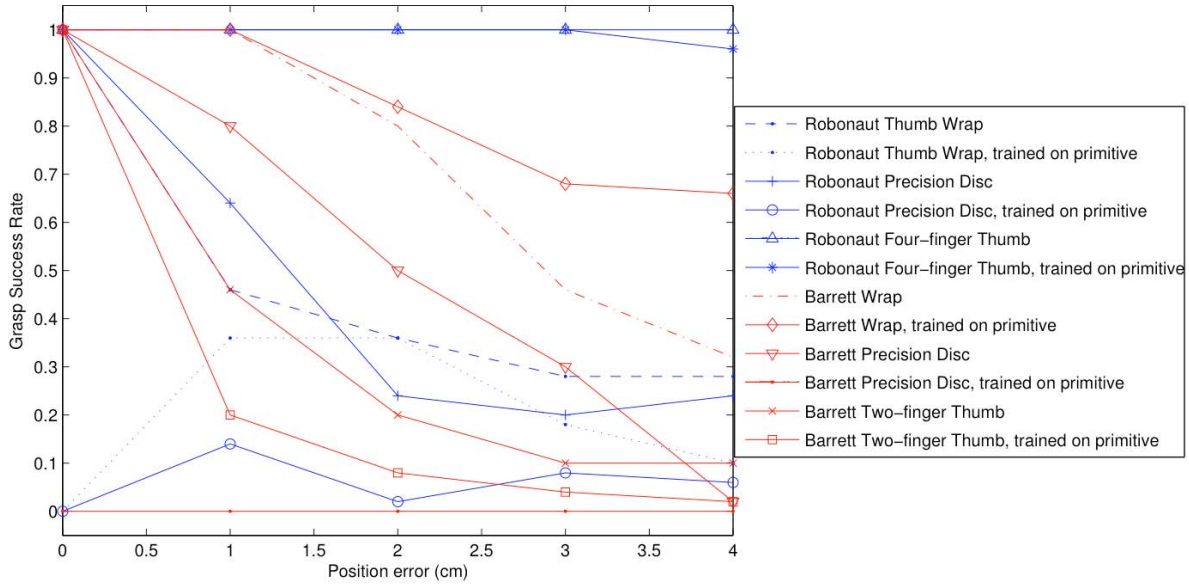


Figure 8. Grasping the mug.

4.2 Modeling of corrective movements using tactile information once contact with the object has been established

Once the hand has been placed into the appropriate position relative to the object and a suitable grasp type has been chosen, the grasping sequence can be seen as comprised of two final phases; first closing the fingers until contact and then maintaining the contact while applying corrective movements based on contact forces. Before the initial contact with the object has been made, 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. For this purpose, 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 joint torque and thus position control requires D-control or friction modelling.

4.2.1 Controller 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 [21]. 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 Figure 9. The transform is

$$\mathbf{x} = \mathbf{T}\mathbf{q} \tag{3}$$

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 angle is not controlled.

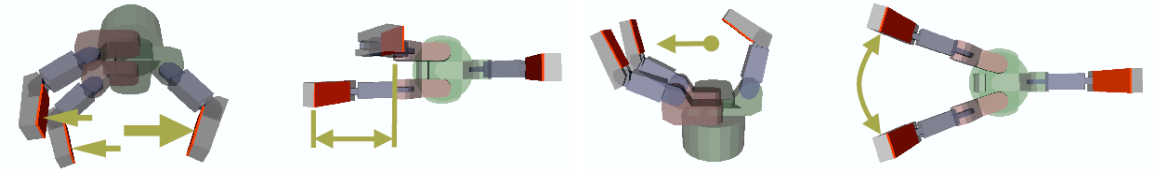


Figure 9. Grasp controllers: total grasp force, stability, centering, and spread.

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

$$x_2 = \frac{q_2 + q_3}{2} + q_4. \tag{4}$$

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:

$$x_3 = \frac{q_2 + q_3}{2} - q_4. \tag{5}$$

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:

$$x_4 = q_2 - q_3. \tag{6}$$

Controlling the force, centering and stability according to the above and Figure 9, the transform becomes:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 1/2 & 1/2 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}. \tag{7}$$

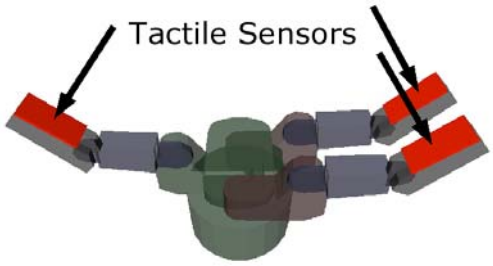


Figure 10. The placement of the tactile sensors.

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

$$\mathbf{F} = \mathbf{T}^{\mathrm{T}}\mathbf{f} = \mathbf{T}^{\mathrm{T}}\mathbf{De}. \tag{8}$$

In the current system, it is assumed that three extrinsic tactile sensors capable of detecting the normal force only were mounted to the distal links, see Figure 10. These sensors 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

$$\begin{aligned} &\mathbf{e} = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 \end{bmatrix}^{\mathrm{T}}, \\ &e_1 = 0,\ e_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}\mathbf{e}_f,\ e_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{e}_x,\ e_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{e}_x \\ &\mathbf{e}_f = \mathbf{f}_{des} - \mathbf{f}_{act} = \mathbf{f}_{des} - \mathbf{T}^{-\mathrm{T}}\mathbf{F}_{act},\ \mathbf{e}_x = \mathbf{x}_{des} - \mathbf{x}_{act} = \mathbf{x}_{des} - \mathbf{T}\mathbf{q}_{act} \end{aligned} \tag{9}$$

To focus on the displacement control, we use

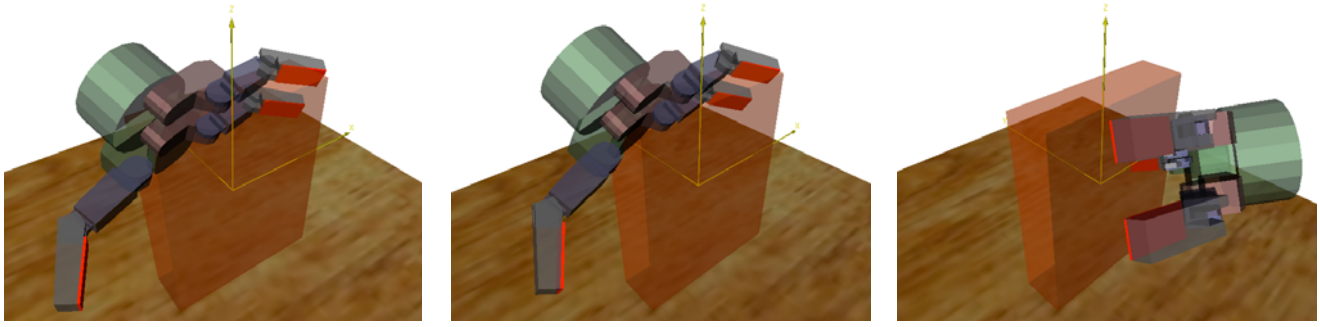
$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & k_p & 0 & 0 \\ 0 & 0 & 5k_p & 0 \\ 0 & 0 & 0 & k_p \end{bmatrix}. \tag{10}$$

4.2.2 Evaluating the Controller

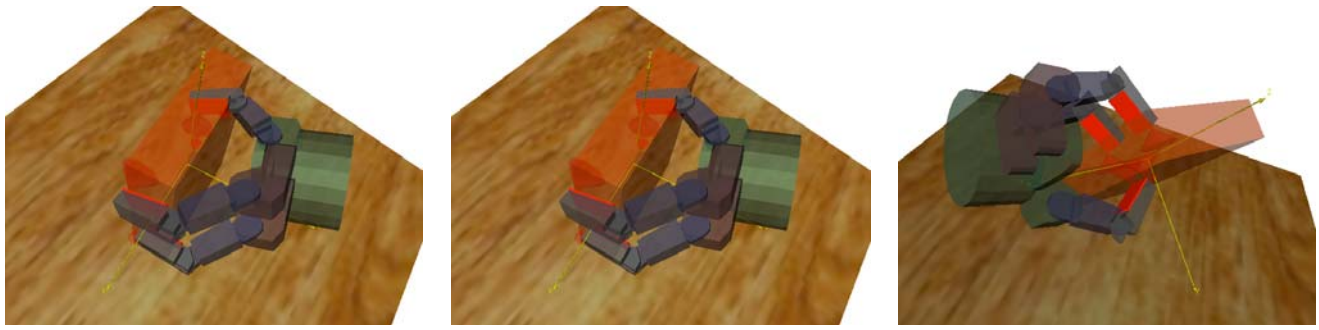
Grasping of a polyhedral object was dynamically simulated using the proposed controller. As in the previous case, 1377 initial hand grasping postures have been evaluated. 1035 were automatically discarded because the hand interfered with the table upon which the box is placed while approaching the object, or that the object was obviously out of reach. The remaining 342 initial robot hand positions were evaluated and resulted in 172 force closure grasps and 170 failed grasps. The major reason for the low sucess rate is the fact that the wrist position is not controlled once the corrective movements are initiated and the low number of degrees of freedom of the hand do not offer much flexibility. Thus, a good initial posture of the hand is very important for the grasp sucess.

The top three hand initial positions and the resulting grasps are shown in Fig. 7. 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.

Some sample data from the third best simulation, Figure 11 c) and f), is shown in Figure 12. 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 passes. 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

Figure 11. The top three approach positions and the final grasps for the rice box.

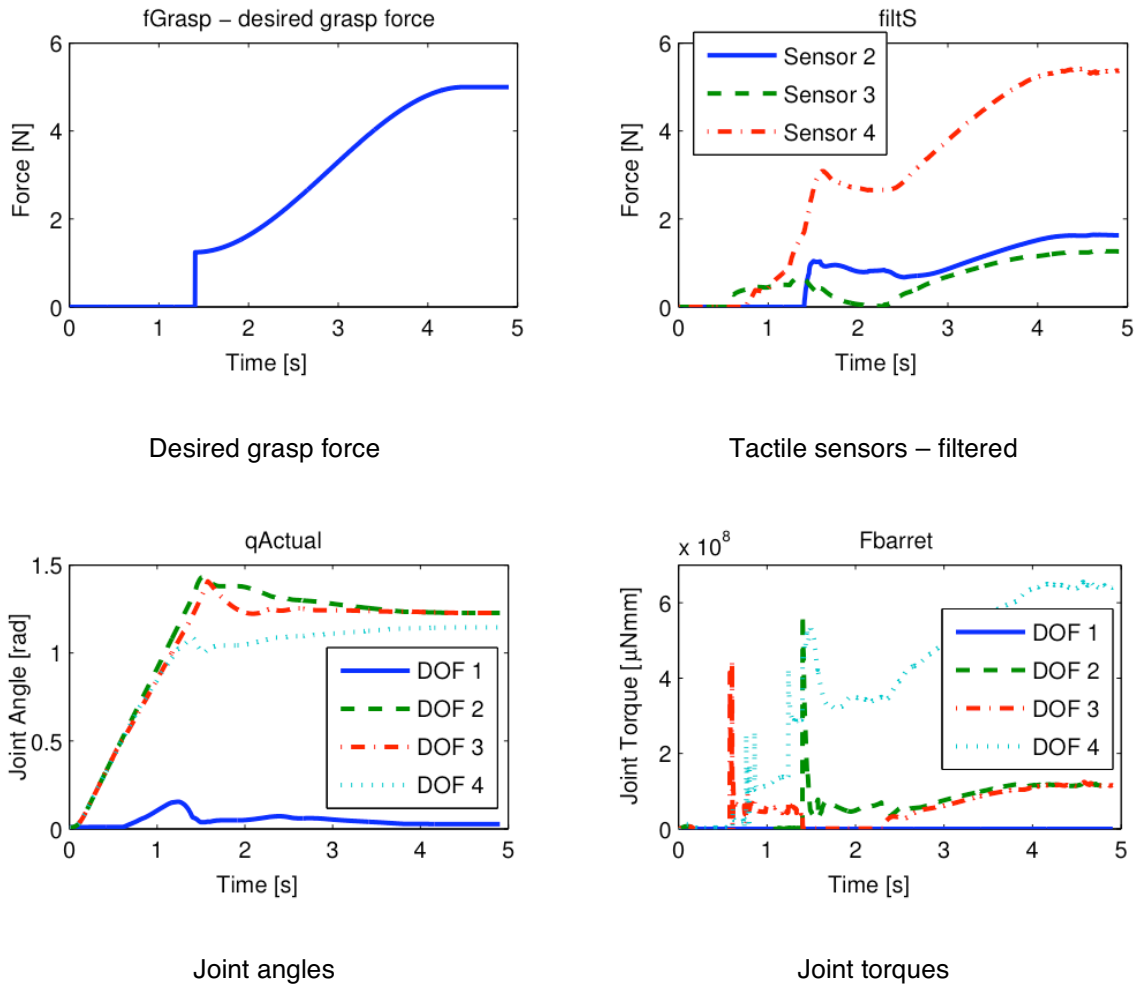


Figure 12. Data logged from the grasp simulation in Figure 11 c and f. The desired grasp force is set to 5 N.

Figure 13 shows the results of grasp success rates with and without corrective movements. Here, only 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 was 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 demonstrate the validity of the proposed system.

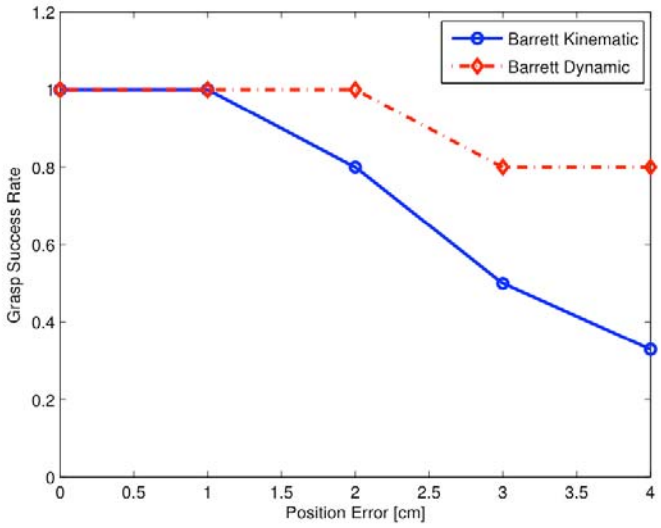


Figure 13. Grasp success rates with and without corrective movements.

5 Inverse Kinematics and Reaching

To enable reaching for objects, we need to represent robot kinematics in some way. The classic Inverse Kinematics (IK) mapping goes from joint angle values (**θ**) to Cartesian coordinates (**x**). One of the challenges when

learning the IK is how to deal with the one-to-many property of the mapping. The formulation of the problem determines which mapping is really learned:

1) $\mathbf{x} \rightarrow \boldsymbol{\theta}$. Direct learning of IK [17]. Most learning systems deal with the uncertainty by averaging the output. Unfortunately, averaging multiple IK solutions does not produce an IK solution in general. Thus, with this strategy one can only learn the IK of non-redundant systems.

2) $(\Delta\mathbf{x}, \boldsymbol{\theta}) \rightarrow \Delta\boldsymbol{\theta}$. Learning how to modify slightly $\mathbf{x}$ by means of small movements in the joints [25]. In the vicinity of a given $\boldsymbol{\theta}$, the average is a truly IK solution. Therefore, incorporating $\boldsymbol{\theta}$ to the input allows valid localized solutions. It is also possible to bias the movements of the robot towards configurations, so that it becomes incorporated in the learned mapping.

3) $\boldsymbol{\theta} \rightarrow \mathbf{x}$.[1] This strategy consists of first learning the Forward Kinematics. The learned forward model can then be processed in several ways to obtain the IK information. One of them is based on the Resolved Motion Rate Control (RMRC) using the forward Jacobian [19]. The second option is to use a forward model like PSOMs do [26]. A search in the joint space is made looking for the values that best match the Cartesian coordinates. The cost function can include in a natural way terms evaluating fixed joint values that can change at any moment, and with little more effort, arbitrary cost terms. The important drawbacks of PSOMs are that the samples must be distributed on a regular grid making it unfeasible for on-line learning.

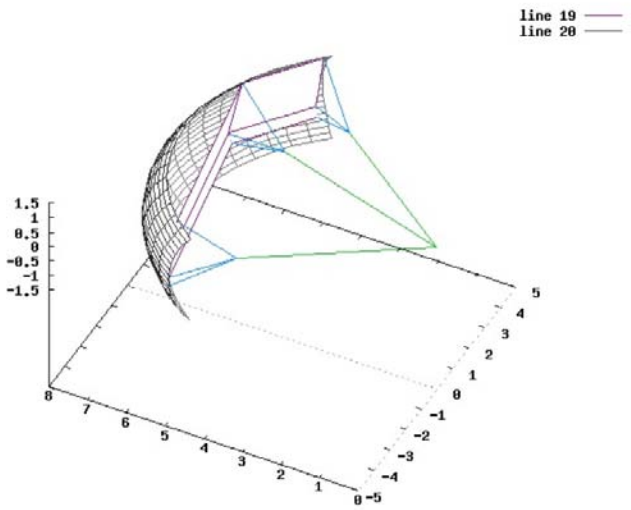


Figure 14. Example model that approximates exactly the kinematics of a two-joint robot arm (whose workspace has the form of a torus) from nine randomly chosen samples.

We have developed a learning model that is based on the third strategy, i. e, a model for the $\boldsymbol{\theta} \rightarrow \mathbf{x}$ mapping. The model is strongly biased, so that it is only able to represent forward kinematics. That means that a lot of a priori knowledge is embedded into the model, which allows the interpolation and even extrapolation with zero error using only $3n$ samples in the absence of noise (Figure 14). The Jacobian of this forward model can be efficiently obtained for use with an RMRC approach, in the PSOMs style. However, for the current implementation we prefer to use a standard optimization algorithm to carry out the search directly in the model, in a way rather similar to PSOMs. Unlike PSOMs, it is very easy to adapt gradually to changes (on-line learning capabilities) and it is possible to cope with irregular sample distributions. Our current work deals with testing our algorithms for obtaining the IK of the ARMAR robot.

Provided the robot learned its kinematics, it can reach for object using standard robotics methods. However, just the reaching position is often not the only relevant parameter for action planning. We are therefore looking at methodologies to generate reaching movements from a library of stored examples that can encode the style and other properties of a given movement type. Our current experiments are geared towards the generation of new movements by superposition of example movements, using for example locally weighted regression [2], and the execution of these movements on the humanoid platform. The data we work with consists of a library of reaching movements captured by a magnetic motion capture system.

[1] Alternatively, it is possible to learn a self-organizing map that matches the arm joints directly with the eye joints (when the eyes are locked on a 3-D target position) instead of with 3-D positions [7].

6 Exploratory sensorimotor primitves for learning object representations

Sensorimotor primitives described in the previous sections are rather general and can be used as such for the initial search and manipulation of objects. However, it is often necessary to study more specific primitives to solve difficult problems in cognition. One such example is the learning of representations for new objects; once the robot found a "thing" in its environment and succeeded to grasp it, how can we discern the full extent of the object from the background and learn a suitable viewpoint-independent representation for it? A human would normally take the object in the hand, place it at the comfortable distance from the eyes and rotate the hand in order to see the object from different viewpoints. We therefore developed the following learning procedure:

- The beginning of learning is initiated by a user who places a new object into the robot's hand.
- Once the robot holds the object, it moves its hand away from the view of the foveal camera and starts learning the stationary background. Typically, the robot first learns the mean values of pixels in a significantly smoothed image for five seconds, followed by another five seconds of learning the variance in color at each pixel.
- The robot moves its hand to the starting position for the observation of the object. Once it reaches the starting point, the procedure for estimating the position and extent of the manipulated object is initiated (see paper [A] attached to this report). Using the control algorithm described in Section 6.2, the optimal configuration for learning is determined. At this configuration the object is placed in such a way that its projection falls onto the center of the foveal image and the size of the object's image is optimal for learning (big enough to ensure decent resolution but not too close to the image boundary).
- Starting from this configuration, the robot begins to rotate the object about the axes that cause the object to rotate in depth. At the same time the object is kept in the center of the foveal image and its size remains constant. The images of object appearance are collected while manipulating the object. This phase finishes once the hand covers the pre-specified range of motion for the two degrees of freedom.
- The object is placed into the robot hand again at a different configuration and the procedure is repeated.
- Once all of the objects were placed at all relevant configurations and all appearance images were collected, a classifier for object recognition is learned by using a method based on nonlinear multi-class support vector machines, which is described in the attached paper [A].

The visual processes necessary to implement the above procedure are described in the attached paper. Here we describe the motor parts of each primitive. The above procedure requires the following motor primitives:

- A movement that allows the robot to determine the relationship between the image and world coordinate system at the given configuration of the eyes.
- An explorative movement primitive that can be used to determine optimal distance and 3-D position of the object from the robot's eyes so that the object will be in the image center and have appropriate size for learning, i. e. it will cover significant portion of the image while being away from the image boundary.
- A primitive motion that can be used to observe the grasped object from various viewpoints. Due to the limited manipulation capabilities of humanoid robots, it is unavoidable to regrasp the observed object to ensure that the robot looks at it from all relevant viewpoints. However, the number of necessary grasps can be

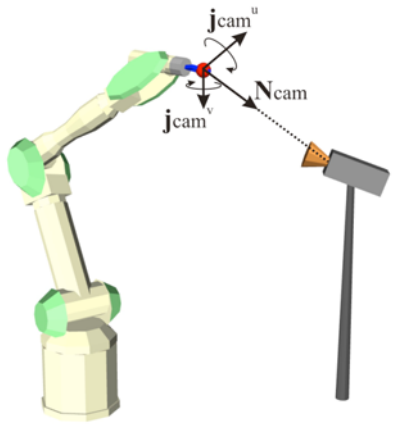


Figure 15. Motion along $\mathbf{j}_{cam}{}^{u}$ and $\mathbf{j}_{cam}{}^{v}$ does not produce any motion along the camera ray. $\mathbf{N}_{cam}$ is the camera null space vector parallel to the camera ray and orthogonal to both $\mathbf{j}_{cam}$ vectors.

reduced by performing the exploratory movements in an optimal way so that the redundancy of the humanoid is exploited and the manipulability of its arm is maximized.

Rotating an object to enable observation from various viewpoints can be quite a challenging task. When the robot rotates the object, joint limits and self-collisions limit its range of movement. Sometimes we can specify the trajectory of the movement in advance considering robot restrictions, but here this is not feasible because the robot head and eyes could be positioned differently in different situations. Therefore the movement trajectories should be adapted or generated on-line.

To observe the object from all sides, the robot should re-grasp the object using one or two robot hands. However, this behavior has not been developed yet and will be implemented in the future.

6.1 Exploratory movements for calibration

We conducted our experiments using a fixed vision system and a robot arm Mitsubishi Pa-10, Figure 15. The experiments on a humanoid platform are currently under development. In the following discussion we limit ourselves to the second hardware system. Mitsubishi Pa-10 has 7 degrees of freedom (DOFs) and is equipped with a gripper. To determine the position of the hand (but not of the object!), we used a previously developed color tracking software [22]. The vision system uses one camera, which is fixed in space. Its position and orientation as well as its intrinsic parameters are not known in advance. In order to position the object in front of the camera, we need to first calibrate the system.

Neglecting the distortion effects, we can write the transformation between the world (= robot) and camera coordinate systems as follows:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{11}$$

where $[x, y, z]^T$ is the 3-D position of the point and $[u, v]^T$ is its projection in the image. The matrix **A** incorporates extrinsic (position and orientation of the camera) and intrinsic (focal lengths, pixel size, image center) camera parameters.

Assuming that we can acquire the position of the hand by vision (this is ensured by putting a marker on the gripper), we can solve the calibration problem by performing a random arm motion observed by the robot's camera. Every point that differs enough from the previous points in the camera or world coordinate system is saved. Once enough points have been acquired, the camera calibration problem can be solved.

Note that the point measurements and the camera calibration process are carried out also in all other phases. When new points are found, the camera is re-calibrated and the calibration gets more precise.

6.2 Placing the object in the image center and determining the optimal distance

With calibrated camera we can place the object grasped by the robot in the center of the camera image. From Eq. (11) we can analytically compute the camera Jacobian that defines relationship between 3-D point velocities and image velocities:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \mathbf{J}_{cam} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}. \tag{12}$$

Since the transformation is underdetermined, one redundant degree of freedom exists, i. e. we can find a vector $\mathbf{N}_{cam}$ in the space of world velocities that does not produce any movement of the point in the image (see Figure 15). This vector is directed along the ray from the projection center to the observed 3-D point.

Figure 15 also shows the two vectors $\mathbf{j}_{cam}{}^{u}$ and $\mathbf{j}_{cam}{}^{v}$, which represent the vectors in the world coordinate system that produce only the movement along u and v direction in the image, respectively, and do not produce any motion along the camera ray. These two vectors are given by the rows of the Jacobian. We can compute $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{u}}$ and $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{v}}$ by normalizing the two rows of the Jacobian

$$\mathbf{j}_{cam}^{u} = \frac{\begin{bmatrix} j_{11} & j_{12} & j_{13} \end{bmatrix}^{\mathrm{T}}}{\left\| \begin{bmatrix} j_{11} & j_{12} & j_{13} \end{bmatrix} \right\|} \text{ and } \mathbf{j}_{cam}^{v} = \frac{\begin{bmatrix} j_{21} & j_{22} & j_{23} \end{bmatrix}^{\mathrm{T}}}{\left\| \begin{bmatrix} j_{21} & j_{22} & j_{23} \end{bmatrix} \right\|}.$$

Vector $\mathbf{N}_{cam}$, which does not produce any movement in the image, is in the null space of the camera Jacobian. It can be calculated using vector product:

$$\mathbf{N}_{cam} = \frac{\mathbf{j}_{cam}^{u} \times \mathbf{j}_{cam}^{v}}{\| \mathbf{j}_{cam}^{u} \times \mathbf{j}_{cam}^{v} \|}. \tag{13}$$

For the special case when the point projects onto the center of the image, $\mathbf{N}_{cam}$ is directed along the optical axis and $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{u}}$ and $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{v}}$ lie in the plane parallel to the image plane.

6.2.1 The control algorithm

The controller is composed of two parts. The first part corresponds to the position control of the object and the second part corresponds to the size control of the object. The task of the position controller is to bring the object to the center of the image. The size controller should act in the null space of the camera Jacobian in order not to disturb the position control. Once the object is in the center of the image, it is only moved directly towards or away from the camera, changing the size while the position is kept constant. Hence the following controller can be used:

$$\dot{\mathbf{x}}_c = \mathbf{J}_{cam}^{+} \dot{\mathbf{i}}_c + \mathbf{N}_{cam} \dot{d}_c, \tag{14}$$

where $\dot{\mathbf{x}}_c = [\dot{x} \quad \dot{y} \quad \dot{z}]^{\mathrm{T}}$ is the control velocity in the world coordinate system, $\mathbf{J}_{cam}{}^{+}$ is the pseudo-inverse of the camera Jacobian, $\mathbf{J}_{cam}{}^{+} = \mathbf{J}_{cam}{}^{\mathrm{T}} * (\mathbf{J}_{cam} * \mathbf{J}_{cam}{}^{\mathrm{T}})^{-1}$, and $\dot{\mathbf{i}}_c$ and $\dot{d}_c$ are control velocity vectors which correspond to the point positioning and size setting, respectively. These two vectors are defined by the following P controllers:

$$\dot{\mathbf{i}}_c = K_p^{i} \begin{bmatrix} u_d - u \\ v_d - v \end{bmatrix} \text{ and } \dot{d}_c = K_p^{d} (size_d - size).$$

$[u_d, v_d]^{\mathrm{T}}$ and $[u, v]$ are the desired and the actual position of the point (or object) in the image coordinate system, respectively, whereas $size_d$ and $size$ are respectively the desired and the estimated size of the object in the image. The size of the object in the image is inversely proportional to the distance of the object from the eyes. $K_p{}^{d}$ and $K_p{}^{i}$ are the control gains.

To control a robot we have to define control velocities in the joint space. Since the task vector in this case has three DOFs (position of all three coordinates in space) and the robot has seven DOFs, the degree of redundancy is four. We applied the following controller:

$$\dot{\mathbf{q}}_c = \mathbf{J}_r^{pos+} \dot{\mathbf{x}}_c + \mathbf{N}_r^{pos} \dot{\mathbf{q}}_{manip}. \tag{15}$$

Here $\mathbf{J}_r^{\mathrm{pos}+}$ is the pseudoinverse of the positional part of the robot Jacobian and $\mathbf{N}_r^{pos}$ is the projection in the null space of $\mathbf{J}_r^{pos}$. Due to the robot's redundancy, we can generate additional movements on the robot in the null space of $\mathbf{J}_r^{pos}$. Our choice for the null space motion is to optimize the robot's manipulability.

To show the objects from different viewpoints, the robot needs to rotate it about both image coordinate axis, which are given by $\mathbf{j}_{cam}{}^{u}$ and $\mathbf{j}_{cam}{}^{v}$ in the world coordinate system. It is therefore advantageous to optimize the manipulability for the rotations about both image axes because high manipulability in a certain direction

usually corresponds to a higher ability of motion in the selected direction. Hence we define the null space term as follows:

$$\dot{\mathbf{q}}_{manip} = K_m \nabla \sqrt{\det(\mathbf{J}_r^{dr} \mathbf{W}_m \mathbf{J}_r^{dr\,\mathrm{T}})}$$

where K_m is the controller gain, $\mathbf{W}_m$ is the weight, and $\mathbf{J}_r^{dr}$ is calculated by rotating the rotational part of the robot Jacobian ($\mathbf{J}_r^{rot}$):

$$\mathbf{J}_r^{dr} = \begin{bmatrix} \mathbf{j}_{cam}^{u} & \mathbf{j}_{cam}^{v} \end{bmatrix}^{+} \mathbf{J}_r^{rot}.$$

Here, $\mathbf{J}_r^{dr}$ is the robot Jacobian, where first row corresponds to the rotation about vector $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{u}}$ and the second row to the rotation about $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{v}}$. These are the rotations that correspond to the coordinate axes of the image plane.

In this phase the robot's task is to place the object (center) onto the optical axis of the camera so that its projection has a desired size. In addition to this task, the robot also optimizes the manipulability of the configuration for both rotations. The goal is to position the robot in an appropriate configuration that enables best showing of the object from different viewpoints. Note that additional conditions can be applied in the null space, e. g. joint limits and / or self-collision avoidance.

6.3 Showing the object from different viewpoints

To acquire data about the object from different viewpoints, the robot needs to rotate it in depth with respect to the camera system. Rotation in depth is defined as any rotation with the rotation axis not parallel to the camera ray. Largest rotations in depth will therefore be caused by rotations about $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{u}}$ and $\mathbf{j}_{\mathrm{cam}}{}^{\mathrm{v}}$. Note that the rotation about the vector in the direction of the camera ray ($\mathbf{N}_{cam}$) is not important and can be considered as redundant. Due to the additional two DOFs for rotation, the task now has 5 DOFs and the degree of redundancy is two. The task space control velocity in this case also includes the angular velocity:

$$\dot{\mathbf{q}}_{c2} = \begin{bmatrix} \mathbf{J}_r^{pos} \\ \mathbf{J}_r^{dr} \end{bmatrix}^{+} \begin{bmatrix} \dot{\mathbf{x}}_c \\ \dot{\mathbf{s}}_c \end{bmatrix} + \mathbf{N}_r^{pos,dr} \dot{\mathbf{q}}_{manip}, \tag{16}$$

where $\dot{\mathbf{s}}_c$ is the vector specifying the rotation in depth about both image axes and $\mathbf{N}_r^{pos,dr}$ is the projection in the null space of $[\mathbf{J}_r^{pos\,\mathrm{T}}, \mathbf{J}_r^{dr\,\mathrm{T}}]^{\mathrm{T}}$. In this way we ensure that the arm retains high manipulability while the robot observes the object.

6.4 Extracting information about objects: Surface segmentation using 3D clustering of stereo images

By manipulating an object, the robot can localize it in an image stream and learn appearance models like described in [B]. However, the identification and segmentation of objects sometimes requires more advanced representations involving combined information from different visual attributes, such as binocular disparity, optic flow, texture, shape, and similarity. Our aim is to extract surfaces from stereo image pairs using superparamagnetic clustering [15]. The method of superparamagnetic clustering represents image pixels by a Potts model of spins, which interact so that neighboring spins corresponding to similar pixels tend to align, given an appropriate similarity measure. Then, image segments are identified as clusters of aligned spins. We have extended this method to 3-D images, i. e., stereo pairs and image sequences, by allowing spins belonging to different frames to interact. Here, we use this method to segment stereo image pairs. The technique uses both information about gray-value pixel similarity and disparity information obtained from sparse and/or dense stereo algorithms. The disparities of the pixels are needed to localize the neighbors of pixels in other frames. By this mechanism, correspondences between frames can be established and stereo clusters can form. Usually, sufficiently accurate disparity values are not available for all pixels. At these points, clustering is merely driven by spin interaction within a single frame. In this way, homogenous image regions for which no disparity information is given can be filled in using disparity information from the

bounding edges.

We demonstrate the technique for a real stereo image, which shows a paper box from two different viewing positions, i.e., left and right (Figure 16). The disparity map was computed using a dense stereo algorithm provided by Karl Pauwels. We only consider those disparity values for which the corresponding amplitude value exceeds a certain threshold. The resulting amplitude map is given in (Figure 17). In the clustering algorithm, spins are only allowed to interact with spins in the other image if their amplitude is equal to one, otherwise only interactions within a single frame are allowed. This has the advantage that (i) reliable disparity information can be used to establish correspondences between the stereo images, and (ii) homogeneous image regions for which the amplitude is zero can be filled in using 2D interactions. The resulting spin states of the box stereo pair are given in Figure 18. The 3-D surfaces could be extracted despite incomplete stereo information (Figure 17). The salt and pepper noise visible in the segmented images is largely caused by erroneous disparity estimates, which lead to wrong correspondences. This effect is particularly strong at the edges. In the future, we aim to incorporate highly accurate but sparse disparity information from primitives to improve image segmentation at the edges.

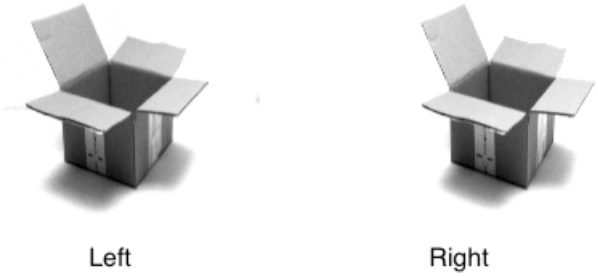


Figure 16. Box stereo pair: Two images, taken from two cameras, left and right view, are shown.

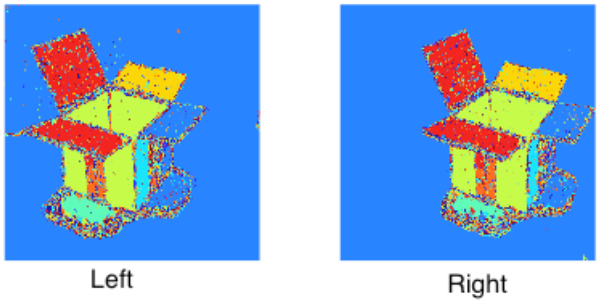


Figure 17. The spin states after 70 iterations: Each color corresponds to a different spin state. Note that the spin states are not identitical with the clusters.

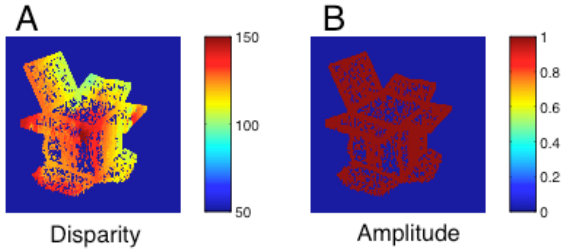


Figure 18. A: Disparity map obtained with Karl Pauwels dense stereo algorithm. B: The corresponding amplitude (confidence values).

7 Action Recognition and Understanding Through Motor Primitives

Neuroscientific and psychological literature states that the core of developmental learning in humans is by watching another person performing a task. This has also motivated the research in the robotics area of learning by imitation and robot programming through demonstration. There is an extensive amount of work dealing with issues of *what, when* and *how* to imitate. In robotics, recognition of human activity has been used extensively for robot task learning through imitation and demonstration. However, there has not been much work on modeling and recognition of activities that involve object manipulation and grasping. In [D], we have modeled and evaluated single arm/hand actions which are very similar to each other in terms of arm/hand motions. The approach is based on the hypothesis that actions can be represented as sequences of motion primitives. The specific questions that the study aims to answer are: 1) Can individual actions be considered as manipulation primitives? 2) If not, can these be broken down into primitives? and 3) How can new actions emerge from known primitives? For this purpose, we consider five different manipulation actions performed on an object: a) pick up, b) rotate, c) push forward, d) push to side, and e) move to side by picking up. To increase the variability, each action is performed by 10 different people in 12 different conditions. To model the process, we are using a combination of discriminative support vector machines and generative hidden Markov models. The experimental evaluation, performed with 10 people, investigates both definition and structure of primitive motions as well as the validity of the modeling approach taken. In [E],

we perform an extensive statistical evaluation for learning and recognition of object manipulation actions. We again concentrate on single arm/hand actions but study the problem of modeling and dimensionality reduction for cases where actions are very similar to each other in terms of arm motions. For this purpose, we evaluate linear and nonlinear dimensionality reduction techniques: Principal Component Analysis and Spatio-Temporal Isomap. Classification of query sequences is based on different variants of Nearest Neighbor classification. We thoroughly describe and evaluate different parameters that affect the modeling strategies and perform the evaluation with a training set of 20 people.

8 Links to other Workpackages

Similarly to WP4.1, WP2 is also concerned with visual and motor primitives needed to explore new objects. Both workpackages thus contribute directly to WP8.1. However, the main concern of WP4.1 was to define visual primitives used to represent objects and to discover strategies for their integration into a coherent object representation (see reference [16], which is included with D4.1.1). On the other hand, the focus in WP2 was to define and implement optimal sensorimotor primitives that can be used to observe objects over a complete view sphere in an optimal way and to build holistic, view-based representations suitable for recognition. Our work on action recognition and understanding through motor primitives is directly linked to WP3, but here the emphasis is on motor primitives. Through this work we also contribute to the demonstration workpackage WP8.2, which is concerned with imitation and other aspects of human-robot interaction.

9 Summary and Outlook

The research conducted until now nicely shows two parallel approaches we are taking towards incorporating sensorimotor knowledge into OACs. On the one hand, we study general approaches to implement and represent sensorimotor primitives in the context of our hardware systems. On the other hand, when necessary we design highly optimized sensorimotor behaviors to accomplish specific tasks arising when learning OACs (like for example the work described in Section 0 and 6). This ensures that the repertoire of available robot actions is large and complex enough, thus avoiding the potential pitfalls of carrying out experiments only in overly simplified environments.

It became clear from our studies that sensorimotor acts can differ significantly among each other. It is unlikely that knowledge about eye movements conveys much information about reaching movements and vice versa. It is therefore reasonable to organize fundamentally different motor acts, such as for example reaching, grasping and eye movements, in separate modules. However, structure enabling interaction between these separate modules needs to be imposed in the future. In addition, while eye movements are pretty independent from the influences of the outside world, other sensorimotor acts, for example reaching and grasping, vary greatly with respect to the target object and the condition of the environment. In the future we intend to study general mechanisms that will allow us to memorize not just each sensorimotor primitive separately, but also to represent the structural relationships among them in the context of the task (reaching towards a target or grasping an object). While some first techniques on combining the existing primitives to novel primitives by superposition have been investigated, more work is necessary to realize representations suitable both for new primitive generation and for action recognition. Besides putting structure on the sensorimotor knowledge (where this is reasonable), we shall also focus on other issues important for PACO-PLUS, e. g. biasing the execution of sensorimotor acts based on the predicted future actions and sequencing of primitives in the context of OACs.

Attached Papers

[A] A. Ude, K. Welke, J. Hale, and G. Cheng. Data Acqusition for Building Object Representations: Discerning the Manipulated Objects from the Background. To be submitted to *IEEE Int. Conf. on Intelligent Robots and Systems*, San Diego, California, 2007.

[B] A. Ude, C. Gaskett, G. Cheng, and M. Kawato. Foveated Vision and Object Recognition on a Humanoid Robot. Submitted to *IEEE Trans. Robotics and Automation*.

[C] J. Tegin, J. Wikander, S. Ekvall, D. Kragic, and B. Illev. Experience based learning and control of robotic grasping. In *IEEE-RAS Int. Conf. on Humanoid Robots, Workshop "Towards cognitive humanoid robots"*, Genova, Italy, December 2006.

[D] I. S. Vicente, V. Kyrki, D. Kragic and M. Larsson, Action Recognition and Understanding Through Motor Primitives. Submitted to *Journal of Advanced Robotics* .

[E] I. S. Vicente and D. Kragic, Learning and Recognition of Object Manipulation Actions Using Linear and Nonlinear Dimensionality Reduction. Submitted to *15th IEEE International Symposiumon Robot and Human Interactive Communication, RO-MAN 2007*

References

[1] M. A. Arbib. Perceptual structures and distributed motor control. In *Handbook of Physiology, Section 2: The Nervous System Vol. II, Motor Control, Part 1*, V. B. Brooks, ed., American Physiological Society, 1981, pp. 1449–1480.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review,* 11:11–73, 1997.

[3] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47:69–77, 2004.

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

[5] S. Ekvall and D. Kragic. Grasp recognition for programming by demonstratio. In *Proc. IEEE/ RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, Canada, August 2005.

[6] S. Ekvall and D. Kragic. Receptive field cooccurrence histograms for object detection. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, Canada, August 2005.

[7] C. Gaskett, A. Ude, and G. Cheng. Hand-eye coordination through endpoint closed-loop and learned endpoint open-loop visual servo control. *International Journal of Humanoid Robotics,* 2(2):203–224, 2005.

[8] T. Inamura, Y. Nakamura, and I. Toshima. Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377, 2004.

[9] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Pattern Anal. Machine Intel., 20(11):154-1259, 1998.

[10] O. C. Jenkins and M. J. Matarić. Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion. *International Journal of Humanoid Robotics,* 1(2):237–288, 2004.

[11] D. Kragic, S. Crinier, D. Brunn, and H. I Christensen. Vision and tactile sensing for real world tasks. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1545–1550, Taipei, Taiwan, September 2003.

[12] M. J. Matarić, M. Williamson, J. Demiris, and A. Mohan. Behavior-based primitives for articulated control. In *Proc. Fifth Int. Conf. on Simulation of Adaptive Behavior*, pp. 165–170, Zurich, Switzerland, August 1998.

[13] A. T. Miller and P. K. Allen. Examples of 3D grasp quality computations. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1240–1246, Detroit, Michigan, USA, 1999.

[14] F. A. Mussa-Ivaldi and E. Bizzi. Motor learning through the combination of primitives. *Phil. Trans. R. Soc. Lond. B*, 355:1755–1769, 2000.

[15] R. Opara and F. Wörgötter. A fast and robust cluster update algorithm. *Neural Computation*, 10(6):1547–1566, 1998.

[16] N. Pugeault, E. Baseski, D. Kraft, F. Wörgötter, and N. Krüger. Extraction of multi–modal object representations in a robot vision system. In Robot Vision Workshop at the Int. Conf. on Computer Vision Theory and Applications (VISAPP), 2007.

[17] H. Ritter, T. Martinetz, and K. J. Schulten. Neural Computation and Self-Organizing Maps. *Addison Wesley*, New York, 1992.

[18] S. Schaal. Dynamic movement primitives – A framework for motor control in humans and humanoid robotics. In *Proc. 2nd Int. Symp. on Adaptive Motion of Anymals and Machines,* Kyoto, Japan, March 2003.

[19] G. Sun and B. Scassellati, Reaching through learned forward model. In *Proc. IEEE/RAS Intl. Conf. on Humanoid Robots*, Los Angeles, California, USA, November 2004.

[20] J. Tani. Self-organization of neuronal dynamical structures through sensory-motor experiences of Robots. In *IEEE-RAS Intl. Conf. on Humanoid Robots, Workshop on Synergistic Intelligence Dynamics*, Tsukuba, Japan, December 2005.

[21] J. Tegin and J. Wikander. A framework for grasp simulation and control in domestic environments. In *IFAC-Symp. on Mechatronic Syst.*, Heidelberg, Germany, September 2006.

[22] A. Ude and C. G. Atkeson. Probabilistic detection and tracking at high frame rates using affine warping. In *Proc. Int. Conf. Pattern Recognition,* Quebec City, Canada, August 2002.

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

[24] A. Ude, J. Moren, and G. Cheng. Visual attention and distributed processing of visual information for the control of humanoid robots. In preparation for publication in a book chapter, *Humanoid Robots*, ARS, 2007.

[25] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Autonomous Robots*, 12(1):59–72, 2002.

[26] J. Walter and H. Ritter. Rapid learning with parametrized self-organizing maps. *Neurocomputing*, 12:131–153, 1996.

Data Acqusition for Building Object Representations: Discerning the Manipulated Objects from the Background

Aleš Ude, Kai Welke, Joshua G. Hale, and Gordon Cheng

***Abstract*— Human vision is very effective at segmenting images into their meaningful constituents and focusing onto the perceptually relevant parts, but this property has proven to be extremely difficult to replicate by machine vision. Unlike static machine vision systems, robots are not passive and can actively observe and manipulate objects. In this paper we propose an approach to learning object representations by manipulation. Taking control of the object allows the robot to focus on the relevant part of the image, thus bypassing potential pitfalls of pure bottom-up attention and segmentation, which leads to a reliable extraction of representations for object recognition. Some experimental results showing object learning and recognition both on standard manipulators and on humanoid robots are presented.**

I. INTRODUCTION

Recent research has shown that object recognition on humanoid robots can be substantially improved by making use of foveated vision setups [1], [2]. Foveated visual systems described in [3]–[5] are useful because, firstly, they enable the robot to monitor and explore its surroundings in images with wider field of view and sparsely distributed pixels, thereby increasing the efficiency of the search process. Secondly, they make it possible to simultaneously extract additional information – once the object of interest appears in the fovea – from the foveal area, which has a denser pixel distribution and contains more detail [6]. While the proposed systems address many issues arising in object recognition on humanoid vision setups, little has been done in using the abilities of humanoid robots to learn complete representations for object recognition from scratch, i. e. before anything is known about the object. In our previous work [1], [6], we have shown how representations for recognition can be learned by actively observing objects manipulated by people using rough prior knowledge about objects' color texture and shape. In this paper we propose to go beyond the classic active vision paradigm that exploits the movement capabilities of the visual system and to make use of the robot's arms as well when learning object representations.

Aleš Ude is with the Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia, and Dept. of Humanoid Robotics and Computational Neuroscience, ATR Computational Neuroscience Laboratories, Kyoto, Japan. ales.ude@ijs.si

Kai Welke is with the Institute of Computer Science and Engineering, University of Karlstuhe, Germany. welke@ira.uka.de

Josh Hale is with the ICORP Computational Brain Project, Japan Science and Technology Agency and Dept. of Humanoid Robotics and Computational Neuroscience, ATR Computational Neuroscience Laboratories, Kyoto, Japan. josh@joshhale.com

Gordon Cheng is with the ICORP Computational Brain Project, Japan Science and Technology Agency and Dept. of Humanoid Robotics and Computational Neuroscience, ATR Computational Neuroscience Laboratories, Kyoto, Japan. gordon@atr.jp

Fig. 1. Simultaneous view of the object from the peripheral and foveal camera of a humanoid robot. The images from foveal camera are used for learning object representations.

Finding objects in images without any prior knowledge is a hard problem and is very difficult if not impossible to achieve in a purely bottom-up manner. Passive computer vision systems usually attempt to solve it by introducing top-down processes, which convey the knowledge about objects that assists the linking and grouping of early features into larger aggregations and sets. It is hoped that groupings of features are more likely to form constituent parts of an object than simple features. At some higher level of the processing hierarchy, the interplay between early features should culminate in the scene decomposition into its meaningful constituents (objects), which can then be used for further scene analysis and interpretation (recognition) purposes.

Unfortunately, it is not easy to formulate the top-down processes guiding the search for objects in a completely general way. We take the view that statistical learning approaches have hard time to learn how to generate such image decompositions from example images because the decomposition of images as done by people depends on the experience we gain when we interact with the environment. This information is not readily available in the images but rather comes from the experience of how our actions affect the external world. It is not clear how such information could be brought into the learning process on a passive system.

A humanoid robot, however, has the potential to explore its world using causality, by performing probing actions and learning from the response [7]. It has been shown that poking an object can be used to extract visual evidence for the boundary of the object, which is well suited for segmentation [8]. Our focus is after the initial, rough object segmentation and is in a sense complementary to this work. We study what the robot can do to facilitate the learning of object representations after it grasps the object. While it is surely possible to build object representations without manipulation, for how could we otherwise learn to recognize large objects like houses, we believe that manipulation can significantly

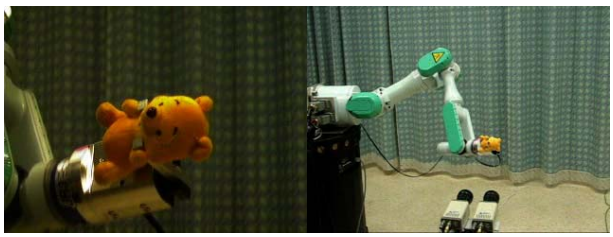


Fig. 2. The robot arm used in some of the experiments to manipulate the objects.

aid and speed up the learning process.

Since object recognition is an essential prerequisite for an autonomous robot, it has received a lot of attention in the past. Most of the currently successful object recognition systems are view-based and build suitable representations from snapshots of objects [9]–[13] . While early approaches used the collected patterns of objects without much preprocessing, most of the current works use local image features, e. g. scale invariant feature transform (SIFT keys) [12], Gabor jets [14], and others.

In this paper we propose a learning system that can be used to generate views of objects suitable for the above systems. We also propose our own classifier based on multi-class support vector machines that can be used to classify objects represented by collections of Gabor jets.

II. OUTLINE OF THE APPROACH AND DISCUSSION

We designed the following learning procedure to extract images of object appearance while the robot manipulates the object (see also Fig. 2):

1) The beginning of learning is initiated by the user, who places a new object into the robot's hand.
2) Once the robot holds the object, it moves its hand away from the view of the foveal cameras and starts learning the stationary background. Typically, we first learn the mean values of pixels in a significantly smoothed image for five seconds, followed by another five seconds of learning the variance in color at each pixel.
3) The robot moves its hand to the starting position for the observation of the object. Once it reaches the starting point, the procedure for estimating the position and extent of the manipulated object is initiated (see Section III).
4) The robot moves the object along the predefined straight-line trajectory, attempting to keep the object within the fovea. Only the arm and hand degrees of freedom are used in this stage. The position and the extent of the object in each of the captured images along the trajectory are estimated. After the hand returns to the starting point, the optimal position on the trajectory (with respect to the estimated positions and sizes) for learning object appearance is determined.
5) The robot moves the object into the optimal position for learning as determined in the previous step. After reaching the desired hand position and orientation, it starts exercising the hand along the two degrees of freedom that cause the object to rotate in depth. The images of object appearance are collected while manipulating the object. This phase finishes once the hand covers the prespecified range of motion for the two degrees of freedom.
6) The object is placed into the robot hand again at a different configuration and the procedure is repeated.
7) Once all of the object have been placed at all relevant configurations and all appearance images have been collected, a classifier for object recognition is learned by using a method based on nonlinear multi-class support vector machines, which is described in Section IV.

The above outline needs several clarifications. While it is certainly among of our future goals that the robot would pick up the object by itself, this has not been implemented yet. We envision a procedure similar to [8] for the automatic generation of hypotheses about the existence and position of the object. This needs to be followed by grasping of the unknown object, which is a difficult task in its own right. Instead of the user, who places the object into the robot hand at different configurations, in the automatic mode the robot would need to re-grasp the object by itself. While difficult this opens a new possibility for associating the object postures with the appearance images because the robot can use proprioceptive information to estimate each pose of the object with respect to the initial pose. Standard view-based approaches for pose estimation can then be used [15].

Background models like the one we learn in step 2) are subject to frequent changes due to factors like object movements and variations in lighting conditions. This is, however, of minor concern here because having full control of the object, the robot can ensure that nothing else moves in the environment during learning. In addition, the learned background models are short-lived and are learned anew every time the object is re-grasped.

The test movement towards and away from the eyes described in step 4) is necessary to place the object in a suitable posture for learning object appearances. This posture should be close enough to the eyes so that the object does not appear too small in the image, but also far enough so that its projection does not fall outside of the image. Our criterion is that the object should appear as big as possible in the foveal view while its boundary needs to be at least 40 pixels away from the image boundary, where the image size was 320×240 pixels. The ideal position based on this criteria is determined by moving the object along the straight line towards the center of the foveal image.

On an accurately calibrated humanoid robot, Cartesian straight line trajectories can be designed easily. For the cases when accurate models are not available, we developed an automatic procedure based on a rough open loop and more accurate closed loop control system described in [16]. To get the objects into the center of the fovea based on information from the peripheral images, the system is guided by foveation

principles governing the relationship between the foveal and peripheral views [6]. The developed technique allows us to start from a rough straight-line trajectory towards the fovea as designed by the open-loop control system, which can then be refined based on the closed-loop control system. This procedure generates a number of postures on the trajectory, which are then interpolated to generate an accurate joint space trajectory resulting in a straight-line movement in the Cartesian space.

The manipulation procedure designed to extract object views for training purposes can also be used to determine the ideal position for recognition. Snapshots of the object are captured by executing steps 1) to 5). This approach facilitates the invariance against scaling because the object is always viewed from about the same distance. While it is still necessary to carry out the warping step that results in normalized snapshots such as the ones shown in Fig. 4, various digitalization artifacts that could be introduced by doing such a mapping on object images at different scales are greatly reduced. By monitoring the results of the classifier on snapshots of the object taken from slightly different viewing angles, we can also improve the reliability of the classification because classification results are usually stable only when they are correct.

III. DISCERNING THE OBJECT

At the heart of our system is the ability to discern the objects from the image while it is manipulated by the robot. To achieve this goal, we need to model the following image processes:

- the unknown object (denoted by process Θ_{o}),
- the background (Θ_{b}),
- the hand (Θ_{h}), and
- the outlier process ((Θ_{t}), modeling any unexpected event in the scene.

We model the color intensity of each pixel in the stationary background by Gaussian processes $\Theta_{\mathrm{b}} = \{\overline{I}\boldsymbol{u}, \overline{\boldsymbol{\Sigma}}\boldsymbol{u}\}\boldsymbol{u}$, which are characterized by means $\overline{I}\boldsymbol{u}$ and covariance matrices $\overline{\boldsymbol{\Sigma}}\boldsymbol{u}$ at each pixel $\boldsymbol{u}$ with the associated probability distributions

$$\mathrm{p}(I\boldsymbol{u}, \boldsymbol{u}|\Theta_{\mathrm{b}}) = \frac{1}{2\pi\sqrt{\det(\overline{\boldsymbol{\Sigma}}\boldsymbol{u})}} \cdot \exp\left(-\frac{1}{2}(I\boldsymbol{u} - \overline{I}\boldsymbol{u})^T \overline{\boldsymbol{\Sigma}}\boldsymbol{u}^{-1}(I\boldsymbol{u} - \overline{I}\boldsymbol{u})\right). \quad (1)$$

To obtain a certain degree of robustness against the brightness changes, we characterize the color intensities by either hue and saturation or by the normalized RGB values

$$r = \frac{R}{R+B+G},\ g = \frac{G}{R+B+G},\ b = \frac{B}{R+B+G}.$$

Since these three values are not independent, we use in our system only normalized red and normalized green, which makes color a two dimensional value for both color spaces. The means and the covariances are learnt by gathering statistics of the background pixels just before the robot brings the object into the fovea. We did not observe big differences when using either of the two color spaces, but more experiments are needed to confirm this point.

Even though the hand position in the image could be calculated using proprioceptive information, this information is not sufficient because we cannot know in advance which part of the hand is visible and which part is covered by the manipulated object. We thus need to model the appearance of the hand in the image. For the modelling of the hand appearance, we experimented with standard approaches from the object tracking theory such as color histograms [17] and Gaussian (mixture) models [18]. Unlike in tracking, we are not really interested in computing the hand position but only in estimating the probability that a particular pixel belongs to the hand. Both color histograms and Gaussian mixture models offer this ability. Gaussian mixture models are defined as follows

$$\mathrm{p}(I\boldsymbol{u}|\Theta_{\mathrm{h}}) = \sum_{k=1}^{K} \frac{\omega_k}{2\pi\sqrt{\det(\boldsymbol{\Sigma}_k)}} \cdot \exp\left(-\frac{1}{2}(I\boldsymbol{u} - \overline{I_k})^T \boldsymbol{\Sigma}_k^{-1}(I\boldsymbol{u} - \overline{I_k})\right). \quad (2)$$

In our experiments the hand could be characterized by one color and we could thus use unimodal Gaussians ($K = 1$) to model the appearance of the hand.

While motion cues could certainly help to extract the object from the hand and background, such cues alone are not sufficient for the extraction of the object appearance. When the robot holds the object, the object motion is the same as the motion of the robot hand. We can thus not distinguish between the object and the hand based on the motion cue only. In addition, motion estimates are normally calculated by differential methods which makes them relatively noisy. Hence motion should be used only as support for other cues and not as the sole feature for segmentation.

Since we have no prior knowledge about the object, we obviously cannot model its appearance, which is actually what we want to learn. The open-loop trajectory that we use to manipulate the object is, however, well defined and we know approximately where the object is in the image. We can thus model the probability that an image pixel falls within the extent of the object by using the mean value $\overline{\boldsymbol{u}}$ and the covariance $\overline{\boldsymbol{\Sigma}}$ of pixels belonging to the object in the previous step. This results in the following distribution

$$\mathrm{p}(\boldsymbol{u}|\Theta_{\mathrm{o}}) = \frac{1}{2\pi\sqrt{\det(\overline{\boldsymbol{\Sigma}})}} \exp\left(-\frac{1}{2}(\boldsymbol{u} - \overline{\boldsymbol{u}})^T \overline{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{u} - \overline{\boldsymbol{u}})\right). \quad (3)$$

Since the robot attempts to move the object along the straight line going through the projection center and the center of the image, the object's position is always close to the image center and we can initialize the appearance extraction by assuming that the object is centered in the image with an initially small extent.

Fig. 1 shows that foveal images sometimes contain other parts of the arm besides the hand. Having no prior infor-

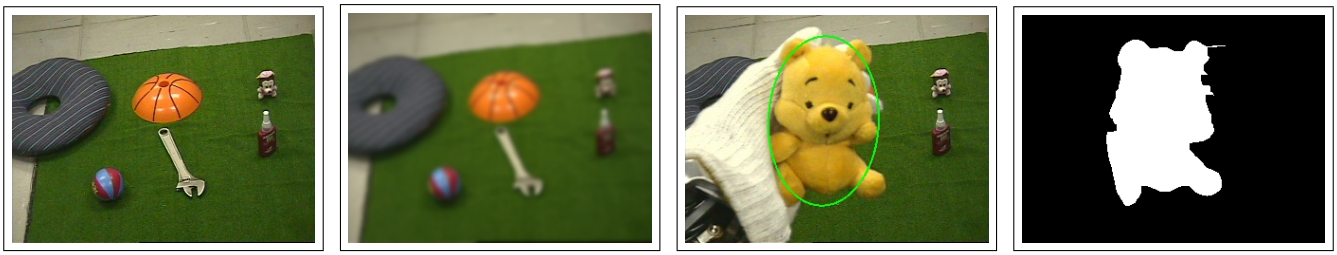


Fig. 3. Example for the extraction of object appearance. From left to right we have images showing one of the images used for background learning, the smoothed version of this image that we use for the collection of background statistics, the estimated extent of the object in the image while being manipulated by the robot, and the binary image containing the largest connected component of object pixels after thresholding probabilities $\mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b})$ and applying the morphological operation close. The image within the enclosing ellipse is used for learning object representations. The binary image is shown only for presentations purposes and is never used in the calculations. Additional objects were placed in the scene to show that our system does not require simple backgrounds.

mation about the outlook of the arm and other unexpected objects that might appear in the scene, we model such events in the image by an outlier process, which is assigned a small, constant probability $\mathrm{P}(\Theta_\mathrm{t})$ regardless of the position of the pixel in the image or color intensity value at this pixel. The interaction between this process and the object process Θ_o occurs in such a way that an area with a texture different from the background and the hand will be classified as an object of interest if it is close to the expected object position and outlier otherwise (see Eq. (7).

As for the arm, the part of the image containing it can be excluded from calculations using proprioceptive information. On a dynamic humanoid robot like CB proprioceptive information provides only a rough estimate for the location of the arm in the image. It is nevertheless sufficient to exclude from the calculations most of the image containing the arm. Our experiments showed that combined with the outlier process, this is sufficient to filter out the arm when estimating the extent of the object of interest in the image.

Assuming that every pixel in the image stems from one of the mutually independent processes $\boldsymbol{\Theta} = \{\Theta_\mathrm{b}, \Theta_\mathrm{h}, \Theta_\mathrm{o}, \Theta_\mathrm{t}\}$ (closed-world assumption), we can write the probability that color $I_{\boldsymbol{u}}$ was observed at location $\boldsymbol{u}$ using the total probability law

$$\begin{aligned}\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\boldsymbol{\Theta}) &= \omega_\mathrm{b}\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_\mathrm{b}) + \omega_\mathrm{h}\mathrm{P}(I_{\boldsymbol{u}}|\Theta_\mathrm{h}) + \\ &\quad \omega_\mathrm{o}\mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{o}) + \omega_\mathrm{t}\mathrm{P}(\Theta_\mathrm{t}),\end{aligned} \tag{4}$$

where ω_x are the prior (mixture) probabilities to observe the processes Θ_x and $\omega_\mathrm{b} + \omega_\mathrm{h} + \omega_\mathrm{o} + \omega_\mathrm{t} = 1$.

We need to estimate the current position of the unknown object and its extent, which will provide us with an appearance image for learning. This can be achieved by maximizing the probability of observing image $\boldsymbol{I}$ given processes $\boldsymbol{\Theta} = \{\Theta_\mathrm{b}, \Theta_\mathrm{h}, \Theta_\mathrm{o}, \Theta_\mathrm{t}\}$. Neglecting the correlation of assigning neighboring pixels to processes, we can evaluate the overall probability of observing image $\boldsymbol{I}$ as follows

$$\mathrm{P}(\boldsymbol{I}) = \mathrm{P}(\boldsymbol{I}|\boldsymbol{\Theta}) = \prod_{\boldsymbol{u}} \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\boldsymbol{\Theta}). \tag{5}$$

Since the background and the color distribution of the hand are assumed stationary, we can maximize (5) with respect to the position $\overline{\boldsymbol{u}}$ of the object, the covariance $\overline{\boldsymbol{\Sigma}}$ of pixels belonging to the object, and mixture probabilities ω_b, ω_h, ω_o, and ω_t. Instead of maximizing (5), it is easier to minimize the negative log likelihood

$$\mathrm{L}(\boldsymbol{\Theta}, \boldsymbol{\omega}) = -\log(\mathrm{P}(\boldsymbol{I}|\boldsymbol{\Theta})) = -\sum_{\boldsymbol{u}} \log\left(\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u}|\boldsymbol{\Theta})\right). \tag{6}$$

where $\boldsymbol{\omega} = (\omega_\mathrm{b}, \omega_\mathrm{h}, \omega_\mathrm{o}, \omega_\mathrm{t})$. Using the Lagrange multipliers theory, it is possible to show that the above log likelihood can be minimized by an EM algorithm. Writing

$$\mathrm{P}(\boldsymbol{I}_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_x) = \frac{\omega_x \mathrm{p}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_x)}{\sum\limits_{y\in\{\mathrm{o,h,b,t}\}} \omega_y \mathrm{p}(I_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_y)} \tag{7}$$

where $x = \mathrm{o},\ \mathrm{h},\ \mathrm{b},\ \mathrm{t}$, the EM-algorithm consists of the expectation step, in which pixel probabilities (7) are estimated, and the maximization step, in which the probabilities $\mathrm{P}(\boldsymbol{I}_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_\mathrm{b}) = \mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b})$ are used to estimate the mean and the covariance of the object pixels

$$\overline{\boldsymbol{u}} = \frac{1}{\sum_{\boldsymbol{u}} \mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b})} \sum_{\boldsymbol{u}} \mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b})\boldsymbol{u}, \tag{8}$$

$$\overline{\boldsymbol{\Sigma}} = \frac{1}{\sum_{\boldsymbol{u}} \mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b})} \sum_{\boldsymbol{u}} \mathrm{P}(\boldsymbol{u}|\Theta_\mathrm{b}) (\boldsymbol{u} - \overline{\boldsymbol{u}}) (\boldsymbol{u} - \overline{\boldsymbol{u}})^T. \tag{9}$$

Note that probabilities $\mathrm{P}(\boldsymbol{I}_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_\mathrm{b})$ and $\mathrm{P}(\boldsymbol{I}_{\boldsymbol{u}}|\Theta_\mathrm{h})$ remain constant throughout the EM process and thus need to be estimated only once for each image. This helped us to implement the extraction of the object appearance at video rate, i. e. at 30 Hz. The mixture probabilities can either be assumed to be constant or we can estimate them as part of the EM-process

$$\omega_x = \frac{1}{n} \sum_{\boldsymbol{u}} \mathrm{P}(\boldsymbol{I}_{\boldsymbol{u}}, \boldsymbol{u}|\Theta_x), \tag{10}$$

where n is the number of pixels and $x = \mathrm{o},\ \mathrm{h},\ \mathrm{b},\ \mathrm{t}$.

IV. LEARNING OBJECT REPRESENTATIONS

It remains to show how we learn the classifier for recognition using the output of the object manipulation procedure in conjunction with the technique for the extraction of object appearance described in Section III. After estimating the enclosing ellipse, the image is warped onto a window of

Fig. 4. Images of four objects used in the experiments after warping. Such images are used as input to Gabor jet calculations and SVM training.

constant size. This ensures invariance against scaling and planar rotations and also provides images of standard size, which can be compared to each other. Fig. 4 shows the warped images of four objects used in our experiments.

To ensure maximum classification performance the data fed into general classifiers such as SVMs needs some kind of preprocessing, which is especially important for high-dimensional input data. Most modern view-based approaches characterize the views by ensembles of local features. We use complex Gabor kernels to identify local structure in the images, which are first transformed to grayscale. Currently, color is not used in our system for recognition, although features like color histograms could certainly be beneficial [2]. Gabor kernels are given by

$$\begin{aligned}\boldsymbol{\Phi}_{\mu,\nu}(\boldsymbol{x}) &= \frac{\|\boldsymbol{k}_{\mu,\nu}\|^2}{\sigma^2}\cdot\exp\left(-\frac{\|\boldsymbol{k}_{\mu,\nu}\|^2\|\boldsymbol{x}\|^2}{2\sigma^2}\right)\cdot \\ &\left(\exp\left(i\boldsymbol{k}_{\mu,\nu}^T\boldsymbol{x}\right)-\exp\left(-\frac{\sigma^2}{2}\right)\right), \quad (11)\end{aligned}$$

where $\boldsymbol{k}_{\mu,\nu} = k_\nu[\cos(\phi_\mu),\ sin(\phi_\mu)]^T$. A Gabor jet at pixel $\boldsymbol{x}$ is defined as a set of complex coefficients $\{J_j^{\boldsymbol{x}}\}$ obtained by convolving the image with a number of Gabor kernels at this pixel. Gabor kernels are selected so that they sample a number of different wavelengths k_ν and orientations ϕ_μ. Wiskott et al. [14] proposed to use $k_\nu = 2^{-\frac{\nu+2}{2}}$, $\nu = 0,\ldots,4$, and $\phi_\mu = \mu\frac{\pi}{8}$, $\mu = 0,\ldots,7$, but this depends both on the size of the incoming images and the image structure. They showed that the similarity between the jets can be measured by

$$\mathrm{S}\left(\{J_i^{\boldsymbol{x}}\},\{J_i^{\boldsymbol{y}}\}\right) = \frac{\boldsymbol{a}_{\boldsymbol{x}}^T * \boldsymbol{a}_{\boldsymbol{y}}}{\|\boldsymbol{a}_{\boldsymbol{x}}\|\|\boldsymbol{a}_{\boldsymbol{y}}\|}, \quad (12)$$

where $\boldsymbol{a}_{\boldsymbol{x}} = [|J_1^{\boldsymbol{x}}|,\ldots,|J_s^{\boldsymbol{x}}|]^T$ and s is the number of complex Gabor kernels. This is based on the fact that the magnitudes of complex coefficients vary slowly with the position of the jet in the image.

In our system, feature vectors are built by sampling Gabor jets on a regular grid of pixels $\boldsymbol{X}_G$. At each grid point we calculate the Gabor jet and add it to the feature vector. Naturally, the grid points need to be parsed in the same order in every image. The grid size used in our experiments was 6×6, the warped image size was 160×120 with pixels outside the enclosing ellipse excluded, and the dimension of each Gabor jet was 40, which resulted in feature vectors of dimension 16080. These feature vectors were supplied to the SVM for training.

A. Nonlinear Multi-Class Support Vector Machines

Now we turn to the problem of finding a suitable classifier for object recognition using nonlinear multi-class support vector machines. Classification based on nonlinear multi-class SVMs [19] is carried out using the following decision function

$$\boldsymbol{H}(\boldsymbol{x}) = \arg\max_{r\in\boldsymbol{\Omega}}\left\{\sum_{i=1}^{m}\tau_{i,r}\mathrm{K}(\boldsymbol{x}_i,\boldsymbol{x})+b_r\right\}. \quad (13)$$

Here $\boldsymbol{x}$ is the input feature vector to be classified (in our case a collection of Gabor jets), $\boldsymbol{x}_i$ are the feature vectors supplied to the SVM training, $\tau_{i,r}$, b_r are the values estimated by SVM training, and $\Omega = \{1,\ldots,N\}$ are the class identities (objects in our case). The feature vectors $\boldsymbol{x}_i$ with $\tau_{i,r} \neq 0$ are called the support vectors. The SVM training consists of solving a quadratic optimization problem which convergence is guaranteed for all kernel functions K that fulfill the Mercer's theorem [20].

The similarity measure for Gabor jets (12) provides a good motivation for the design of a kernel function for the classification of feature vectors consisting of Gabor jets. Let $\boldsymbol{X}_G$ be the set of all grid points within two normalized images on which Gabor jets are calculated and let $J_{\boldsymbol{X}_G}$ and $L_{\boldsymbol{X}_G}$ be the Gabor jets calculated in two different images, but on the same grid points. A suitable kernel function can be defined as follows

$$\mathrm{K}_G(J_{\boldsymbol{X}_G},L_{\boldsymbol{X}_G}) = \exp\left(-\rho\frac{1}{M}\sum_{\boldsymbol{x}\in\boldsymbol{X}_G}\left(1-\frac{\boldsymbol{a}_{\boldsymbol{x}}^T * \boldsymbol{b}_{\boldsymbol{x}}^T}{\|\boldsymbol{a}_{\boldsymbol{x}}\|\|\boldsymbol{b}_{\boldsymbol{x}}\|}\right)\right), \quad (14)$$

where M is the number of grid points in $\boldsymbol{X}_G$. This function satisfies the Mercer's condition [20] and can thus be used for support vector learning. Parameter ρ needs to be supplied experimentally.

V. EXPERIMENTAL RESULTS

In our experiments we tested how effective is the manipulation of objects as outlined in Section II combined with the Bayesian technique of Section III at extracting views for training and recognition. Based on what we believe are reasonable assumptions about how the robot interacts with its environment, we were able to collect the object views shown in Fig. 4 without any prior knowledge about the objects. The procedure for discerning the object from the rest of the scene has proven to be reliable as long as the assumptions made by the Bayesian approach were fulfilled.

To prove that the proposed approach can indeed be used for learning object representations, we compared it to the classification results achieved when known color textures were used to discern the object from the rest of the image [1]. To train the SVM, we collected 104 views of 14 different objects. The appearance images of four of them were extracted using the proposed approach, while the images of the rest were collected by applying models of color texture for segmentation. For training of a fully rotationally and

TABLE I
CLASSIFICATION RESULTS

	Correct	Errors	Recognition rate
Full library	7307	421	94.6 %
Objects detected by color	4897	303	94.2 %
Objects detected by manipulation	2410	118	95.3 %

scale invariant classifier on a library of 14 objects, we thus employed 1456 feature vectors of dimension 16080. We use the implementation of nonlinear multi-class SVMs by [21], which allows for the user-defined kernels. Hence we were able to make use of the specially designed kernel of Section IV-A.

For testing we collected another 7728 appearance images of objects from the library. Results in Tab. I prove that the views collected by the proposed approach are just as usable as the views that we collected using prior color texture models. The recognition results with the proposed approach were even a bit better, although this was caused by a relatively bad classification rate for one the object for which we used color texture segmentation to extract the views. Excluding this object, the recognition rates were almost identical.

VI. SUMMARY

The main result of this paper is the procedure for learning complete object representations for recognition by a humanoid robot without any prior knowledge about objects and without manual tinkering with the images. To our knowledge CB is the first humanoid that can collect the views fully automatically provided that it can grasp the objects. Our experiments showed that the built models are fully scale and rotationally invariant in 3-D and that we achieve comparable recognition rates on the proposed system as on the earlier system that used prior knowledge about the object's color texture to discern its image from the rest of the scene.

We also developed a new kernel for the classification of views represented by Gabor jets, which allowed us to classify the images more reliably, especially when lighting conditions in training and recognition phase differ. Note also that the proposed approach for collecting the views is fully general and is not limited to the developed classification technique. We could as well apply it to other popular approaches such as SIFT keys + Hough transform proposed in [12].

There are many issues that could be considered to improve the system in the future. One of them is using motion cues to make the Bayesian approach of Section III more robust. Another interesting issue to consider is the use of proprioceptive information to organize the training views by orientation. It has been shown that dynamic information can be useful for recognition [22]. The robot taking control of the object can provide the necessary input for the view-based dynamic object recognition. In addition, such information can be used to estimate the orientation of an object after recognition. We are currently working on these issues.

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

REFERENCES

[1] A. Ude, C. G. Atkeson, and G. Cheng, "Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, Nevada, 2003, pp. 2173–2178.

[2] M. Björkman and D. Kragić, "Combination of foveal and peripheral vision for object recognition and pose estimation," in *Proc. IEEE Conf. Robotics and Automation*, New Orleans, Louisiana, 2004, pp. 5135–5140.

[3] B. Scassellati, "Eye finding via face detection for a foveated, active vision system," in *Proc. Fifteenth Nat. Conf. Artifficial Intelligence (AAAI '98)*, Madison, Wisconsin, 1998, pp. 969–976.

[4] C. G. Atkeson, J. G. 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 Intelligent Systems*, vol. 15, no. 4, pp. 46–56, 2000.

[5] G. Sandini and G. Metta, *Sensors and Sensing in Biology and Engineering*. Wien-New York: Springer-Verlag, 2003, ch. Retina-like sensors: motivations, technology and applications.

[6] A. Ude, C. Gaskett, and G. Cheng, "Foveated vision systems with two cameras per eye," in *Proc. IEEE Int. Conf. Robotics and Automation*, Orlando, Florida, 2006, pp. 3457–3462.

[7] P. Fitzpatrick and G. Metta, "Grounding vision through experimental manipulation," *Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences*, vol. 361, no. 1811, pp. 2165–2185, 2003.

[8] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, Nevada, 2003, pp. 2161–2166.

[9] T. Poggio and S. Edelman, "A network that learns to recognize three-dimensional objects," *Nature*, vol. 343, pp. 263–266, 1990.

[10] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[11] B. Schiele and J. L. Crowley, "Recognition without correpsondence using multidimensional receptive field histograms," *Int. J. Computer Vision*, vol. 36, no. 1, pp. 31–52, 2000.

[12] D. G. Lowe, "Local feature view clustering for 3D object recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Kauai, Hawaii, 2001, pp. 682–688.

[13] C. Wallraven, A. Schwaninger, and H. H. Bülthoff, "Learning from humans: Computational modeling of face recognition," *Network: Computation in Neural Systems*, vol. 16, no. 4, pp. 401–418, 2005.

[14] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 775–779, 1997.

[15] S. Srinivasan and K. L. Boyer, "Head pose estimation using view based eigenspaces," in *Proc. 16th Int. Conf. Pattern Recognition, vol. 4*, Quebec, Canada, 2002, pp. 302–305.

[16] C. Gaskett, A. Ude, and G. Cheng, "Hand-eye coordination through endpoint closed-loop and learned endpoint open-loop visual servo control," *International Journal of Humanoid Robotics*, vol. 2, no. 2, pp. 203–224, 2005.

[17] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 564–577, 2003.

[18] S. J. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing*, vol. 17, pp. 225–231, 1999.

[19] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[20] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[21] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999.

[22] H. H. Bülthoff, C. Wallraven, and A. Graf, "View-based dynamic object recognition based on human perception," in *Proc. Int. Conf. Pattern Recognition, vol. III*, Québec City, Canada, August 2002, pp. 768 – 776.

Foveated Vision and Object Recognition on a Humanoid Robot

Aleš Ude[1,2] Chris Gaskett[2] Gordon Cheng[2,3] Mitsuo Kawato[2,3]

[1]Jožef Stefan Institute
Dept. of Automatics, Biocyb. and Robotics
Jamova 39, 1000 Ljubljana, Slovenia

[2]ATR Computational Neuroscience Lab.
Dept. of Humanoid Robotics and Computational Neuroscience
2-2-2 Hikaridai, Seika-cho, Kyoto, Japan

[3]Japan Science and Technology Agency
ICORP Computational Brain Project
2-2-2 Hikaridai, Seika-cho, Kyoto, Japan

***Abstract*— In this paper we present a complete, biologically motivated system for 3-D object recognition on a humanoid robot with foveated vision. It includes an eye and body control scheme, which enables the robot to maintain the object view in the fovea, and image processing techniques necessary to foveate on the objects and to recognize them. We show the degree of precision that can be achieved by a foveated setup that uses two cameras per eye to mimic the foveated structure of a human eye. We demonstrate that tracking results can be utilized to reduce the amount of training images necessary to learn object representations from foveal views. The proposed recognition approach is view-based and can learn truly three-dimensional object representations. It is build around a classifier based on multi-class support vector machines. The objects can be represented by various features such as Gabor jets and color histograms. To increase the classification rates, we designed a special kernel function that can incorporate the segmentation probabilities computed by the tracker.**

The proposed system was fully implemented and runs in real-time, which is essential for meaningful interaction with a humanoid robot.

I. Introduction

A robot vision system can be called humanoid if it firstly possesses an oculomotor system similar to human eyes, and secondly if it is capable of simultaneously acquiring and processing images of varying resolution. Designers of a number of humanoid robots attempted to mimic the foveated structure of the human eye. Foveation is useful because, firstly, it enables the robot to monitor and explore its surroundings in images of low resolution, thereby increasing the efficiency of the search process, and secondly, it makes it possible to simultaneously extract additional information – once the area of interest is determined – from higher resolution foveal images that contain more detail. There are several visual tasks that can benefit from foveated vision. One of the most prominent among them is object recognition. General object recognition on a humanoid robot is difficult because it requires the robot to detect objects in dynamic environments and to control the eye gaze to get the objects into the fovea and to keep them there. Once these tasks are accomplished, the robot can determine the identity of the object by processing foveal views.

Approaches proposed to mimic the foveated structure of biological vision systems include the use of two cameras per eye [1]–[4] (Cog, DB, Infanoid, Kismet, respectively), i. e. a narrow-angle foveal camera and a wide-angle camera for peripheral vision; lenses with space-variant resolution [5] (humanoid head ESCHeR), i. e. a very high definition area in the fovea and a coarse resolution in the periphery; and space-variant log-polar sensors with retina-like distribution of photo-receptors [6] (Babybot). It is also possible to implement log-polar sensors by transforming standard images into log-polar ones [7], but this approach requires the use of high definition cameras to get the benefit of varying resolution. Systems with zoom lenses have some of the advantages of foveated vision, but cannot simultaneously acquire wide angle and high resolution images.

Fig. 1. Humanoid head with foveal cameras above the peripheral cameras. The peripheral and foveal cameras are rigidly connected with parallel optical axes. The motor system of each eye consists of two independent degrees of freedom.

Our work follows the first approach (see Fig. 1) and explores the advantage of foveated vision for object recognition over current approaches which use equal resolution across the visual field. While we acknowledge that log-polar sensors are a closer match to biology, we note that using two cameras per eye can be advantageous because cameras with standard CCD/CMOS chips can be utilized. This makes it possible to equip a humanoid robot with miniature cameras (lipstick size and smaller), which facilitates the mechanical design of the eye and improves its motion capabilities.

There has been extensive work both on controlling the eyes to steer the gaze over salient areas and on processing

information acquired by a humanoid vision system. Studies on oculomotor control in a humanoid robot included vestibulo-ocular and optokinetic reflex, smooth pursuit, saccades, and vergence control [5], [8]–[10]. Other researchers considered head and arm movements for reaching [11]–[13] and mimicking behaviors [14], [15]. On the image processing side, researchers studied humanoid vision for visual attention [4], [16], segmentation [17], and tracking [5], [18], [19]. The analysis of this work shows that the utilization of foveation for the processing of visual information was not of major concern in most of these papers. Even though any system implemented on log-polar cameras or space variant lenses implies the processing of foveal information, researchers who worked with such systems appeared to have been more concerned with oculomotor control issues than with how to exploit the benefits of higher resolution in the fovea to solve specific visual problems. One notable exception is the early work of Scasselatti [1], in which foveal images were used to detect the eyes of people whose faces were first located by peripheral vision. This is a very specialized problem and the authors relied on the underlying behavioral context to simplify the computations. In our recent work we demonstrated how foveation [20] can be used for object recognition. Our initial system employed LoG (Laplacian of the Gaussian) filters at a single, manually selected scale and principal component analysis to represent objects. The nearest neighbor approach was used to identify objects from a database which was acquired by training. A similar approach using foveation and PCA – extended by the application of multi-modal cues for recognition – was later described in [21].

Starting from our previous work, the goal of this paper is to propose a comprehensive framework for object recognition on a humanoid with foveated vision. It is important for the humanoid robotics community to thoroughly study the design of essential visual routines such as object recognition in humanoids as opposed to being quickly satisfied with easily implementable basic solutions. Otherwise higher-level cognitive tasks that build upon such processes could be unnecessarily limited by poor performance of lower-level visual routines. Our system consists of three main integrated modules:

- attention and object tracking (peripheral vision);
- object recognition (foveal vision);
- oculomotor control (including the supportive head and body movements).

To facilitate later analysis, we start by briefly presenting our approach to visual attention and tracking. Section III describes what kind of precision can be expected from a foveated setup with two cameras per eye. It also presents a motor control framework that enables the robot to bring the objects of interest over the fovea. The section concludes with experimental results showing that the accuracy of the proposed motor control scheme is sufficient to make use of foveal vision for recognition when observing objects manipulated by people interacting with the robot. We continue by introducing a biologically motivated view-based object representation scheme, followed by a classification algorithm based on multi-class support vector machines. We present a specially designed kernel function that can accommodate multiple cues and can make use of segmentation (tracking) probabilities. These probabilities are used to reduce the amount of distractors from the background that can negatively influence the classification results and also to reduce the computing time. Finally, we test the performance of the recognizer as it pertains to the domain of object recognition for person-robot interaction.

II. Probabilistic Tracking and Attention

To direct its eyes towards objects and to classify them, a robot must be able to identify areas of interest and to track the interesting regions. Tracking and visual attention are not of major concern for this paper, but since the results of these subsystems are needed for foveated objected recognition, we here briefly describe the tracking approach as it is necessary for the understanding of the operation of the complete system.

We represent the observed environment by a number of random processes, each process representing one of the currently observed objects. Our approach uses color and shape properties to evaluate the probability that a pixel was generated by one of these processes. What is important here for object recognition is that we assume that 2-D shape of the tracked objects is roughly ellipsoidal and can be approximated by the center of the object's image $\boldsymbol{x}$ and by the covariance matrix $\boldsymbol{\Sigma}$ of pixels contained in it.

Given the appropriate probability distributions for objects and background, we can use an expectation-maximization (EM) algorithm to estimate the most probable location and shape of the object in the image. The EM-algorithm alternatingly estimates the posterior probabilities (expectation), and the shape parameters and the object location (maximization). The color distributions are learned off-line and are kept constant in the tracking phase. We omit the details of the real-time implementation and relate the reader to our earlier papers [18], [22]. What is important for foveated recognition is that the tracker not only estimates the object locations but also its 2-D shape and orientation as well as the posterior pixel probabilities evaluating the confidence that a pixel belongs to the object.

One of the features utilized by the proposed tracker is color. To make sense of the recognition task, we later consider recognition of objects having same or similar color. This is, however, solely due to the limitations of the segmentation and tracking techniques. On the other hand, the recognition strategy is fully general. The tracker and the following oculomotor behaviors are initialized based on the results of a distributed visual attention system, which also takes over if the object is lost during pursuit.

III. Maintaining the Foveal Views of Objects

The results of the tracker enable the robot to direct its eyes towards potential objects of interest. The main task of the control system is to place a salient region over the field of view of both foveal cameras so that further analysis and eventually

Fig. 2. Simultaneous views from the peripheral and foveal cameras. The high resolution of object image and better localization makes foveal images suitable for recognition (right), while the wide field of view from the peripheral camera is suitable for smooth pursuit (left).

object recognition can be accomplished. Although the focus of the task is to bring an object into the center of the fovea, the control system uses the view from peripheral cameras as the basis for control. Data from peripheral images is more reliable for tracking and pursuit because objects can easily be lost from the foveal views (see Fig. 2).

Two issues need to be considered when analyzing the foveation setup with two cameras:

1) Given a 3-D point that projects onto the center of the foveal image, where will the point be projected onto the peripheral image? This will be the ideal position in the periphery for foveation.
2) If a 3-D point projects onto the peripheral image away from the ideal position described above, how far is the projection of the point from the center of the foveal image?

A. Camera Model

For the theoretical analysis, we model both cameras by a standard pinhole camera model. We denote a 3-D point by $\boldsymbol{M} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ and a 2-D point by $\boldsymbol{m} = \begin{bmatrix} x & y \end{bmatrix}^T$. Let $\tilde{\boldsymbol{M}} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ and $\tilde{\boldsymbol{m}} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ be the homogeneous coordinates of $\boldsymbol{M}$ and $\boldsymbol{m}$, respectively. The relationship between a 3-D point $\boldsymbol{M}$ and its projection $\boldsymbol{m}$ is then given by [23]

$$s\tilde{\boldsymbol{m}} = \boldsymbol{A} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \tilde{\boldsymbol{M}}, \tag{1}$$

where s is an arbitrary scale factor, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the extrinsic parameters denoting the rotation and translation that relate the world coordinate system to the camera coordinate system and $\boldsymbol{A}$ is the intrinsic matrix

$$\boldsymbol{A} = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

α and β are the scale factors, γ is the parameter describing the skewness of the two image axes, and (x_0, y_0) is the principal point.

In the following we assume without loss of generality that the origin of the image coordinate system coincides with the principal point (x_0, y_0), thus $x_0 = y_0 = 0$. Note that on a real camera the principal point does not coincide with the image center in pixel coordinates exactly. However, since the distortion effects are smallest around the principal point and since making this assumption significantly simplifies the equations, it makes sense to attempt to bring the point of interest to the position that projects onto the principal point of the foveal camera and not to the precise image center. Taking a standard video camera producing 640 x 480 images, the distance of the principal point from the image center in pixel coordinates is usually less than 10 pixels.

The pinhole camera model (1) does not consider the effects of lens distortion. Such an assumption is justified for foveal cameras, which are equipped with lenses with relatively long focal lengths that normally do not exhibit noticeable distortion effects. This is especially true because the distortion function is usually dominated by radial components [23], [24]. Hence the distortion effects are larger at the edges than in the center of an image and therefore have only limited effects on foveation. Conversely, to achieve wide field of view, peripheral cameras need to have lenses with shorter focal lengths. Cameras with such lenses often produce significantly distorted images. However, the distortion can be corrected in a preprocessing step using a suitable distortion correction procedure, e. g. the one described in [23]. Equation (1) is valid for the distortion-corrected pixels and we conclude that we do not need to consider the distortion effects in our analysis.

B. Principal point in the fovea and peripheral images

We denote by $\boldsymbol{A}_f$, $\boldsymbol{R}_f$, $\boldsymbol{t}_f$ and $\boldsymbol{A}_p$, $\boldsymbol{R}_p$, $\boldsymbol{t}_p$ the intrinsic and extrinsic parameters of the foveal and peripheral camera, respectively. Lets now assume that the world coordinate system is aligned with the coordinate system of the foveal camera. In this case we have $\boldsymbol{R}_f = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{t}_f = 0$. Let $\hat{\boldsymbol{t}}$ be the position of the origin of the peripheral coordinate system expressed in the foveal coordinate system and let $\hat{\boldsymbol{R}}$ be the rotation matrix that rotates the basis vectors of the peripheral coordinate system into the basis vectors of the foveal coordinate system. We then have

$$\boldsymbol{R}_p\boldsymbol{M} + \boldsymbol{t}_p = \hat{\boldsymbol{R}}(\boldsymbol{M} - \hat{\boldsymbol{t}}). \tag{3}$$

The projections of a 3-D point $\boldsymbol{M}$ onto the foveal and peripheral image are then given by

$$x_f = \frac{\alpha_f X + \gamma_f Y}{Z}, \tag{4}$$

$$y_f = \frac{\beta_f Y}{Z}, \tag{5}$$

and

$$x_p = \frac{\alpha_p \boldsymbol{r}_1 \cdot (\boldsymbol{M} - \hat{\boldsymbol{t}}) + \gamma_p \boldsymbol{r}_2 \cdot (\boldsymbol{M} - \hat{\boldsymbol{t}})}{\boldsymbol{r}_3 \cdot (\boldsymbol{M} - \hat{\boldsymbol{t}})}, \tag{6}$$

$$y_p = \frac{\beta_p \boldsymbol{r}_2 \cdot (\boldsymbol{M} - \hat{\boldsymbol{t}})}{\boldsymbol{r}_3 \cdot (\boldsymbol{M} - \hat{\boldsymbol{t}})}, \tag{7}$$

where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}_3$ are the rows of the rotation matrix $\hat{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{r}_1^T & \boldsymbol{r}_2^T & \boldsymbol{r}_3^T \end{bmatrix}^T$. $\boldsymbol{M}$ projects onto the principal point in the fovea if $x_f = y_f = 0$. Assuming that $\boldsymbol{M}$ is in front of the camera, hence $Z > 0$, we obtain from Eq. (4) and (5) that $X = Y = 0$, which means that the point must lie on the optical axis of the foveal camera. Inserting this into Eq.

(6) and (7), we obtain the following expression for the ideal position $(\hat{x}_p, \hat{y}_p)$ in the peripheral image that results in the projection onto the principal point in the foveal image

$$\hat{x}_p = \frac{\alpha_p \boldsymbol{r}_1 \cdot \hat{\boldsymbol{t}} + \gamma_p \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}} - (\alpha_p r_{13} + \gamma_p r_{23}) Z}{\boldsymbol{r}_3 \cdot \hat{\boldsymbol{t}} - r_{33} Z}, \quad (8)$$

$$\hat{y}_p = \frac{\beta_p \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}} - \beta_p r_{23} Z}{\boldsymbol{r}_3 \hat{\boldsymbol{t}} - r_{33} Z}, \quad (9)$$

where $\begin{bmatrix} r_{13} & r_{23} & r_{33} \end{bmatrix}^T$ is the third column of $\hat{\boldsymbol{R}}$. Note that the ideal position in the periphery is independent from the intrinsic parameters of the foveal camera. It depends, however, on the distance of the point of interest from the foveation setup.

C. Displacement from the ideal position

Lets assume now that the 3-D point of interest $\boldsymbol{M}$ projects onto a pixel away from the principal point in foveal image by displacement (D_x, D_y). From (4) and (5) we have

$$D_x = \frac{\alpha_f X + \gamma_f Y}{Z}, \quad (10)$$

$$D_y = \frac{\beta_f Y}{Z}, \quad (11)$$

thus

$$X = \frac{D_x - \gamma_f D_y / \beta_f}{\alpha_f} Z, \quad (12)$$

$$Y = \frac{D_y}{\beta_f} Z. \quad (13)$$

Point $\begin{bmatrix} 0 & 0 & Z \end{bmatrix}^T$ is the point on the optical axis which is closest to $\boldsymbol{M}$. It projects onto $(\hat{x}_p, \hat{y}_p)$ in the peripheral view. We define (d_x, d_y) to be the displacement of the projection of $\boldsymbol{M}$ from this point in the peripheral view and we would like to express (D_x, D_y) in terms of (d_x, d_y). We have the following relationship

$$s \begin{bmatrix} \hat{x}_p + d_x \\ \hat{y}_p + d_y \\ 1 \end{bmatrix} = \boldsymbol{A}_p \begin{bmatrix} \left(r_{11} \left(D_x - \gamma_f D_y / \beta_f\right) / \alpha_f + r_{12} D_y / \beta_f + r_{13}\right) Z - \boldsymbol{r}_1 \cdot \hat{\boldsymbol{t}} \\ \left(r_{21} \left(D_x - \gamma_f D_y / \beta_f\right) / \alpha_f + r_{22} D_y / \beta_f + r_{23}\right) Z - \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}} \\ \left(r_{31} \left(D_x - \gamma_f D_y / \beta_f\right) / \alpha_f + r_{32} D_y / \beta_f + r_{33}\right) Z - \boldsymbol{r}_3 \cdot \hat{\boldsymbol{t}} \end{bmatrix}. \quad (14)$$

By subtracting (8) and (9) from (14), we can obtain a rather complex expression for the error in the periphery (d_x, d_y) in terms of the error in the fovea (D_x, D_y). This expression also depends on the distance Z of the point of interest from the camera setup. Fortunately, the result can be simplified by making some reasonable assumptions about the foveation setup. It is common to construct a foveated camera system in such a way that the optical axes of the peripheral and foveal camera are parallel. No calibration is needed to achieve this, only the cradles of both cameras must be built with sufficient precision. Standard industrial cameras are constructed precisely enough to support such an arrangement, which is used in most fixed foveation systems (like in Fig. 1). In this case we have $r_{31} = r_{32} = r_{13} = r_{23} = 0$, $r_{33} = 1$ and Eq. (14) becomes

$$s \begin{bmatrix} \hat{x}_p + d_x \\ \hat{y}_p + d_y \\ 1 \end{bmatrix} = \boldsymbol{A}_p \begin{bmatrix} \left(r_{11} \left(D_x - \gamma_f D_y / \beta_f\right) / \alpha_f + r_{12} D_y / \beta_f\right) Z - \boldsymbol{r}_1 \cdot \hat{\boldsymbol{t}} \\ \left(r_{21} \left(D_x - \gamma_f D_y / \beta_f\right) / \alpha_f + r_{22} D_y / \beta_f\right) Z - \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}} \\ Z - t_z \end{bmatrix}, \quad (15)$$

where $\hat{\boldsymbol{t}} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$. The denominator in (8) and (9) coincides with the third component in Eq. (15), hence subtracting (8) and (9) from (15) results in

$$d_x = \frac{Z}{Z - t_z} \left(\frac{r_{11} \alpha_p + r_{21} \gamma_p}{\alpha_f} D_x + \frac{-r_{11} \alpha_p \gamma_f + r_{12} \alpha_p \alpha_f - r_{21} \gamma_p \gamma_f + r_{22} \gamma_p \alpha_f}{\alpha_f \beta_f} D_y \right), \quad (16)$$

$$d_y = \frac{Z}{Z - t_z} \left(\frac{r_{21} \beta_p}{\alpha_f} D_x + \frac{-r_{21} \beta_p \gamma_f + r_{22} \beta_p \alpha_f}{\alpha_f \beta_f} D_y \right). \quad (17)$$

It is easy to invert this equation system to obtain the expression for the error in the fovea in terms of the error in the periphery and distance Z. We omit the details here and only present results with further simplifying assumptions. Lets assume that both cameras are completely aligned, i. e. $r_{21} = r_{12} = 0$ and $r_{11} = r_{22} = 1$ (no rotation around the optical axis). In this case we can calculate a simpler relationship between the error displacements in the foveal and peripheral images

$$D_x = \frac{Z - t_z}{Z} \cdot \frac{\alpha_f}{\alpha_p} \left(d_x + \frac{\alpha_p \gamma_f - \gamma_p \alpha_f}{\alpha_f \beta_p} d_y \right), \quad (18)$$

$$D_y = \frac{Z - t_z}{Z} \cdot \frac{\beta_f}{\beta_p} d_y. \quad (19)$$

The skew parameters γ_p and γ_f are normally much smaller than α_p, α_f, β_p, and β_f. Similarly, the displacement of the camera t_z is usually much smaller than the distance Z of the point of interest from the camera. Note that it is difficult to achieve $t_z = 0$ on a practical camera system because it is not easy to determine the exact positions of the projection centers and to place the cameras accordingly. Thus, for totally aligned cameras we have the following approximation

$$D_x \approx \frac{\alpha_f}{\alpha_p} d_x, \quad (20)$$

$$D_y \approx \frac{\beta_f}{\beta_p} d_y. \quad (21)$$

This means that the error from the ideal displacement in the peripheral image is scaled in the fovea by the ratio of focal lengths. This approximation is exact for perfect pinhole cameras ($\gamma_p = \gamma_f = 0$) with precisely aligned coordinate systems, i. e. $\hat{\boldsymbol{R}} = \boldsymbol{I}$ and $t_z = 0$. Since the focal length of the foveal camera is always greater than the focal length of the peripheral camera, i. e. $\alpha_p, \beta_p < \alpha_f, \beta_f$, the deviation from the principal point in the fovea is greater than the deviation

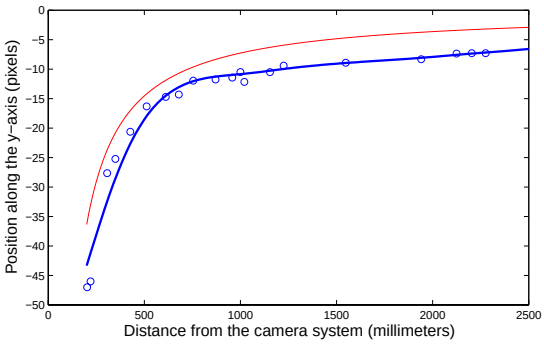


Fig. 3. Red curve: $\hat{y}_p$ with respect to the distance of the object from the camera system as calculated by Eq. (25) (totally aligned, ideal pinhole cameras, $\hat{\boldsymbol{R}} = \boldsymbol{I}$, $t_x = t_z = 0$, $t_y = 25$, $\alpha_p = \beta_p = 290.9$). Blue curve: $\hat{y}_p$ determined experimentally by placing the object manually at various distances so that it projects on the center of the foveal image. At each such configuration we measured the object's position in peripheral image and its distance from the eye (using stereo vision). The blue circles show these measurements.

from the ideal position in the peripheral image. This is, of course, an expected result.

D. Analysis of foveated vision systems

Making the same assumptions as when calculating (15), we obtain from Eq. (8) and (9) the following expression for the ideal position in the peripheral image

$$\hat{x}_p = \frac{\alpha_p \boldsymbol{r}_1 \cdot \hat{\boldsymbol{t}} + \gamma_p \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}}}{t_z - Z} \approx -\frac{\alpha_p \boldsymbol{r}_1 \cdot \hat{\boldsymbol{t}}}{Z}, \quad (22)$$

$$\hat{y}_p = \frac{\beta_p \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}}}{t_z - Z} \approx -\frac{\beta_p \boldsymbol{r}_2 \cdot \hat{\boldsymbol{t}}}{Z}. \quad (23)$$

We can again neglect the influence of γ_p, which is always significantly smaller than α_p and β_p. Note, however, that it is important that the cradles are built precisely and that the optical axes are aligned accurately because in Eq. (8) and (9) the zero terms r_{13} and r_{23} are multiplied by Z, which is normally large. Hence if the system is not built precisely, the above approximations are not valid. This is intuitive because if optical axes diverge, the foveal image will not overlap with the peripheral image as the distance increases and it is impossible to get a point into the fovea based on information from the peripheral image. The above equation system can be further simplified by assuming totally aligned pinhole cameras ($r_{21} = r_{12} = 0$ and $r_{11} = r_{22} = 1$, $t_z = 0$, $\gamma_p = 0$), which results in

$$\hat{x}_p = -\frac{\alpha_p t_x}{Z}, \quad (24)$$

$$\hat{y}_p = -\frac{\beta_p t_y}{Z}. \quad (25)$$

For the head in Fig. 1, the peripheral cameras are equipped with 3mm lenses and with CCD chips of size 6.6×4.4 millimeters, while the foveal cameras are equipped with 12mm lenses and with CCD chips of size 3.3×2.2 millimeters. The distance between them is about 25mm along the y-axis ($t_x \approx 0$, $t_y \approx 25$). Theoretically, the scaling factors of such cameras are $\alpha_p = \beta_p \approx 290.9$ and $\alpha_f = \beta_f \approx 1306.8$ when the cameras are calibrated for images of size 640×480.

Fig. 3 shows the variation of $\hat{y}_p$ with respect to Z under such assumptions and proves that our system indeed exhibits such characteristics. For $Z = 1$ meter, the ideal position in the periperal image is given by $\hat{x}_p = 0$ and $\hat{y}_p = -290.9 \times 25/1000 = -7.3$ pixels. For objects further away, $\hat{y}_p$ tends to zero. From Eq. (25) it follows that the necessary displacement doubles to -14.6 when $Z \approx 498$mm. Hence, if we fix Z to 1 meter and observe objects more than 0.5m away from the camera, the systematic error in the peripheral images will be less than 7.3 pixels. Eq. (21) tells us that the displacement from the central position in the foveal view will be at most $1306.8/290.9 \times 7.3 \approx 32.8$ pixels, hence we are still relatively close to the principal point in the foveal image. Note that fixing the distance Z is equivalent to replacing the perspective projection with the orthographic projection in our model.

IV. Closed-Loop Eye, Head, and Torso Control System

We developed a control system that is appropriate for the tracking task and enhances the appearance of the humanoid through mimicking aspects of human movement: human eyes follow object movement, but without head and body movement have a limited range; thus, the robot's control system supports its eye movements through head and body movements.

Our humanoid DB has 30 degrees of freedom; we used 10 degrees of freedom (4 on the eyes + 3 on the head + 3 on the torso) to maintain the view of the object. It was important for the control system not to rely on exact knowledge of the robot's kinematics since the action of the robot's joints varies over time due to joint and valve wear-and-tear and maintenance activities.

The robot's primary mechanism for maintaining the view of the object of interest is eye movement: the control system continuously alters the pan and tilt of each eye to keep the object near the center of the corresponding view (i.e. visual servo control [25]). Independent eye motion is acceptable when the object is being tracked properly in both peripheral views, but looks rather unnatural when one eye loses its view of the object while the other eye continues to roam. Our solution is to introduce a gentle cross-coupling between a camera's view and the control of the other eye. Thus, when a camera's view of the target is lost, its corresponding eye continues to move, fairly slowly, under the influence of the other camera's view. As well as appearing natural, such eye movements improve the likelihood of re-locating the object.

We consider that the task of the robot's head is to assist the eyes by increasing the viewable area and avoiding unnatural eye poses. Similarly, we consider that the task of the robot's body is to assist the head, further increasing the viewable area and avoiding unnatural head poses.

To aid in coordinating the joints, we assign a relaxation position to each joint and vision blob [1]. The relaxation position for the blobs is near the center of the view, and the eyes' task is to bring the blobs to that position. The relaxation position

[1]Vision blobs give the hypothesized 2-D object locations in the image.

for the 4 eye joints is to face forward, and the head's task is to bring the eyes to that position. Further, the 3 neck joints have a relaxation position, and the torso's task is to bring the head to that position. For example, if the object of interest is up and to the left, the eyes would tilt up and pan left, causing the head would tilt up and turn left, and the torso to lean back and turn.

The complete control system is implemented as a network of PD controllers expressing the assistive relationships. The PD controllers are based on simplified mappings between visual coordinates and joint angles, which are described below, rather than on a full kinematic model. Such mappings are sufficient because the system is closed-loop and can make corrective movements to converge towards the desired configuration.

We define the *desired change* for self-relaxation, D, for each joint,

$$D_{joint} = \left(\theta^*_{joint} - \theta_{joint}\right) - K_d \dot{\theta}_{joint}, \tag{26}$$

where K_d is the derivative gain for joints; θ is the current joint angle; $\dot{\theta}$ is the current joint angular velocity, and the asterisk indicates the relaxation position. The derivative components help to compensate for the speed of the blobs and assisted joints.

The desired change for a vision blob is:

$$D_{blob} = (x^*_{blob} - x_{blob}) - K_{dv} \dot{x}_{blob}, \tag{27}$$

where K_{dv} is the derivative gain for vision blobs; and x is position in pixels.

The purpose of the *left eye pan* ($\mathsf{L}EP$) joint is to move the target into the center of the left camera's field of view:

$$\begin{aligned}\hat{\dot{\theta}}_{\mathsf{L}EP} = K_p \times \Big[& K_{\text{relaxation}} D_{\mathsf{L}EP} \\ & - K_{\text{target}\rightarrow EP} K_v C_{\mathsf{LX}\text{target}} D_{\mathsf{LX}\text{target}} \\ & + K_{\text{cross-target}\rightarrow EP} K_v C_{\mathsf{RX}\text{target}} D_{\mathsf{RX}\text{target}} \Big],\end{aligned} \tag{28}$$

where $\hat{\dot{\theta}}_{\mathsf{L}EP}$ is the new target velocity for the joint; L and R represent left and right; X represents the x pixels axis; K_p is the proportional gain; K_v is the proportional gain for vision blobs; C_{blob} is the tracking confidence for that blob; and the gain $K_{\text{cross-target}\rightarrow EP} < K_{\text{target}\rightarrow EP}$.

The purpose of the *left eye tilt* ($\mathsf{L}ET$) joint is to move the target into the center of the left camera's field of view:

$$\begin{aligned}\hat{\dot{\theta}}_{\mathsf{L}ET} = K_p \times \Big[& K_{\text{relaxation}} D_{\mathsf{L}ET} \\ & - K_{\text{target}\rightarrow ET} K_v C_{\mathsf{LY}\text{target}} D_{\mathsf{LY}\text{target}} \\ & - K_{\text{cross-target}\rightarrow ET} K_v C_{\mathsf{RY}\text{target}} D_{\mathsf{RY}\text{target}} \Big].\end{aligned} \tag{29}$$

The equations for the right eye pan and tilt joints are the same as for the left, except that L becomes R and vice versa.

Head nod joint (HN) assists the eye tilt joints:

$$\begin{aligned}\hat{\dot{\theta}}_{HN} = K_p \times \Big[& K_{\text{relaxation}} D_{HN} \\ & - K_{ET\rightarrow HN} \left(D_{\mathsf{L}ET} + D_{\mathsf{R}ET}\right) \Big].\end{aligned} \tag{30}$$

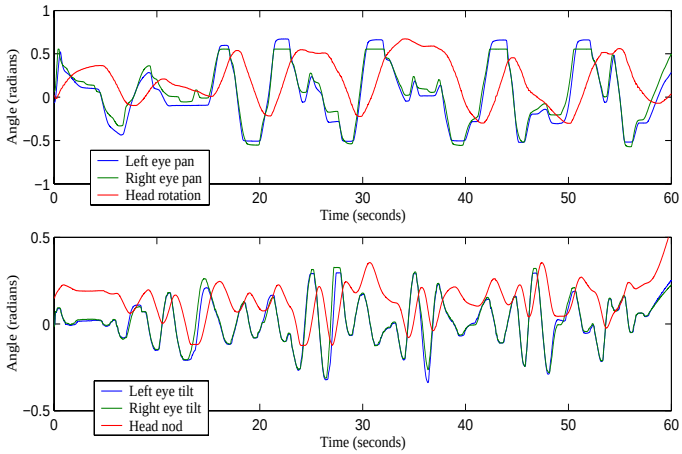


Fig. 4. Eye (blue and green trajectories) and head (red trajectories) joints during foveation. On DB both eye pans and head rotation represent rotations around axes parallel to the vertical body axis (left - right change of viewing direction), while both eye tilts and head nod represent rotations around axes parallel to the shoulder axis (up - down change of view). These are the most important movements for foveation. As expected left and right eye pans as well as left and right eye tilts follow similar trajectories to maintain the direction of view. Due to the assistive relationship between eye pans and head rotation, the head rotation joint follows the pan angles of both eyes. There is a similar relationship between eye tilts and head nod motion.

The *head tilt joint* (HT), which tilts the head from side to side, moves to assist the pan (EP) and equalize the tilt (ET) of the eyes:

$$\begin{aligned}\hat{\dot{\theta}}_{HT} = K_p \times \Big[& K_{\text{relaxation}} D_{HT} \\ & - K_{EP\rightarrow HT} \left(D_{\mathsf{L}EP} - D_{\mathsf{R}EP}\right) \\ & - K_{ET\rightarrow HT} \left(D_{\mathsf{L}ET} - D_{\mathsf{R}ET}\right) \Big].\end{aligned} \tag{31}$$

The *torso flexion-extension joint* (TFE) assists the head nod joint:

$$\hat{\dot{\theta}}_{TFE} = K_p \times \Big[K_{\text{relaxation}} D_{TFE} - K_{HN\rightarrow TFE} D_{HN} \Big]. \tag{32}$$

We omitted the control rules for *head rotate joint* (HR), *torso rotate joint* (TR) and *torso abduction-adduction joint* (TAA). These controllers are defined equivalently to the controllers for the head nod joint and torso flexion-extension joint, the only difference being that they support different preceding degrees of freedom. We used a similar closed-loop control scheme as part of a humanoid reaching system [13].

A. Control experiments

The graphs in Fig. 4 show the assistive relationships between different degrees of freedom. Even though the eye joints often hit the joint limits, the robot could still maintain the view of the object by making use of head movements. This is also useful if one of the joints fails; the robot can still function, although with degraded performance.

To demonstrate the reliability of the pursuit strategy, we measured the velocity of object motion while the robot attempted to maintain its view in the fovea. The object was moved in front of the robot by an experimenter. For

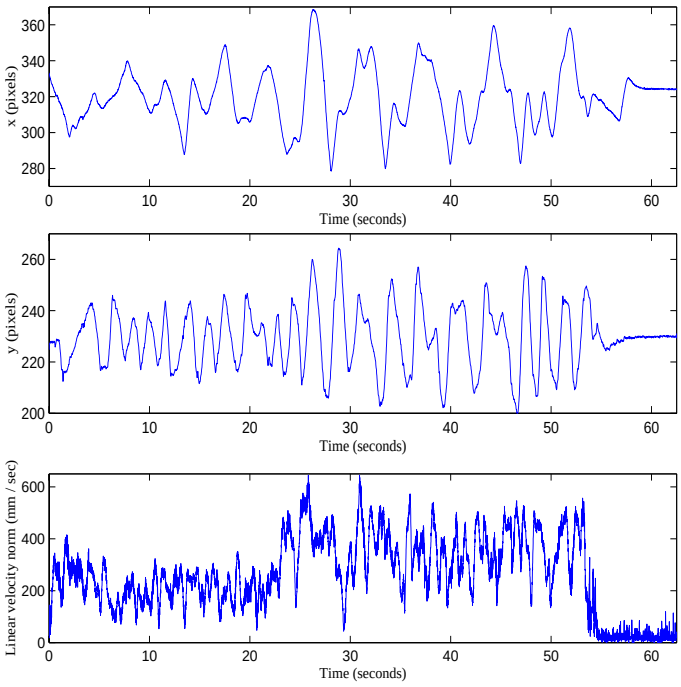


Fig. 5. Effect of object velocity on accuracy of foveation. The upper two graphs show object position in one of the peripheral images. The desired image position was set at $(320, 230.3)$. It was slightly displaced from the image center, which was at $(320, 240)$, to account for the displacement of foveal cameras from the peripheral ones. The lower graph depicts the norm of linear velocity of the tracked object.

this purpose we attached three active markers to the object and measured their motion with an optical tracking system Visualeyze. Since the rotational component of motion was negligible, we estimated only linear velocity of object motion. Not surprisingly, the robot was more successful at maintaining the view of the object when the object motion was slower. In the graphs the object velocity increases after approximately 23 seconds. Nevertheless, even with fairly fast motions (more than 0.5 m/sec), the robot was still able to maintain the view of the object using information from peripheral cameras. Although the accuracy in this case is often not sufficient for object recognition, the robot can at least restart the classifier once the object appear in the fovea again. As shown in the graphs, the robot quickly directs its eyes towards the object and maintains its foveal view after the object comes to a standstill (approximately at 60 seconds).

Our experiments demonstrate that the proposed control strategy is successful at smoothly pursuing objects of interest and that it can maintain the view of the object in the fovea. In the following we propose a new classification approach to object recognition that can be used once the object is stabilized in the fovea.

V. Object Representation

Early approaches to object recognition in static images were implemented predominantly around the 3-D reconstruction paradigm of Marr [26], but many of the more recent recognition systems make use of viewpoint-dependent models [27]–[29]. View-based strategies are receiving increasing attention because it has been recognized that 3-D reconstruction is difficult in practice (mainly due to difficulties in segmentation). There is also psychophysical evidence that supports these techniques [30].

In a view-based approach, each object is represented by a number of images (or features extracted from 2-D images) taken from different viewpoints. These model images are compared to test images acquired by the robot. However, since both a humanoid robot and objects can move in space, objects appear in images at different positions, orientations and scales. It is obviously not feasible to learn all possible views due to the memory limitations. The number of required views can, however, be reduced by normalizing the subimages that contain objects of interest to images of fixed size. Here we demonstrate how to accomplish this by exploiting the results of the tracker.

A. Normalization through Affine Warping

As described in Section II, our tracker estimates the shape of the tracked object using second order statistics of pixels that are probabilistically classified as "blob pixels". By computing the eigenvalue decomposition of the associated covariance matrix $\mathbf{\Gamma}$, we can estimate the object orientation in the image and the extent of the object along its major and minor axes. In other words, we can estimate the ellipse enclosing the object pixels. As the lengths of both axes can differ significantly, it makes sense to normalize each image along the principal axis directions instead of image coordinate axes and to apply a different scaling factor along each of these directions, taking into account the length of the corresponding axis. By aligning the object's axes with the coordinate axes, we can also achieve invariance against planar rotations. This reduces the number of views that need to be stored to represent an object. The estimation of planar orientation angle is often noisy, especially for complex objects. In such cases, we need to collect more training data to account for all possible views in the model, but the number of necessary views is still significantly reduced by this approach.

Normalization along the principal axes is implemented by applying the following transformations: (1) translate the blob so that its center is aligned with the origin of the image, (2) rotate the blob so that its principal directions are aligned with the coordinate axes, (3) scale the blob so that its major and minor axis are as long as the sides of a predefined window, (4) translate the blob so that its center is aligned with the center of the new window. The resulting mapping in homogeneous coordinates is given by the following affine transformation:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \frac{w_x}{2} \\ 0 & 1 & \frac{w_y}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{w_x}{2a} & 0 & 0 \\ 0 & \frac{w_y}{2b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}(\theta)^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -u \\ 0 & 1 & -v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}, \quad (33)$$

Fig. 6. Example images of ten objects. Scaling and planar rotations are accounted for by affine warping using the results of visual tracking.

where $\boldsymbol{u} = [u, v]^T$ and θ are the position and orientation of the blob, a and b are the half lengths of its major and minor axis, and w_x and w_y are the predefined width and height of the window onto which we map the window containing the blob.

B. Gabor Jets

Early view-based approaches used raw grayscale images as input to the selected classifier, e. g. principal component analysis [31]. This kind of approaches turned out to be fairly successful as long as the amount of noise in the images is small and the illumination conditions do not change. To achieve robustness against brightness changes, it is necessary to compute an improved, illumination insensitive characterization of the local image structure. Some of the more recent recognition systems therefore apply a bank of illumination-insensitive filters to the original images before starting the recognition process. We follow the biologically motivated approach of Wiskott et al. [32], who proposed to apply a bank of Gabor filters to the incoming images containing objects of interest. Gabor filters are known to be good edge detectors and are therefore robust against varying brightness. They have limited support both in space and frequency domain and have certain amount of robustness against translation, distortion, rotation, and scaling [32].

Complex Gabor kernels are defined by

$$\begin{aligned} \boldsymbol{\Phi}_{\mu,\nu}(\boldsymbol{x}) &= \frac{\|\boldsymbol{k}_{\mu,\nu}\|^2}{\sigma^2} \cdot \exp\left(-\frac{\|\boldsymbol{k}_{\mu,\nu}\|^2 \|\boldsymbol{x}\|^2}{2\sigma^2}\right) \cdot \\ &\left(\exp\left(i\boldsymbol{k}_{\mu,\nu}^T \boldsymbol{x}\right) - \exp\left(-\frac{\sigma^2}{2}\right)\right), \end{aligned} \quad (34)$$

where $\boldsymbol{k}_{\mu,\nu} = k_\nu[\cos(\phi_\mu),\ sin(\phi_\mu)]^T$. Gabor jet at pixel $\boldsymbol{x}$ is defined as a set of complex coefficients $\{J_j^{\boldsymbol{x}}\}$ obtained by convolving the image with a number of Gabor kernels at this pixel. Gabor kernels are selected so that they sample a number of different wavelengths k_ν and orientations ϕ_μ. Wiskott et al. [32] proposed to use $k_\nu = 2^{-\frac{\nu+2}{2}},\ \nu = 0, \ldots, 4$, and $\phi_\mu = \mu\frac{\pi}{8},\ \mu = 0, \ldots, 7$, but this depends both on the size of the incoming images and the image structure. They showed that the similarity between the jets can be measured by

$$\mathrm{S}\left(\{J_i^{\boldsymbol{x}}\}, \{J_i^{\boldsymbol{y}}\}\right) = \frac{\boldsymbol{a}_{\boldsymbol{x}}^T * \boldsymbol{a}_{\boldsymbol{y}}}{\|\boldsymbol{a}_{\boldsymbol{x}}\| \|\boldsymbol{a}_{\boldsymbol{y}}\|}, \quad (35)$$

where $\boldsymbol{a}_{\boldsymbol{x}} = [|J_1^{\boldsymbol{x}}|, \ldots, |J_s^{\boldsymbol{x}}|]^T$ and s is the number of complex Gabor kernels. This is based on the fact that the magnitudes of complex coefficients vary slowly with the position of the jet in the image.

The calculation of Gabor jets is computationally expensive because the underlying Gabor kernel functions have large support. It is therefore advantageous to compute the convolutions $\boldsymbol{I} * \boldsymbol{\Phi}_{\mu,\nu}$ with the help of the Fast Fourier Transform

$$\boldsymbol{I} * \Phi_{\mu,\nu} = \mathcal{F}^{-1}\left(\mathcal{F}(\boldsymbol{I}) \cdot \mathcal{F}(\boldsymbol{\Phi}_{\mu,\nu})\right) \quad (36)$$

Using FFT we were able to compute 40 Gabor filters on images of resolution 120x160 at 15 frames per second on a dual processor 2.8GHz PC. This was essential for achieving real-time operation of the system.

We use Gabor jets to generate feature vectors for recognition. To reduce the dimensionality of these feature vectors, we did not make use of all jets. Ideally, one would calculate the jets only at important local features. We did not attempt to extract local features because it is often difficult to extract them in a stable manner. Instead we decided to build the feature vectors from Gabor jets positioned on a regular grid of pixels (the selected grid size was 5×5). Normalized jets $\{a_j^{\boldsymbol{x}}/\|\boldsymbol{a}^{\boldsymbol{x}}\|\}_{j=1}^n$ calculated on this grid and belonging to the ellipse enclosing the object like in Fig. 7 were finally utilized to build feature vectors.

It is important to note that we first scale the object images to a fixed size and then apply Gabor filters. In this way we ensure that the size of local structure in the acquired images does not change and consequently we do not need to change the frequencies k_ν of the applied filters.

C. Training

Our goal is to learn a three-dimensional representation for each object of interest. To achieve this, it is necessary to show the objects to the humanoid from all relevant viewing directions. In computer vision this is normally achieved by

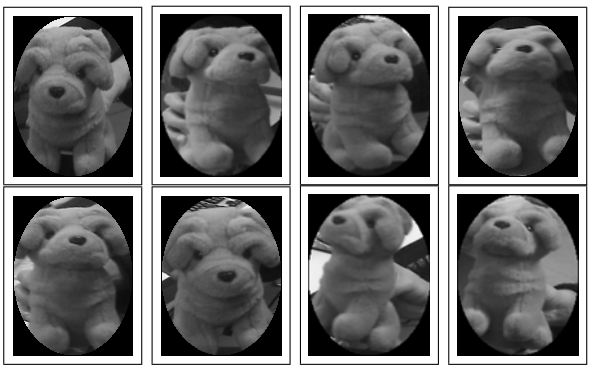


Fig. 7. Training images for one of the objects used in statistical experiments. To take care of rotations in depth, we must collect a sufficient amount of typical viewpoints.

accurate turntables that enable the collection of images from regularly distributed viewpoints. However, this solution is not practical for humanoid robotics, where on-line interaction is often paramount. We therefore explored whether it is possible to reliably learn 3-D descriptions from images collected while a human teacher moves the object in front of the robot. Using the previously described attention, tracking, and smooth pursuit systems, the robot acquires the foveal images of the object in motion and collects feature vectors from many different viewpoints. We show in the result section that such collection of feature vectors is sufficient for 3-D object recognition.

VI. Recognition with Support Vector Machines

Support vector machines (SVMs) are a relatively new classification system rooted in the statistical learning theory. They are considered as state of the art classifiers because they deliver high performance in real-world applications. To distinguish between two different classes, a support vector machine draws the (optimal) separating hyperplane between training data points belonging to the two classes. The hyperplane is optimal in the sense that it separates the largest fraction of points from each class, while maximizing the distance from either class to the hyperplane. First approaches that utilized SVMs for object recognition used the basic form that deals with the two-class classification problem. A binary tree strategy [33], [34] was proposed to solve the multi-class problem. While this approach provides a simple and powerful classification framework, it cannot capture correlations between the different classes since it breaks a multi-class problem into multiple independent binary problems [35]. In addition, the result is not independent of how the candidate objects are paired. There were attempts to generalize SVMs to multi-class problems, but practical implementation have started to emerge only recently. In this paper we follow the generalization proposed in [35]. We made use of the implementation described in [36], [37].

In the following we explain the basic theory of multi-class SVMs. We continue with the definition of suitable kernel functions based on Gabor filtered images and color histograms. We finally show how the segmentation probabilities can be incorporated into the support vector learning and classification.

A. Multi-class Support Vector Machines

Multi-class classification addresses the problem of finding a function defined from an input space $\boldsymbol{\Psi} \subset \mathcal{R}^n$ onto a set of classes $\boldsymbol{\Omega} = \{1, \ldots, m\}$. Let $\boldsymbol{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, $\boldsymbol{x}_i \in \boldsymbol{\Psi}$, $y_i \in \boldsymbol{\Omega}$, be a set of n training samples. We look for a function $\boldsymbol{H} : \boldsymbol{\Psi} \rightarrow \boldsymbol{\Omega}$ so that $\boldsymbol{H}(\boldsymbol{x}_i) = y_i$. Crammer and Singer [35][2] proposed to look for $\boldsymbol{H}$ among classifiers of the form

$$\boldsymbol{H}_{\boldsymbol{M},\boldsymbol{b}}(\boldsymbol{x}) = \arg\max_{r \in \boldsymbol{\Omega}} \{\boldsymbol{M}_r * \boldsymbol{x} + b_r\}, \tag{37}$$

where $\boldsymbol{b} = [b_1, \ldots, b_k]^T$, $\boldsymbol{M} \in \mathcal{R}^{m \times n}$ is a matrix of size $m \times n$ and $\boldsymbol{M}_r$ is the r-th row of $\boldsymbol{M}$. Standard two-class SVMs result in classifiers $\boldsymbol{H} = (\boldsymbol{w}, b)$ that predict the label of a data point $\boldsymbol{x}$ as 1 if $\boldsymbol{w} * \boldsymbol{x} + b \geq 0$ and 2 otherwise. They can be expressed in the above form by taking a matrix $\boldsymbol{M}$ with rows $\boldsymbol{M}_1 = \boldsymbol{w}$, $\boldsymbol{M}_2 = -\boldsymbol{w}$, and $b_1 = b$, $b_2 = -b$.

Crammer and Singer [35] showed that the optimal multi-class support vector machine can be calculated by solving the following optimization problem

$$\min_{\boldsymbol{M},\boldsymbol{b},\boldsymbol{\xi}} \frac{1}{2}\beta \left(\sum_{r=1}^{m} \sum_{i=1}^{n} M_{ri}^2 + \sum_{r=1}^{m} b_r^2 \right) + \sum_{i=1}^{n} \xi_i \tag{38}$$

$$\text{subject to : } \boldsymbol{M}_{y_i} * \boldsymbol{x}_i + b_{y_i} + \delta_{y_i,r} - \boldsymbol{M}_r * \boldsymbol{x}_i - b_r \geq 1 - \xi_i$$
$$\xi_i \geq 0, \forall\, i, r$$

Here $\boldsymbol{\xi}_i \geq 0$ are the slack variables that need to be introduced to solve non-separable problems. This constrained quadratic optimization problem is convex and can therefore be solved efficiently. Note that in optimization problem (38) the data points appear only in inner products $\boldsymbol{M}_j * \boldsymbol{x}_i$. Moreover, it can be shown that its solution can be written as

$$\boldsymbol{M}_r^T = \sum_{i=1}^{n} \tau_{ir} \boldsymbol{x}_i, \;\; \tau_{i,r} \geq 0, \;\; r = 1, \ldots, m, \tag{39}$$

and the corresponding decision function is

$$\boldsymbol{H}_{\boldsymbol{M},\boldsymbol{b}}(\boldsymbol{x}) = \arg\max_{r \in \boldsymbol{\Omega}} \left\{ \sum_{i=1}^{n} \tau_{i,r} \boldsymbol{x}_i^T * \boldsymbol{x} + b_r \right\}. \tag{40}$$

Again, the data points appear only in inner products $\boldsymbol{x}_i^T * \boldsymbol{x}$. This makes it possible to transform the data points with a nonlinear map $\boldsymbol{\Phi}$ into a higher dimensional feature space and to construct the optimal hyperplane in this space. The nonlinear maps of interest are those that allow for an efficient calculation of high-dimensional inner products via kernel functions

$$\mathrm{K}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\Phi}(\boldsymbol{x}) * \boldsymbol{\Phi}(\boldsymbol{y}). \tag{41}$$

To find the optimal multi-class support vector machine in a higher dimensional feature space, we need to solve a constrained quadratic optimization problem in which inner products $\boldsymbol{\Phi}(\boldsymbol{x}_i) * \boldsymbol{\Phi}(\boldsymbol{x}_j)$ are replaced with the kernel function $\mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The decision function (40) becomes

$$\boldsymbol{H}_{\boldsymbol{M},\boldsymbol{b}}(\boldsymbol{x}) = \arg\max_{r \in \boldsymbol{\Omega}} \left\{ \sum_{i=1}^{n} \tau_{i,r} \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b_r \right\}. \tag{42}$$

The convergence of the optimization algorithm can be guaranteed for all kernel functions K that allow the construction of nonlinear mapping $\boldsymbol{\Phi}$ such that (41) holds. The condition for this is given by the Mercer's theorem [38].

B. Kernel Functions for Foveated Recognition

Now we turn to the problem of finding a suitable decision function for object recognition. The similarity measure for Gabor jets (35) provides a good starting point. Let $\boldsymbol{X}_G$ be the set of all grid points within two normalized images at which Gabor jets are calculated and let $J_{\boldsymbol{X}_G}$ and $L_{\boldsymbol{X}_G}$ be

[2]The bias parameters b_r were omitted in [35] to simplify the optimization problem. Here we keep them in the interest of clarity of presentation.

the Gabor jets calculated in two different images, but on the same grid points

$$\mathrm{K}_G(J_{\boldsymbol{X}_G}, L_{\boldsymbol{X}_G}) = \exp\left(-\rho_1 \frac{1}{M} \sum_{\boldsymbol{x}\in\boldsymbol{X}_G} \left(1 - \frac{\boldsymbol{a}_{\boldsymbol{x}}^T * \boldsymbol{b}_{\boldsymbol{x}}^T}{\|\boldsymbol{a}_{\boldsymbol{x}}\|\|\boldsymbol{b}_{\boldsymbol{x}}\|}\right)\right), \tag{43}$$

where M is the number of grid points in $\boldsymbol{X}_G$. This function satisfies the Mercer's condition [38] and can thus be used for support vector learning.

Kernel function 43 assumes that the sets of grid points $\boldsymbol{X}_G$ does not change from image to image. Wallraven at al. [39], however, showed that it is possible to define kernel functions using local feature detectors computed on sets of image points that vary from image to image. They proposed kernel functions defined on feature vectors of variable lengths and with different ordering of features. While feature order is not a problem for our system due to the affine warping procedure, it would be advantageous to exclude some of the grid points. Images presented in Fig. 6 and 7 make it clear that some of the points in the warped images do not belong to the object, even after clipping the parts outside of the enclosing ellipse. We can, however, use the results of the tracking/segmentation to exclude such points from the calculation. For each pixel, the tracker described in Section II can estimate the probability whether or not this pixel belong to the tracked object. We can thus define the set $\boldsymbol{X}_G$ on each image to include only points for which these probabilities are greater than a pre-specified threshold. Since we have not yet tested the exclusion of feature points for kernel functions using local color histograms, we here limit the discussion to Gabor jets. Let $\boldsymbol{X}_G^1$ and $\boldsymbol{X}_G^2$ be two sets of grid points with tracking probabilities greater than a pre-specified threshold. We can define a new kernel function

$$\mathrm{K}'_G(J_{\boldsymbol{X}_G^1}, L_{\boldsymbol{X}_G^2}) = \boldsymbol{K}_G(J_{\boldsymbol{X}_G^1 \cap \boldsymbol{X}_G^2}, L_{\boldsymbol{X}_G^1 \cap \boldsymbol{X}_G^2}) \cdot \exp\left(-\rho_1 \frac{1}{M} \left(\sum_{\boldsymbol{x}\in\boldsymbol{X}_G^1 \cup \boldsymbol{X}_G^2 - \boldsymbol{X}_G^1 \cap \boldsymbol{X}_G^2} 2\right)\right), \tag{44}$$

where M is the number of grid points in $\boldsymbol{X}_G^1 \bigcup \boldsymbol{X}_G^2$. We add the penalty of 2 for grid points that are not classified as object points only in one of both images because this is the highest possible value for one term in the criterion function (43). While this kernel function assumes that the ordering of grid points is the same in both images, it is much less computationally expensive than the more general functions proposed in [39]. This is important both for faster training and for real-time recognition.

VII. Experimental results

We used a set of ten objects to test the performance of the recognition system on a humanoid robot (6 teddy bears, two toy dogs, a coffee mug, and a face, see Fig. 6). For each object we recorded two or more movies using a video stream coming from DB's foveal cameras. The cameras were controlled as described in Section III. In each of the recording sessions the experimenter attempted to show one of the objects to the robot from all relevant viewing directions. One movie per object was used to construct the SVM classifier, while one of the other movies served as input to test the classifiers. Each movie was one minute long and we used at most 4 images per second for training. Since slightly more than first ten seconds of the movies were needed to initialize the tracker, we had at most 208 training images per object. For testing we used 10 images per second, which resulted in 487 test images per object. Except for the results of Table IV, all the percentages presented here were calculated using the classification results obtained from 4870 test images. Three types of classifiers were used to test the performance of foveated recognition. The first two were multi-class support vector machines based on kernel functions K_G and K'_G from Eq. (43) and (44), respectively. They are denoted as SVM in Table I - VI. Gabor jets were calculated at 8 different orientations and 5 different scales and the grid size was 5 pixels in both directions. The filters were scaled appropriately when using lower resolution images. We did not use color for SVM learning because all test objects had similar color. We utilized this feature to construct a common color mixture model for tracking and probabilistic segmentation. The third classifier was the nearest neighbor classifier (NNC) that used the similarity measure (35) – summed over all grid points – to calculate the nearest neighbor based on the same Gabor jets as input data.

Results in Tables I - III demonstrate that foveation is very useful for recognition. Kernel function (44) was used here. The classification results clearly become worse with the decreasing resolution. In fact, the data in Table III had to be calculated differently because we could not estimate the planar orientation accurately enough for affine warping, which made the normalization procedure fail. This resulted in significantly worse classification results. To calculate the results of Table III, we sampled the Gabor jets on a 20 pixel grid, which resulted in the same number of grid points as when the image resolution is reduced from 160×120 to 40×30 and the grid size is kept the same. Still, the recognition rate dropped even with such data. Our results also show that we can collect enough training data even without using accurate turntables to systematically collect the images.

We also tested the performance of the system on data captured under changed lighting condition (see Fig. 8) and on noise corrupted data (see Fig. 9). We used two objects (the last teddy bear and the toy dog from Fig. 6). The

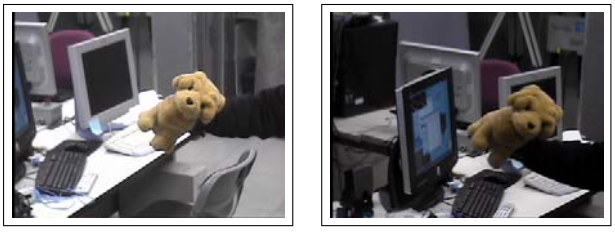


Fig. 8. Images taken under different lighting conditions

Fig. 9. Images degraded with white Gaussian noise (std. dev. = 10)

TABLE I

CORRECT CLASSIFICATION RATE (IMAGE RESOLUTION 120×160 PIXELS)

Training views per object	SVM	NNC
208	97.6 %	95.9 %
104	96.7 %	93.7 %
52	95.1 %	91.5 %
26	91.9 %	86.7 %

TABLE II

CORRECT CLASSIFICATION RATE (IMAGE RESOLUTION 60×80 PIXELS)

Training views per object	SVM	NNC
208	94.2 %	89.3 %
104	92.4 %	87.3 %
52	90.7 %	84.4 %
26	86.7 %	79.2 %

TABLE III

CORRECT CLASSIFICATION RATE (IMAGE RESOLUTION 30×40 PIXELS)

Training views per object	SVM	NNC
208	91.0 %	84.7 %
104	87.2 %	81.5 %
52	82.4 %	77.8 %
26	77.1 %	72.1 %

classification performance for these two objects on original images was a bit higher than the combined performance, but this was purely coincidental and we did not intentionally select these two object to test the varying brightness condition. For classification we used the same SVMs as in Tables I-III. While the performance decreased slightly on darker images, the results show that the method still performs well in such conditions. This is due to the properties of Gabor jets and due to the normalization of jets given by the similarity function (35). Our experiments showed that the classification rate drops significantly if one of the standard kernel functions, e. g. a linear kernel, is used for the support vector learning.

Unlike in other tables, the results of Table V and VI were calculated using SVMs based on kernel function K_G from Eq. (43), thus not taking into account the segmentation results. The segmentation results were not used for the nearest neighbor classification either. Comparing Tables V and VI we can say that SVMs are robust against noise as well. The results of

TABLE IV

CORRECT CLASSIFICATION RATE FOR IMAGES WITH VARYING LIGHTING CONDITIONS (SEE FIG. 8). ONLY TWO OBJECT WERE TESTED IN THIS CASE (THE DATABASE STILL CONTAINED TEN OBJECTS) AND SVMS CALCULATED BASED ON 208 VIEWS PER TRAINING OBJECTS WERE USED.

Image resolution	normal	dark	very dark
120×160	99.5 %	97.7 %	97.9 %
60×80	96.7 %	93.5 %	95.0 %
30×40	93.6 %	89.3 %	88.2 %

Table VI can be directly compared to Table II, the only difference being in the use of segmentation results. While both types of SVMs performed well in this case, the performance of the nearest neighbor classification dropped significantly when all the data from the enclosing ellipse was used. This shows that unlike nearest neighbor classification, SVMs can cope well with outliers. Nevertheless, it is still advantageous to use the kernel function that can include the segmentation results because such an approach reduces the amount of data that needs to be considered to calculate SVMs, hence resulting in faster computing times. We expect that differences between the two types of kernel functions would become more significant for objects that cannot be accurately enclosed within an ellipse.

The presented results cannot be directly compared to the results on standard databases for benchmarking object recognition algorithms because here the training sets are much less complete. Some of the classification errors are caused by the lack of training data rather than by a deficient classification approach. Unlike many approaches from the computer vision literature that avoid the problem of finding objects, we tested the system on images obtained through a realistic object track-

TABLE V

CORRECT CLASSIFICATION RATE FOR NOISE DEGRADED IMAGES (SEE FIG. 9). THE IMAGE RESOLUTION WAS 60×80 AND SEGMENTATION RESULTS WERE NOT USED.

Training views per object	SVM	NNC
208	91.5 %	79.8 %
104	90.7 %	74.5 %
52	90.5 %	68.0 %
26	87.1 %	60.3 %

TABLE VI

CORRECT CLASSIFICATION RATE WITHOUT NOISE DEGRADATION. THE IMAGE RESOLUTION WAS 60×80 AND SEGMENTATION RESULTS WERE NOT USED.

Training views per object	SVM	NNC
208	94.4 %	75.8 %
104	93.1 %	69.2 %
52	91.4 %	60.3 %
26	88.1 %	53.6 %

Fig. 10. DB in behavioral experiment. The task of the robot is to point towards the desired object (toy dog in the right image) and to ignore the rest of the objects.

ing and segmentation procedure. Only such data is relevant for foveated object recognition because without some kind of segmentation it is not possible to direct the fovea towards the objects of interest.

Except for the training of SVMs, all the proposed methods work in real-time and we used the system in interactive tasks such as selecting the desired toy and reaching for it (see Fig. 10). In these experiments we exploited the dynamic nature of the system and ran the recognition process on a time sequence of images. The object was deemed recognized only if the classification result does not change over a certain period of time (few seconds in our experiments). This is based on the assumption that correct classifications are stable whereas misclassifications are not and change as the viewpoint changes. Due to the high classification rates presented in Tab. I - VI, this approach was efficient at filtering out the classification errors.

VIII. Conclusions

Our experiments have shown that the proposed system is effective for foveated object recognition. We presented the first systematic study in humanoid robotics that takes into account various factors influencing the performance of the foveated object recognition. We showed both theoretically and empirically what kind of accuracy we can expect from the foveated setup using two cameras per eye. A closed-loop control method that can be used to quickly direct the robot's eyes towards the object of interest was also presented. It does not need accurate knowledge of eye and body kinematics and can exploit the redundancies of the humanoid.

Most of the previous approaches that employed support vector machines for object recognition used binary SVMs combined with decision trees [33], [34], [39] to solve the multi-class recognition problem. Our system is one of the first (if not the first) object recognition systems that makes use of multi-class SVMs to solve the multi-class recognition problem. We developed a new kernel function based on the Gabor jet similarity measure that can utilize the results of bottom-up segmentation. Finally, we showed that normalization based on affine warping effectively reduces the amount of data needed to train the support vector machines. Object representations can be effectively learned just by collecting the data while the demonstrator attempts to show the objects from all relevant viewing directions. Experimental results show high recognition rates in realistic test environments.

Acknowledgment

This work was supported in part by the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

[1] B. Scassellati, "Eye finding via face detection for a foveated, active vision system," in *Proc. Fifteenth Nat. Conf. Artifficial Intelligence (AAAI '98)*, Madison, Wisconsin, 1998, pp. 969–976.

[2] C. G. 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 Intelligent Systems*, vol. 15, no. 4, pp. 46–56, July/August 2000.

[3] H. Kozima and H. Yano, "A robot that learns to communicate with human caregivers," in *Proc. Int. Workshop on Epigenetic Robotics*, Lund, Sweden, 2001.

[4] C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati, "Social constraints on animate vision," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 31, no. 5, pp. 443–452, July/August 2001.

[5] S. Rougeaux and Y. Kuniyoshi, "Robust tracking by a humanoid vision system," in *Proc. IAPR First Int. Workshop on Humanoid and Friendly Robotics*, Tsukuba, Japan, 1998.

[6] G. Sandini and G. Metta, *Sensors and Sensing in Biology and Engineering*. Wien-New York: Springer-Verlag, 2003, ch. Retina-like sensors: motivations, technology and applications.

[7] G. Engel, D. N. Greve, J. M. Lubin, and E. L. Schwartz, "Space-variant active vision and visually guided robotics: Design and construction of a high-peformance miniature vehicle," in *Proc. 12th IAPR Int. Conf. Pattern Recognition. Vol. 2 - Conf. B: Computer Vision & Image Processing*, Jerusalem, Israel, 1994, pp. 487 – 490.

[8] T. Shibata, S. Vijayakumar, J. Jörg Conradt, and S. Schaal, "Biomimetic oculomotor control," *Adaptive Behavior*, vol. 9, no. 3/4, pp. 189–208, 2001.

[9] F. Panerai, G. Metta, and G. Sandini, "Visuo-inertial stabilization in space-variant binocular systems," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 195–214, 2000.

[10] R. Manzotti, A. Gasteratos, G. Metta, and G. Sandini, "Disparity estimation on log-polar images and vergence control," *Computer Vision and Image Understanding*, vol. 83, no. 2, pp. 97–117, 2001.

[11] G. Metta, F. Panerai, R. Manzotti, and G. Sandini, "Babybot: an artificial developing robotic agent," in *Proc. Sixth Int. Conf. on the Simulation of Adaptive Behaviors (SAB 2000)*, Paris, France, September 2000.

[12] G. Sun and B. Scassellati, "Reaching through learned forward model," in *Proc. IEEE-RAS/RSJ Int. Conf. Humanoid Robots (Humanoids 2004)*, Los Angeles, California, USA, 2004.

[13] C. Gaskett, A. Ude, and G. Cheng, "Hand-eye coordination through endpoint closed-loop and learned endpoint open-loop visual servo control," *International Journal of Humanoid Robotics*, vol. 2, no. 2, pp. 203–224, 2005.

[14] G. Cheng, A. Nagakubo, and Y. Kuniyoshi, "Continuous humanoid interaction: An integrated perspective – gaining adaptivity, redundancy, flexibility – in one," *Robotics and Autonomous Systems*, vol. 37, pp. 161–183, 2001.

[15] A. Ude and C. G. Atkeson, "Online tracking and mimicking of human movements by a humanoid robot," *Advanced Robotics*, vol. 17, no. 2, pp. 165–178, 2003.

[16] S. Vijayakumar, J. Conradt, T. Shibata, and S. Schaal, "Overt visual attention for a humanoid robot," in *Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Maui, Hawaii, USA, 2001, pp. 2332–2337.

[17] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, Nevada, 2003, pp. 2161–2166.

[18] A. Ude, T. Shibata, and C. G. Atkeson, "Real-time visual system for interaction with a humanoid robot," *Robotics and Autonomous Systems*, vol. 37, no. 2-3, pp. 115–125, 2001.

[19] G. Metta, A. Gasteratos, and G. Sandini, "Learning to track colored objects with log-polar vision," *Mechatronics*, vol. 14, pp. 989–1006, 2004.

[20] A. Ude, C. G. Atkeson, and G. Cheng, "Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act," in *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, Nevada, 2003, pp. 2173–2178.

[21] A. M. Arsenio, "Object recognition from multiple percepts," in *Proc. IEEE-RAS/RSJ Int. Conf. Humanoid Robots (Humanoids 2004)*, Los Angeles, California, USA, 2004.

[22] A. Ude and C. G. Atkeson, "Probabilistic detection and tracking at high frame rates using affine warping," in *Proc. 16th Int. Conf. Pattern Recognition, Vol. II*, Quebec City, Canada, 2002, pp. 6–9.

[23] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[24] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE J. Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[25] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[26] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proc. R. Soc. of London, B*, vol. 200, pp. 269–294, 1978.

[27] T. Poggio and S. Edelman, "A network that learns to recognize three-dimensional objects," *Nature*, vol. 343, pp. 263–266, 1990.

[28] H. C. Longuet-Higgins, "Recognizing three dimensions," *Nature*, vol. 343, pp. 214–215, 1990.

[29] P. Sinha and T. Poggio, "Role of learning in three-dimensional form perception," *Nature*, vol. 384, pp. 460–463, 1996.

[30] M. J. Tarr and H. H. Bülthoff, "Image-based object recognition in man, monkey, and machine," *Cognition*, vol. 67, no. 1-2, pp. 1–20, 1998.

[31] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[32] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 775–779, 1997.

[33] M. Pontil and A. Verri, "Support vector machines for 3D object recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 6, pp. 637–646, 1998.

[34] G. Guo, S. Z. Li, and K. L. Chan, "Support vector machines for face recognition," *Image and Vision Computing*, vol. 19, no. 9-10, pp. 631–638, 2001.

[35] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[36] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999.

[37] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. Twenty-first Int. Conf. Machine Learning*, Banff, Alberta, Canada, 2004, article No. 104.

[38] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 121–167, 1998.

[39] C. Wallraven, B. Caputo, and A. Graf, "Recognition with local feature: the kernel recipe," in *Proc. Ninth IEEE Int. Conf. Computer Vision*, Nice, France, 2003, pp. 257–264.

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.

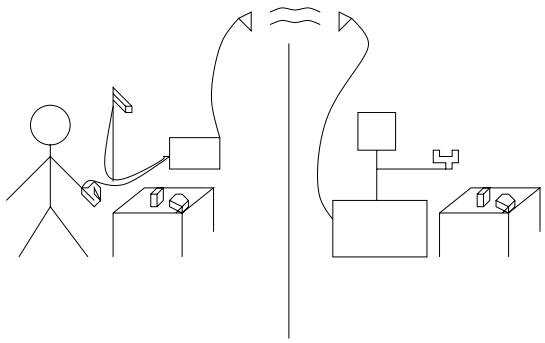


Fig. 1. Left: A human demonstrates object manipulation tasks to the robot. A camera and data glove equipped with magnetic trackers provide sensory input for task recognition. Right: The robot uses this information to reproduce the demonstrated task using its own frame of reference.

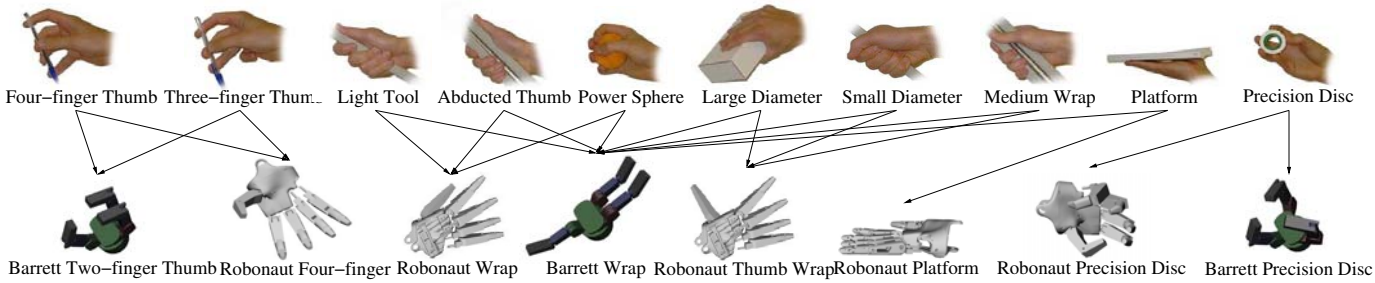


Fig. 2. Initial robot hand postures for different grasp types.

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].

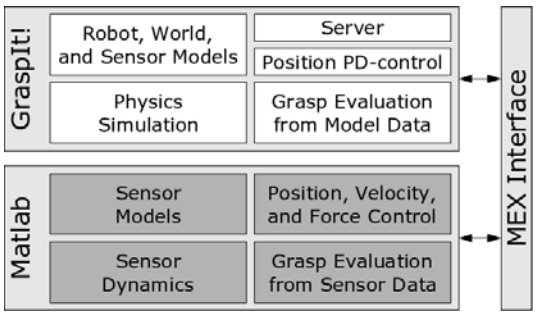


Fig. 3. Software layout for the simulation environment.

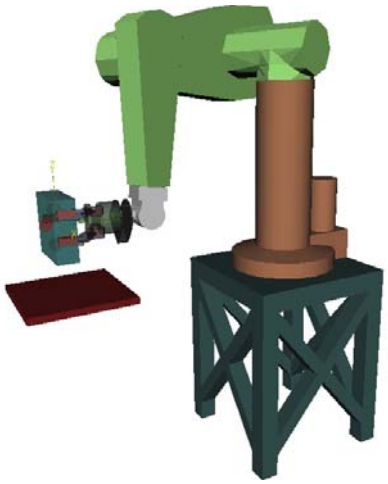


Fig. 5. The box is grasped and lifted by the Barrett hand mounted on a Puma arm.

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:

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

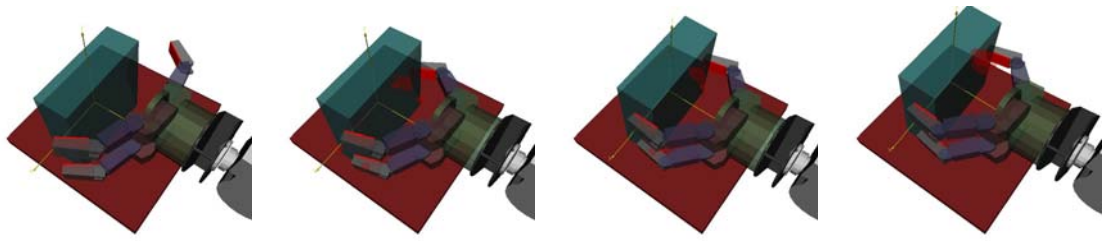


Fig. 4. Execution of a sample task where *corrective movements* are used to center the object.

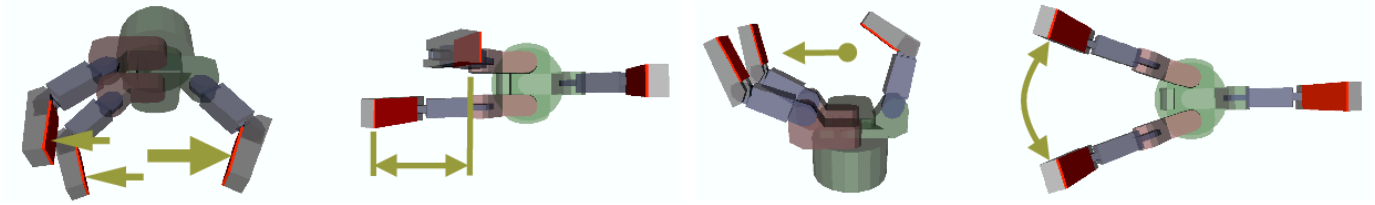


Fig. 6. Grasp controllers: total grasp force, stability, centering, and spread.

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

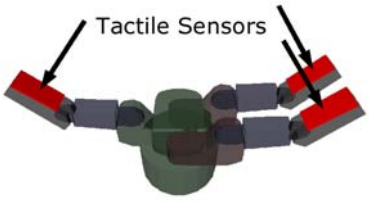


Fig. 7. The placement of the tactile sensors.

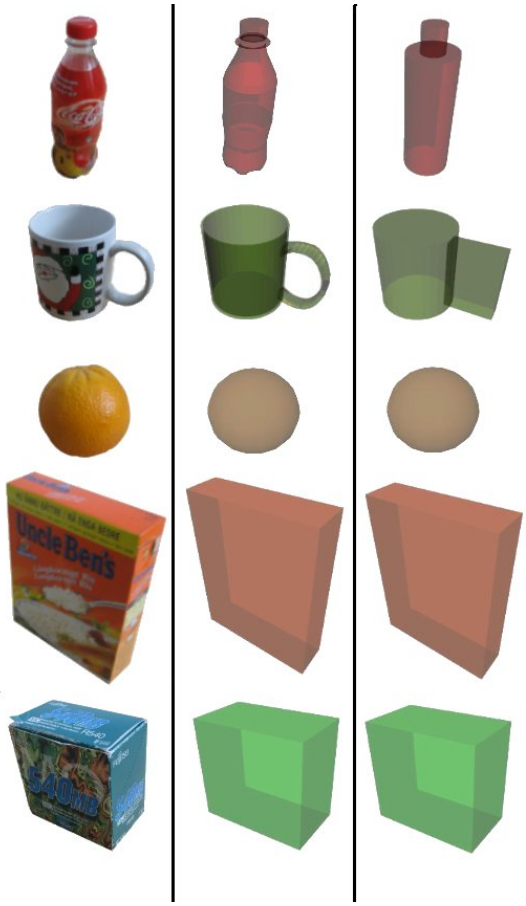


Fig. 8. Left: The real objects. Center: The modelled objects. Right: The object primitives used for training.

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.

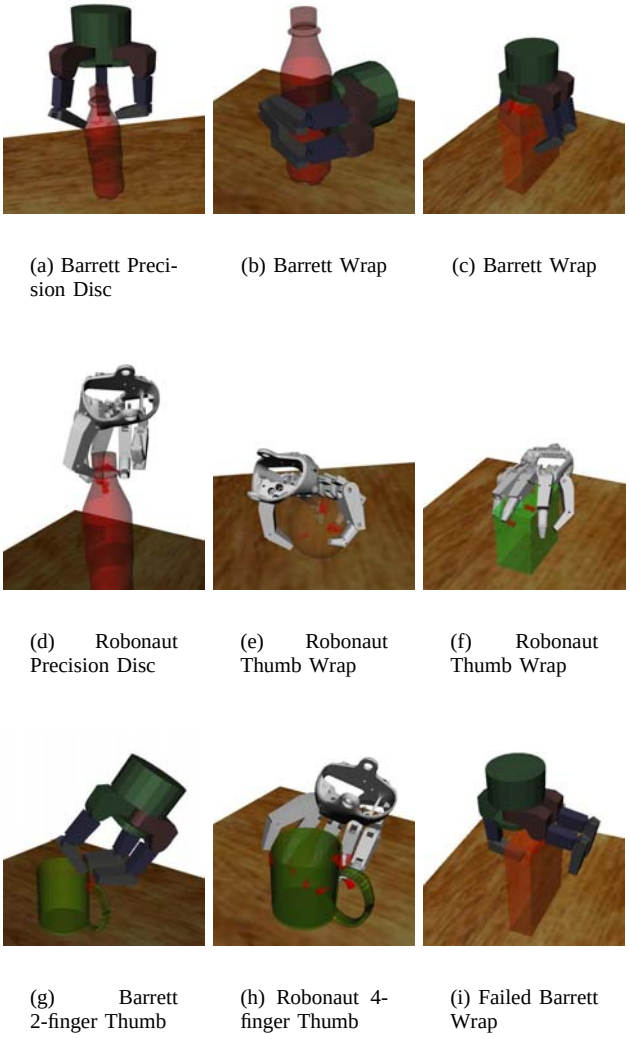
The human teacher, wearing a data-glove with magnetic trackers, moves one object. The move is recognized by the vision system and so is the grasp the teacher used. This information is used to generate a suitable robot grasp (grasp mapping) that controls the movement of the robot hand in the simulator.

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

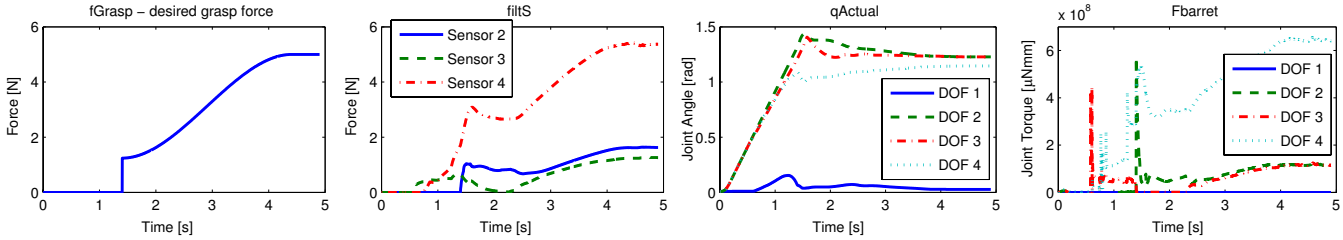
To evaluate the performance under imperfect pose estimation, we have simulated errors in pose estimation by providing an object pose with an offset. As pointed out in [35], the robustness of a grasp to positioning the end-effector has not been widely addressed in the literature.

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.

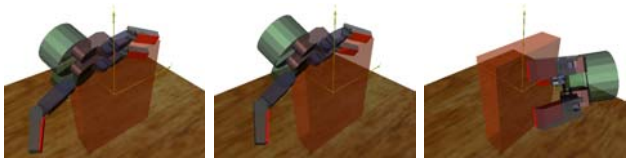
It is clear that the Barrett hand is more robust than the Robonaut hand, likely due to its long fingers. The exception is the grasping of the mug, Fig. 14, where the Robonaut Four-finger Thumb grasp is the best.

The bottle and the mug have been trained both using a

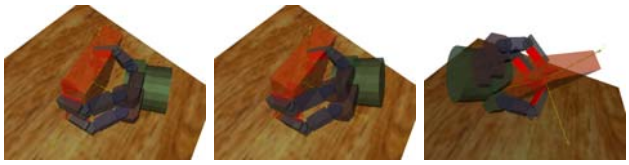


(a) Desired grasp force (b) Tactile sensors - filtered (c) Joint angles (d) Joint torques

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.

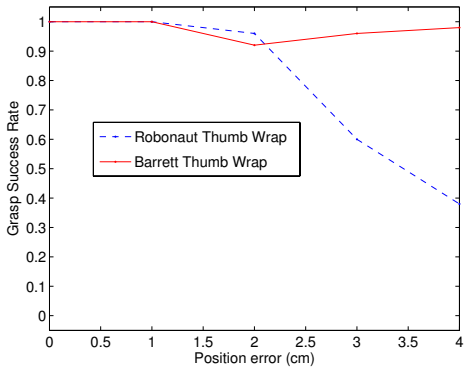


Fig. 15. Grasping the orange.

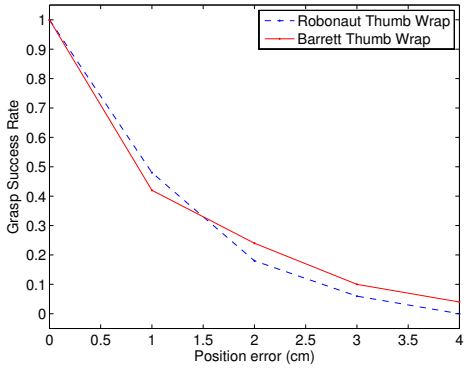


Fig. 16. Grasping the zip disc box.

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

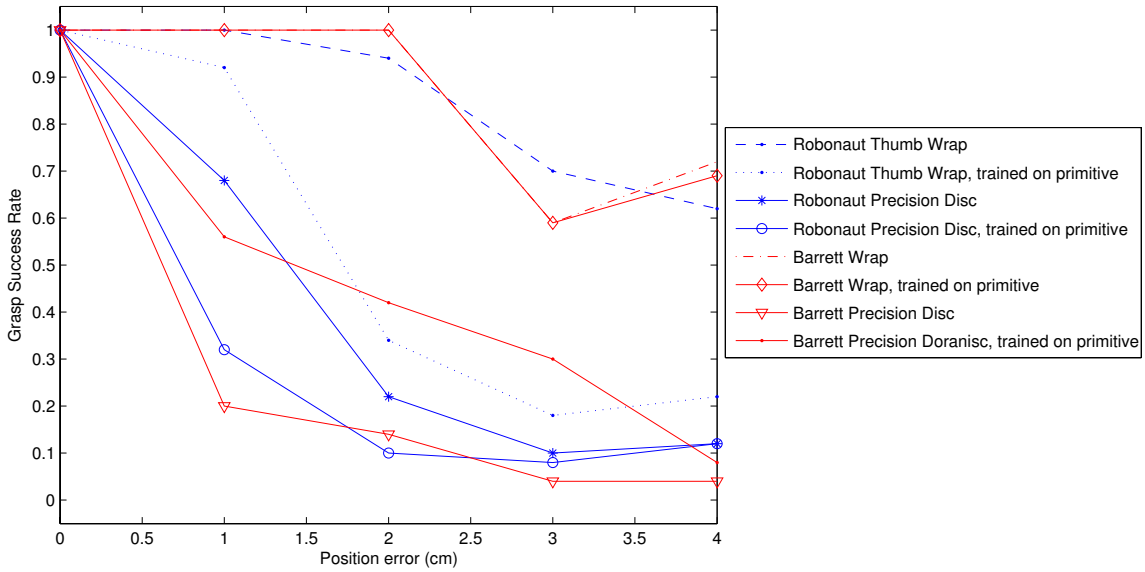


Fig. 13. Grasping the bottle.

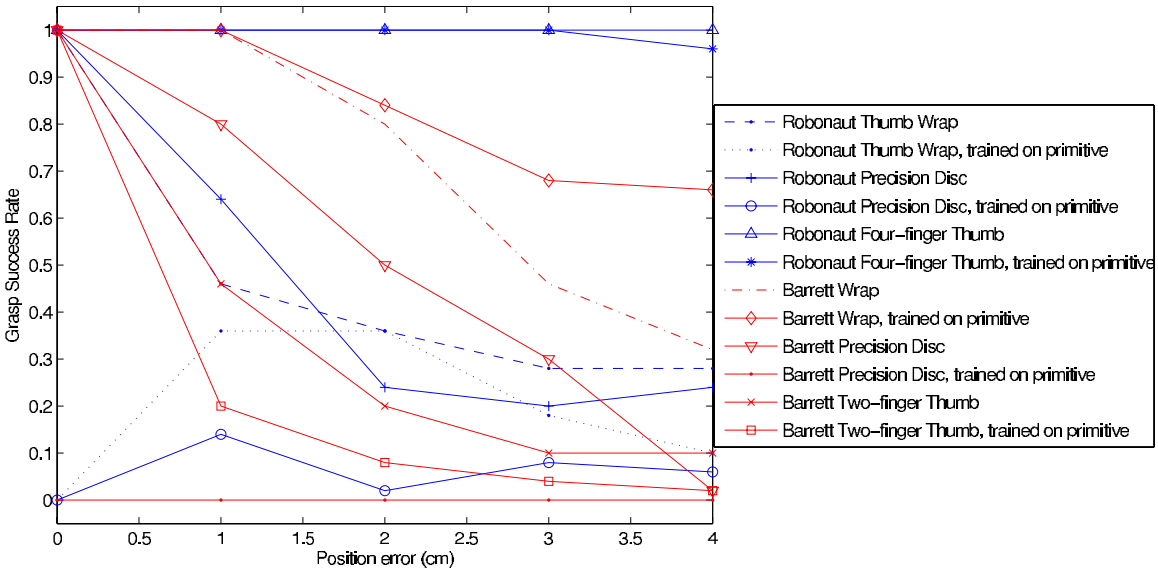


Fig. 14. Grasping the mug.

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;

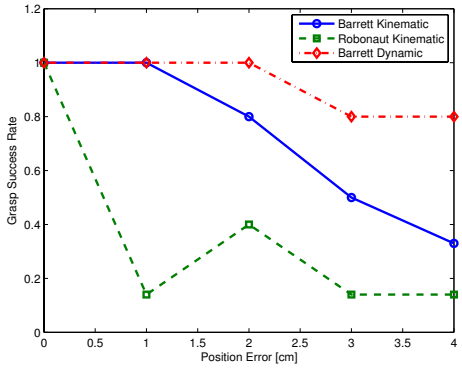


Fig. 17. Grasp success rates for different controllers and simulations. The dynamic grasp is the one from Fig. 11 c) and f). (Due to some problems with the simulator, a limited number of samples were used in the evaluation of dynamic grasping. For the 0, 1, 2, 3, and 4 cm random displacement, the number 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.

Action Recognition and Understanding Through Motor Primitives

Isabel Serrano Vicente†, Ville Kyrki‡, Danica Kragic† and Martin Larsson†

† *Royal Institute of Technology, Computational Vision and Active Perception lab, Centre for Autonomous Systems, S-100 44 Stockholm, Sweden*

‡ *Lappeenranta University of Technology, Department of Information Technology, Finland*

isabelsv@nada.kth.se,kyrki@lut.fi,danik@nada.kth.se,martinla@nada.kth.se

Abstract

In robotics, recognition of human activity has been used extensively for robot task learning through imitation and demonstration. However, there has not been much work on modeling and recognition of activities that involve object manipulation and grasping. In this work, we deal with single arm/hand actions which are very similar to each other in terms of arm/hand motions.

The approach is based on the hypothesis that actions can be represented as sequences of motion primitives. Given this, a set of 5 different manipulation actions of different levels of complexity are investigated. To model the process, we are using a combination of discriminative support vector machines and generative hidden Markov models. The experimental evaluation, performed with 10 people, investigates both definition and structure of primitive motions as well as the validity of the modeling approach taken.

keywords: action recognition, primitive actions, Hidden Markov Models, Support Vectors Machines, object manipulation

1 INTRODUCTION

Neuroscientific and psychological literature states that the core of developmental learning in humans is by watching another person performing a task. This has also motivated the research in the robotics area of learning by imitation and robot programming through demonstration. There is an extensive amount of work dealing with issues of *what*, *when* and *how* to imitate.

Human-computer interaction, surveillance, video retrieval are just a few examples of areas that require human activity recognition [1]. In robotics, recognition of human activity has been used extensively for robot task learning through imitation and demonstration, [14, 23, 3, 18, 19, 16, 13, 7, 4]. For humans, one of the fundamentals of social behaviors is the understanding of each others' intentions

through perception and recognition of performed actions. However, the neural and functional mechanisms underlying this ability in human are still poorly understood [11] which makes it difficult to develop the necessary models needed for designing a robot system that can learn just by observing a human or another robot performing an action. The recent discovery of *mirror neurons* in monkey's brain [22, 9] has nevertheless introduced new hypotheses and ideas about the process of imitation and its role in the evolution.

It has been shown in [8] that an action perceived by a human can be represented as a sequence of clearly segmented *action units*. This motivates the idea that the action recognition process may be considered as an interpretation of the continuous human behaviors which, in its turn, consists of a sequence of action primitives [13] such as *reaching*, *picking up*, *putting down*. In relation, learning *what* and *how* to imitate has been recognized as an important problem, [4]. It has been argued that the data used for imitation has statistical dependencies between the activities one wishes to model and that each activity has a rich set of features that can aid both the modeling and recognition process.

Most of the actions that the future service robot needs to perform are non-cyclic in nature. In this work, we are investigating non-cyclic actions, with a focus on manipulation actions, which have not been studied extensively earlier. The specific questions that the study aims to answer are: 1) Can individual actions be considered as manipulation primitives? 2) If not, can these be broken down into primitives? and 3) How can new actions emerge from known primitives? For this purpose, we consider five different manipulation actions performed on an object: a) pick up, b) rotate, c) push forward, d) push to side, and e) move to side by picking up. To increase the variability, each action is performed by 10 different people in 12 different conditions. We strongly believe that the findings of the study will facilitate imitation learning in robots, both in terms of what vocabulary of primitives to learn and how to combine the individual primitives in order to form more complex actions.

To model the process, we are using a combination of discriminative and generative models. A support vector machine (SVM) is used to model and recognize individual primitives, while the sequences of primitives are modeled using a hidden Markov model (HMM). The measurements are based on magnetic pose sensors. Experimental evaluation demonstrates the feasibility and validity of the adopted approach.

This paper is organized as follows. First, we review related work in Sec. 2. Then, the theoretical basis for the work and two different approaches for primitive based modeling of manipulation actions are described in Sec. 3. Section 4 describes our experimental system. Experiments and their results are reported in Sec. 5. Finally, the results are discussed and a conclusion given in Sec. 6.

2 RELATED WORK

In [18, 19], a framework for acquiring hand-action models by integrating multiple observations based on gesture spotting is proposed. [16] present a gesture imitation system where the focus is put on the coordinate system transformation so that the teacher induced gesture is transformed into the robot's egocentric system. This way the robot observes the gesture as it was generated by the observer himself.

[13] approaches the task learning problem by proposing a system for deriving behavior vocabularies or simple action models that can be used for more complex task extraction and learning. [4] presents a learning system for one and two-hand motions where the robot's body constraints are considered as a part of the optimal trajectory generation process. An interesting trend to note here is that most of the studies are based on a single user generated motion. A natural question to pose here is how the underlying modeling methods scale and apply for cases when the robot is supposed to learn from multiple teachers. The experimental evaluation conducted in our work is based on 10 people.

Related to the theoretical framework used in this work, support vector machine (SVM) has been applied to several different application areas. Two very common data types are visual and speech data [10, 5, 2]. SVMs have also been used with success in computational biology, for example in protein classification [15]. Most of the work dealing with SVMs and time-series data has been done in speech recognition. Earlier work with SVMs [10] presented one drawback when working with sequential data, namely that SVM lacks a way of handling the time dependencies in the data. In order to use time sequences as SVM input, variable length time sequences can be either normalized to same length before applying the SVM. Another approach is to embed dynamic time warping (DTW) directly into the SVM kernel function [24]. Third, probably most common way to handle the "time problem" is to combine a SVM with Hidden Markov Models (HMM) [2, 10, 25]. SVM is still used to classify single points or brief time windows, but the output of the SVM is then used an input to a HMM which then finds the most probably path or sequence in consideration of time. It is also well known that the choice of the SVM kernel function has a significant effect on the results but unfortunately the best choice is application dependent.

In action recognition and understanding, it is most common to take a holistic approach, that is, to consider all measurements as a single feature. This in contrast to speech recognition where it is common to divide the data into individual phonetics or words. From the point of view of imitation learning or "learning by showing", the primitives are an attractive option since they can alleviate mapping motion from humans to robots which differ in their embodiment. In addition, having a common vocabulary of primitives can aid in task understanding and planning as the task can be then described as a sequence of events. For this reason, we now concentrate on this body of work. Ogawara et al. [20] propose to extract primitive actions by learning several HMMs and then cluster these HMMs such that each cluster represents one primitive. Thus, variability within each primitive can be modeled as each cluster can contain several examples. Vecchio et al. [6] model two-dimensional drawing actions as dynamical systems and classify and segment motions according to a priori known motion classes. Representation and segmentation of repetitive movements has been studied by Lu et al. [17] using an auto-regressive model and detecting changes in the model parameters. Finally, stochastic parsing has been proposed for primitive-based action recognition and understanding [12, 26].

3 MODELING METHODS

Next, we present the theoretical basis on recognizing individual primitives using SVMs and the time sequence modeling using hidden Markov models. Then, two different approaches of primitive based modeling of manipulation actions are described.

3.1 Support vector machines

Support vector machines (SVMs) are a popular margin maximizing classification method for tasks involving two or more classes. The aim of support vector classification is to separate two classes, mapped into a high dimensional feature space, by a hyperplane with a maximal margin to both classes. The hyperplane is the decision boundary of the classifier with feature vectors on one side belonging to a first class and vectors on the other side to a second one. To represent complex decision boundaries, the mapping from the original feature space to the high dimensional space is nonlinear. Instead of using the nonlinear mapping explicitly, a kernel function can be used to implicitly map from the original feature space to the high dimensional space. This makes the use of high dimensional mappings computationally feasible.

Let us define the input data as a set of N feature vectors $\mathbf{x}_i$ which belong to either of two classes. The dataset can then be written as $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ where $y_i \in \{-1, 1\}$ represents the class corresponding to $\mathbf{x}_i$.

The classifier having maximum margin to both classes can be discovered by solving the constrained minimization problem

$$Q(\boldsymbol{\alpha}) = \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{k=1}^{N} \sum_{j=1}^{N} \alpha_k \alpha_j y_k y_j K(\mathbf{x}_k, \mathbf{x}_j) \tag{1}$$

subject to constraints

$$\forall k : 0 \leq \alpha_k \leq C, \quad \sum_{k=1}^{N} \alpha_k y_k = 0. \tag{2}$$

Here, α_i are the support vector weights, which represent the contribution of each training sample to the resulting decision boundary. Each sample $\mathbf{x}_i$ with α_i greater than zero is called a support vector as it affects the classification result. $K(\cdot, \cdot)$ is the kernel function corresponding to the dot product of two vectors in the high dimensional space, and C is a penalty parameter for misclassified samples.

The kernel function used in this work is the Gaussian kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \tag{3}$$

where σ is the bandwidth parameter of the Gaussian kernel. Using the kernel function, the classification is performed by

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \tag{4}$$

where $\mathbf{x}$ is the feature vector to be classified and $b = 1 - \sum_{k=1}^{N} \alpha_k y_k K(\mathbf{x}_1, \mathbf{x_k})$ where $\mathbf{x}_1$ is any of the support vectors belonging to class 1.

To extend the above for more than two classes, we take the one-against-one approach. That is, by denoting the number of classes by k, $k(k-1)/2$ classifiers are trained using all pairs of classes. To classify a sample from an unknown class, it is classified by all classifiers, and each result is a vote for the class. Majority voting is then used to decide the class of the sample. The one-against-one approach has been found very successful with SVMs but it suffers from increased number of individual classifiers when the number of classes is very high.

3.2 Markov chain and hidden Markov models

A hidden Markov model (HMM) is one of the most common statistical models for time-series data, having applications in speech, gesture, and handwriting recognition, as well as bioinformatics. An HMM can be considered a probabilistic version of a finite state machine. The time evolution of states is modeled as a Markov chain, a discrete-time stochastic process with the Markov property, that is, the probability distribution of the future states depend only on the current state and not on any of the past states. In this work, we are interested in time-homogeneous Markov chain models, that is, the state transition probabilities are invariant over time. Denoting the state i by ω_i, the time evolution of states can then be described using the state transition probabilities $P(\omega_j(t+1)|\omega_i(t)) = a_{ij}$. The states themselves are hidden, not directly observable. Instead, in each state, an observation $\mathbf{x}(t)$ is made. The observation depends only on the current state according to a selected probabilistic model, that is, $P(\mathbf{x}(t)|\omega_i(t)) = P(\mathbf{x}|\omega_i)$. If the set of observations X is discrete and finite, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the observation probabilities can be written more shortly as $P(\mathbf{x}_j|\omega_i) = b_{ij}$. Finally, the probability of starting in state ω_i can be defined as $\pi_i = P(\omega_i(1))$. Thus, the parameters can be collected to matrices $\mathbf{A}$ and $\mathbf{B}$ and a vector $\boldsymbol{\pi}$.

In this study, the objective is to model actions based on motor primitives. The motor primitives correspond to individual states of the HMM. A typical approach for using HMMs in recognition is to build a single HMM for each class to be recognized and then determine the class of an unknown sample by using the maximum likelihood method to identify the most likely class. In this work, we take another approach and represent the whole set of actions with a single HMM, such that different paths through the HMM correspond to different actions. This is because many actions contain similar parts. For an example, see Fig. 1 where both rotating and pushing an object both require first the hand to approach the object. Our hypothesis is also that more complex actions can be modeled using a set of motor primitives. Thus, in recognition, instead of making a choice between several HMMs, the most probable path through the HMM is sought.

The Viterbi algorithm [21] is a dynamic programming based algorithm for determining the maximum likelihood path through a HMM given a sequence of observations $(\mathbf{x}(1), \mathbf{x}(2), \ldots)$. That is, it finds the state sequence $(\omega(1), \ldots)$ for which

$$P\left(\mathbf{x}(1), \ldots, \mathbf{x}(T)|\omega(1), \ldots, \omega(T)\right) \tag{5}$$

is maximal. The solution by enumerating all possible state sequences is not computationally tractable.

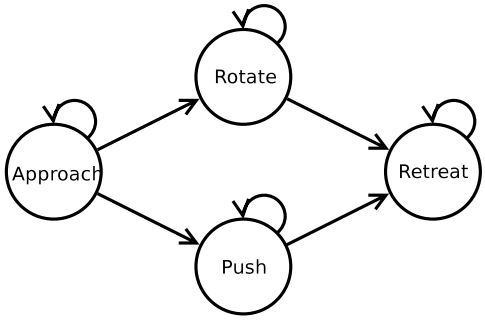


Figure 1: Modeling two actions (rotate, push) using primitives.

Instead, the solution is based on defining (5) recursively as

$$P\left(\mathbf{x}(t), \mathbf{x}(t-1), \ldots | \omega(t), \omega(t-1), \ldots\right) = P\left(\mathbf{x}(t) | \omega(t)\right) P\left(\omega(t) | \omega(t-1)\right) P\left(\mathbf{x}(t-1), \ldots | \omega(t-1), \ldots\right) \tag{6}$$

and noting the optimal substructure of the problem. The expression $P\left(\mathbf{x}(t)|\omega(t)\right)P(\,\omega(t)|\omega(t-1))$ defines a cost term for a single state transition. Therefore, the optimization problem can be transformed to minimum length path search, solvable in $O(TN^2)$ time.

The most common approach to learn HMMs is the Baum-Welch algorithm, an iterative expectation maximization (EM) approach to learn the observation and transition probabilities. However, the approach is only guaranteed to converge to a local optimum, not a global one. In this study, we take an alternative approach. We use labeled examples as training data, that is, for each time step, the current motor primitive is known. Then, the transition probability matrix $\mathbf{A}$ can be directly estimated from the training data, as if in the case of Markov chain model instead of a HMM. We use the maximum likelihood estimate, in other words, the transition probabilities are calculated directly from the training data. The output of the SVM is used as the observations of the HMM. The observation probabilities need also be estimated as it is not expected that the classifier will be able to classify all samples correctly. Maximum likelihood estimation using the known correct classes is also used to estimate the observation probabilities. Therefore, the observation matrix $\mathbf{B}$ corresponds to the confusion matrix of the classifier.

3.3 Action modeling

The hypothesis in the modeling is that each of the manipulation primitive is generic and that their number is limited. The limited number of primitives is supported, for example, by the knowledge that a limited number of different grasp types are possible. However, the best applicable set of primitives is not known and one of the goals of this study is to inspect, how the manipulation actions can be considered in terms of primitives.

We investigate two different models of action representation. These are shown in Fig. 2. Approach 1 considers each of the manipulation actions as a primitive. In addition to the manipulation actions, two assisting actions, *approach* and *remove* are inherent in all action sequences (see Fig. 2). The assisting

actions alleviate the segmentation of the manipulation part of the action. Approach 2 considers that the manipulation part of the action can be composed of multiple primitives. The model on right in Fig. 2 can be chosen based on the knowledge that the rotation and moving the object require grasping. Our working hypothesis is that Approach 2 would be more effective in recognizing actions compared to the first approach. In addition it would allow learning of new actions based on the known primitives.

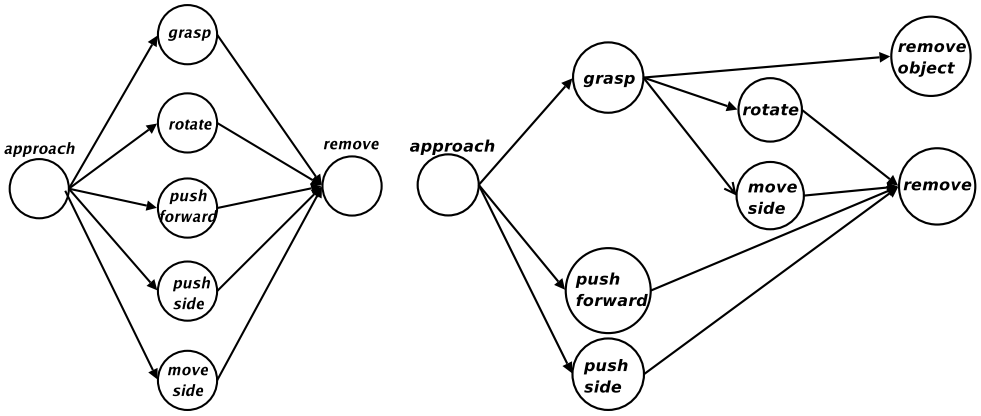


Figure 2: (left) Approach 1: Actions as primitives; (right) Approach 2: Composite actions.

In both approaches, each action is represented by a separate path through the left-to-right Markov model. Considering Approach 2, to learn a new composite action, it is enough to learn the new sequence of primitives, if the primitives are already known. If a hypothesis of the sequence (and order) of primitives is available, the only parameters that have to be learned are the transition probabilities of the model. However, having an unknown sequence, the only available information is the sequence of observations (SVM output) which contains uncertainty. As the transition probabilities are inherent to the underlying hidden states, not the symbols that are observed, the learning must be performed by considering the Baum-Welch re-estimation (forward-backward algorithm) in the case of hidden Markov models [21]. It should be noted that by initializing the estimation with non-zero probabilities only along the desired path, the estimation process will find the locally optimal probabilities within the path such that no new states will be introduced. If the observation probabilities of the primitives are also known in advance, only the transition matrix of the HMM needs to be updated in the estimation.

Upper part of Fig. 3 shows the composite action model without the *move to side* primitive. The lower part of the same figure demonstrates now a single possible representation of the *move to side* primitive. Note that now the new primitive is described fully by existing primitives. The transition probabilities for the new primitive can be estimated as discussed above. After learning a model for a new action, the state transition probabilities of the model containing all actions must be updated according to that of the new action. During the process, new state transitions will be introduced in the model. This is illustrated in Fig. 4. The probabilities can be updated by weighted averaging of the transition probabilities from a state given the two models, with weights given by the number of actions using that state in that particular model. Thus, the upper model of Fig. 3 would have twice the weight compared

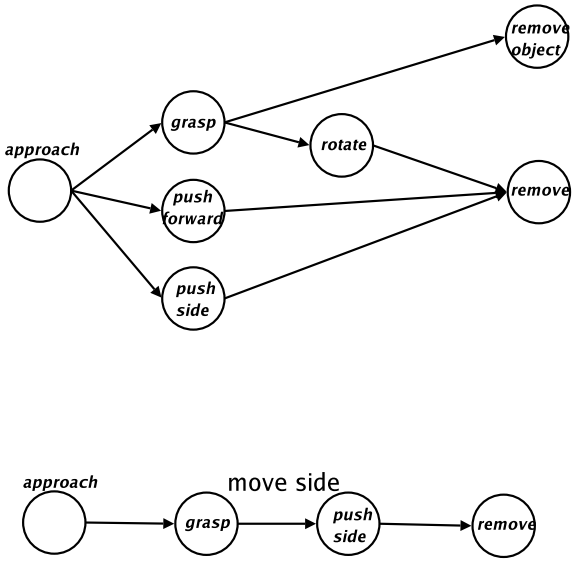


Figure 3: Learning new composite actions.

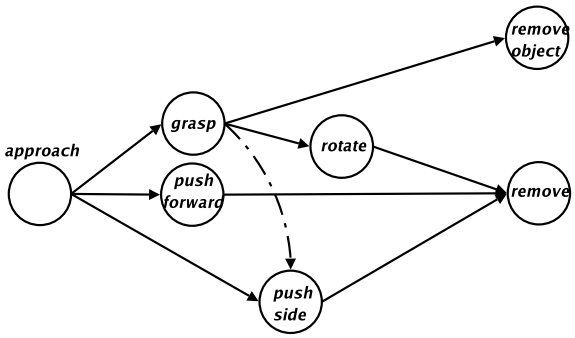


Figure 4: Embedding a new action.

to the lower one for paths leaving *grasp* because in the upper one there are two actions using the state.

To determine the best sequence of primitives for a new action, exhaustive search can be used if the number of primitives is relatively low. Otherwise, search and pruning techniques would be necessary. However, the classification results of individual time instants give a strong cue as to which primitives are present in an unknown action.

4 SYSTEM AND IMPLEMENTATION

The approach for action recognition and understanding is next described, starting with the description of the sensors and the modeled actions. Then, system overview is presented and finally, implementation details are described in more detail.

4.1 Sensors and data

Our aim is to study the modeling and understanding of manipulation actions performed by humans. Five different actions are considered: a) pick up an object from a table, b) rotate an object on a table, c) push an object forward, d) push an object to the side, and e) move an object to the side by picking it up.

To include variation in the actions, each action is performed in 12 different conditions, namely on two different heights, two different locations on the table, and having the demonstrator stand in three different locations (0, 30, 60 degrees) (see Fig. 5). Furthermore, all actions are demonstrated by 10 different people.

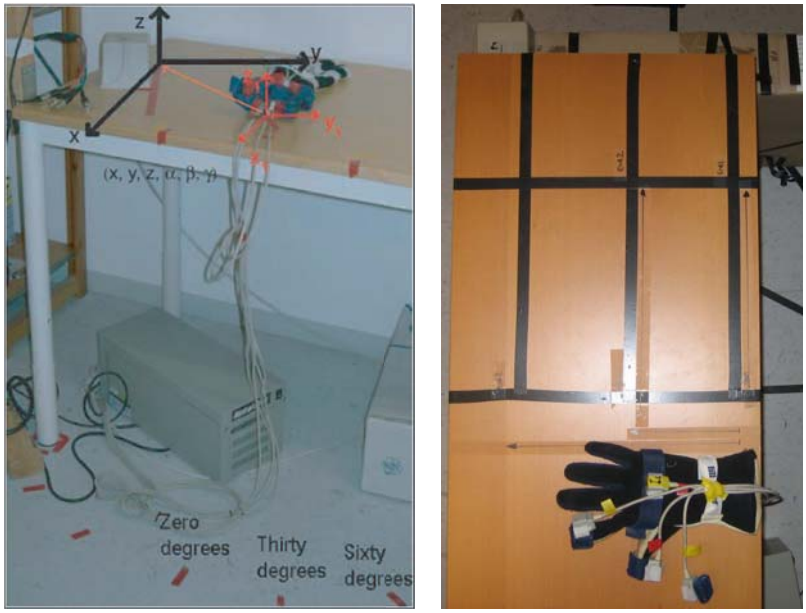


Figure 5: (left) The demonstrator locations; (right) Glove with sensors attached and markers on the table.

The movement is measured using the Nest-of-Birds magnetic sensors. The test subject is endowed with four sensors each registering their full 3-dimensional pose with respect to a reference, which can be seen in the upper left corner of Fig. 5. The sensors are located on: a) chest, b) back of hand, c) thumb, and d) index finger. Figure 6 show the positioning of the sensors. The chest sensor is used to provide a reference to the demonstrator position while the back of the hand can be used as a reference for the thumb and index finger. The measured sequences have been annotated by hand such that the current action primitive is known for training.

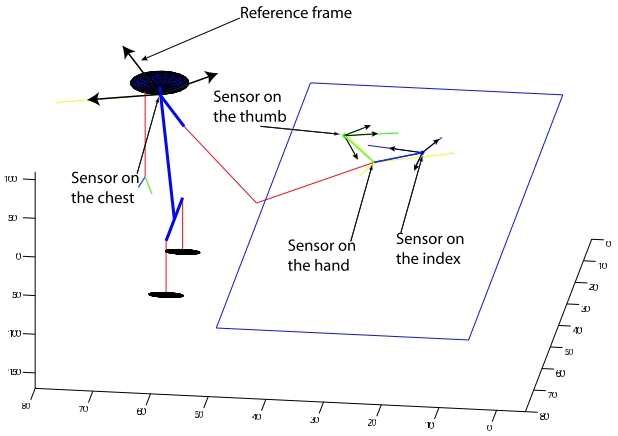


Figure 6: Sensor locations.

4.2 System overview

The goal of the system is to recognize actions, while this study also tries to reveal, how suitable primitive based techniques are for action description of manipulation actions. An overview of the system is given in Fig. 7. First step is to preprocess the data for noise removal. This is necessary as the sensor measurements are corrupted by spurious noise peaks. The primitives are recognized by an SVM and its output is then fed to an HMM which describes the time evolution. SVMs were chosen as they have been demonstrated with great success in many multidimensional classification problems where the training set is relatively sparse. As the true action primitives are known, SVMs can be directly trained. Regular and hidden Markov models are then used to describe the temporal sequence of primitives. Regular Markov models can be used in the training phase since the true class is observable, while in the test phase the action is recognized by the Viterbi algorithm as the true states are then hidden and only the SVM output is available. The lower part of the system in Fig. 7 is concerned with the recognition of new actions based on known primitives. In that case, the models are learned through the standard Baum-Welch re-estimation process of HMM learning.

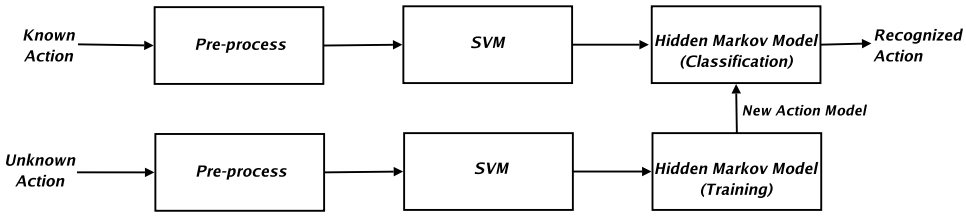


Figure 7: System Overview.

4.3 Pre-processing

We are not using all of the 24 measurements from the Nest-of-Birds sensor were not used because they are highly redundant. For example, the thumb position with respect to the back of the hand correlates with the orientation of the hand. To describe temporal trajectories, also the velocity of the hand was estimated. Thus the following 12 measurements were used:

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

Starting from the raw data, the procedure illustrated in Fig 8 was used to preprocess the data before SVM classification. First, median filter was applied for both the position and the orientation of the three sensors were filtered with a median filtered so to eliminate the noise peaks. The length of the filter was 7 and it was applied twice. After filtering, the hand and finger locations were transformed into the chest reference frame. Next, the position of both the thumb and index was calculated with respect to the back of the hand. A Gaussian filter was then applied for the finger positions to reduce the noise, which was found to be most apparent in the finger position measurements. The velocity was estimated by time differences between two consecutive time instants. It was then filtered by a Gaussian filter to decrease the noise due to the differential nature of the estimation process. Finally, every dimension was linearly scaled. First, the minimum and maximum value of each dimension was found for each sequence and then the average of the minima and maxima were calculated. Then, the scaling was performed as $x_{scaled} = (x - x_{min})/(x_{max} - x_{min})$.

The effect of the preprocessing before scaling is illustrated in Figs. 9 and 10. Figure 9 demonstrates that while the spurious peaks are removed, the overall shape of the trajectory is not changed. Figure 10 demonstrates the statistics of the index finger location with respect to the hand. The center graph shows the histogram for measurements between -15 and 15 cm while the left (right) graph shows the histogram for measurements under -15 (over 15). The values over 15 and under -15 are measurement outliers which should be removed by the filtering. The lower graph shows that the outliers are removed but that the shape of the histogram for valid measurement values is not changed in filtering.

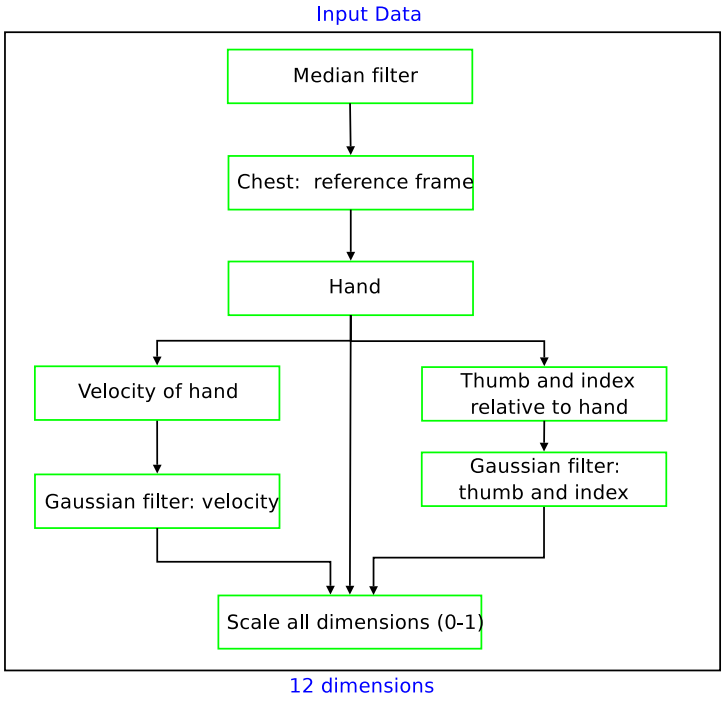


Figure 8: Preprocessing step overview.

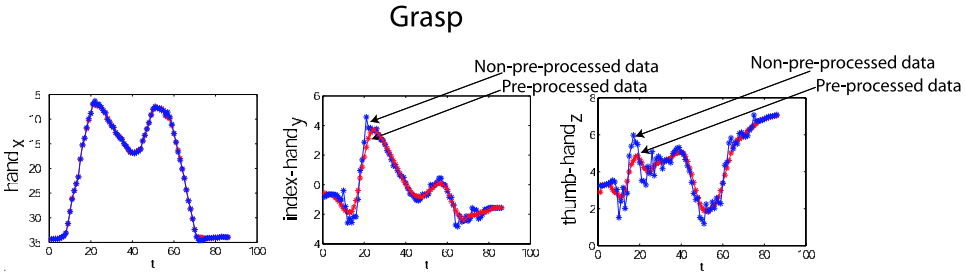


Figure 9: Filtering for noise removal.

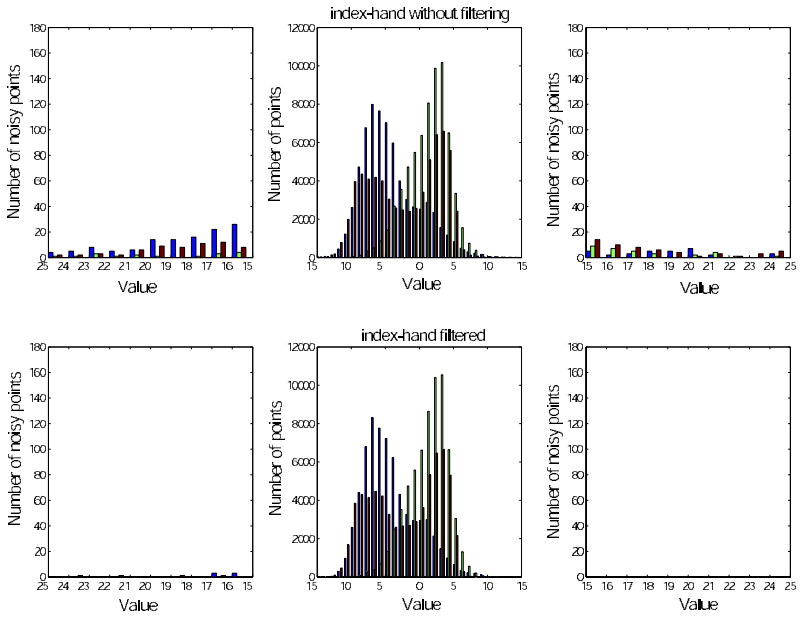


Figure 10: Performance of the pre-processing step before scaling. Upper row: Position of the index (x, y and z) respect to the hand after filtering. Lower row: Position of the index (x, y and z) respect to the hand before filtering.

5 EXPERIMENTS

Experiments and their results are next reported. First, Approach 1, where each action is considered a separate primitive, is considered. Then, the actions are modeled as sequences of primitives, Approach 2. Finally, we study the capabilities of modeling a new action based on learned primitives.

All actions were performed by 10 people in 12 different conditions, such that for each action there were 120 different samples. The demonstrators were given only oral explanations of the task and for that reason, the inter-personal variance in the trajectories was high. This approach was taken to emphasize our goal of understanding actions instead of just tracking the movement. For SVM learning the training sequences were classified and segmented by a human. That result was also used as a ground truth for the experiments. In the following results, leave-one-out testing is always used where not indicated otherwise. Thus, one person was left out of the training set, that person was used to test the system, and this was repeated for all persons. Average performance is then reported.

5.1 Actions as primitives

In this approach (Approach 1), each manipulation action is a separate primitive. In addition, the assisting primitives for approach and remove are present in all actions. The action model used can be seen in Fig. 2. The results of experiments are presented in Fig. 11. The upper table shows the confusion matrix for the SVM classification for each time instant. The rows correspond to the ground truth and

the columns are the SVM output. It can be seen that some primitives (*push forward, rotate, remove*) are classified quite well for even considering only one time instant at a time. In contrast, two primitives, *push side, move side* seem to be overlapping in their representation as they are often confused with each other. This confusion is not surprising as the training data was overlapping for the two different primitives due to the high inter-personal variance of how the actions were performed. For that reason, it is possible that one person's *move side* was very similarly to another's *push side*.

Also the assisting primitives *approach, remove* were confused quite much with each other. A more detailed analysis of the results revealed that this happened particularly when the movement was very slow. This explains the confusion, because with slow movements the velocity can not be estimated reliably enough in order to be used for discriminating between these two. Finally, the *grasp* primitive was confused quite often with *rotate, move side*. This is most likely the result that both of these two primitives also involve grasping. Thus, these primitives can not be recognized reliably considering single time instants.

SVM	approach	push-forward	push-side	rotate	grasp	move-side	remove
approach	62,61	5,2	0,92	4,55	2,57	1,69	22,47
push-forward	1,24	86,05	4,54	0,64	2,84	4,12	0,57
push-side	1,15	13,17	41,75	3,7	6,15	30,63	3,45
rotate	1,56	1,09	4,96	83,29	4,44	3,4	1,27
grasp	0,79	6,09	4,94	10,77	36,1	24,01	17,3
move-side	0,26	4,96	20,13	5,88	7,07	61,11	0,58
remove	0,4	1,77	3,43	3,9	3,25	3,18	84,06

HMM	push-forward	push-side	rotate	grasp	move-side
push-forward	87,5	4,17	0	4,17	4,17
push-side	8,33	48,33	2,5	3,33	37,5
rotate	0,83	2,5	95	1,67	0
grasp	5,83	10	9,17	52,5	22,5
move-side	1,67	24,17	4,17	2,5	67,5

Figure 11: Approach 1. Actions as primitives.

Next, the recognition results were used as an input to the HMM. The results of Viterbi based recognition of actions of the HMM are given in the lower table of Fig. 11. The ground truth is given again in the first column. Note that here each sequence is recognized as belonging to one of the actions instead of labeling all time instants. However, the Viterbi algorithm also gives the most probable primitive for each time instant such that the manipulation part can be segmented from the assisting primitives. The confusion matrix in Fig.11 again supports our earlier results that the pair *push side-move side* is difficult to recognize from each other. However, it can be argued that because also the semantic meanings of the two actions are similar, these errors could be tolerated, at least to some extent, in action understanding. Another finding is that *grasp* action could not be recognized individually as the same primitive also exists in other actions.

5.2 Actions as composites

It is evident from the previous experiment that considering the actions themselves as individual primitives did not yield good results. Next, the actions were modeled in a composite structure of primitives. Our approach was to model the individual primitives such that they had semantic meaning. The model is shown on right in Fig. 2. One new state, *remove with object*, was introduced by the argument that the end state of the environment is different in the case the person is holding the object in the end. This is the end state only for the *grasp* action. In addition, the structure of the model was changed such that all actions requiring grasping employ first the grasp primitive before the second manipulation primitive.

Figure 12 presents the confusion matrix for SVM classification as well as the recognition result by the HMM. The SVM classification results change significantly for two primitives, *grasp, remove*. The results of recognizing *grasp* increase significantly, as it is no longer confused with other actions requiring grasping. Based on this result, we can hypothesize that motion primitives exist and that *grasp* can be considered as one. For the *remove* primitive the recognition rate decreases, because a very similar new primitive *remove with object* was introduced. It should be noted that SVM still confuses *push side* with *move side*.

SVM	approach	push-forward	push-side	rotate	grasp	move-side	remove	remove-object
approach	**62,03**	5,46	0,55	0,21	7,73	0,42	22,24	1,37
push-forward	0,84	**84,06**	4,25	0,56	8,25	1,97	0,07	0
push-side	1,34	12,19	**42,2**	3,03	8,78	29,1	1,09	2,25
rotate	0,49	0,74	4,42	**70,29**	19,31	2,07	1,17	1,51
grasp	4,38	6,4	1,31	3,57	**79,27**	3,88	0,34	0,84
move-side	0,56	3,04	17,77	4,46	6,43	**64,52**	1,7	1,53
remove	8,19	3,11	6,04	6,21	0,28	3,4	**64,99**	7,79
remove-object	2,48	0,1	2,31	1,96	3,24	4,66	22,97	**62,26**

HMM	push-forward	push-side	rotate	grasp	move-side
push-forward	**85**	7,5	5	0,8333	1,6667
push-side	9,1667	**47,5**	4,1667	2,5	36,6667
rotate	0	0	**92,5**	0	7,5
grasp	4,1667	7,5	10,8333	**72,5**	5
move-side	1,6667	10	6,6667	0	**81,6667**

Figure 12: Approach 2: Composite actions.

The confusion matrix for the HMM (Fig. 12) has improved significantly for two actions, *grasp, move side*, compared to Approach 1. For *move side*, this result can be explained by the fact that grasp primitive is required for all actions in this class, making it easier to discriminate between *push side* and *move side*. An important note is that the SVM classification result did not improve from Approach 1, but this results from enforcing a particular time sequence of events for the action. It should be, nevertheless, noted that *push side, move side* are still confused, for the reason given in Sec. 5.1. For *grasp* primitive, the improvement is due to improvement in the SVM classification discussed above.

5.3 Modeling a new action

We now try to investigate if new actions can be modeled using learned primitives. From the earlier results it is known that the *move side* action is similar to *push side*. We performed the investigation by removing the *move side* actions from the training data of the SVM. Thus, the SVM only learned the

other primitives. Our goal was then to see which sequential model using the other primitives would be optimal for modeling the *move side* actions. The experiment was begun by modeling the system (without *move side*) in the way shown in upper part of Fig. 3. Thus, the SVM was also trained without any of the *move side* data. The performance results for this model are shown in Fig. 13. The classification performance improves for those primitives, which were earlier confused with the *move side* primitive.

SVM	approach	push-forward	push-side	rotate	grasp	remove	remove-object
approach	**75,91**	7,68	0,9	0,65	7,83	5,31	1,72
push-forward	0,98	**84,38**	6,78	0,32	7,51	0,04	0
push-side	1,06	13,07	**66,33**	3,66	10,28	2,41	3,19
rotate	1,38	1,2	4,34	**73,26**	18,06	0,59	1,17
grasp	4,2	10,13	2,83	5,84	**75,92**	0,09	0,99
remove	40,98	2,9	10	6,11	0,49	**26,74**	12,77
remove-object	5,27	0,17	6,4	2,35	4,71	10,07	**71,03**

HMM	push-forward	push-side	rotate	grasp
push-forward	**84,1667**	8,3333	5	2,5
push-side	11,6667	**77,5**	7,5	3,3333
rotate	0	0,8333	**99,1667**	0
grasp	3,3333	5,8333	13,3333	**77,5**

Figure 13: Modeling a new action: Before new action.

The best left-to-right state model for *move side* was found among all 3 and 4 state models. The starting state was fixed to *approach* and the end state to *remove* in order to constrain the problem to determining the manipulation primitives used. Exhaustive search was used by enumerating all possible models. Each model was trained using the Baum-Welch re-estimation as described in Sec. 3.3 using all of the *move side* sequences as input. Note that now the sequence was not segmented by hand into primitives but the underlying states were considered hidden, and the SVM confusion matrix in Fig. 13 was used as the model for the measurement uncertainty of the HMM. The goodness of fit for each model was evaluated by calculating the joint probability of observing all the training sequences given the new model, where the forward-algorithm [21] was used for each individual sequence. These results are given in Fig. 14 where the upper part show the log-probabilities for each of the 12 different 3 and 4 state models. The model that fits the data best is *approach - grasp - push side - remove*, shown in the bottom of Fig. 3. This model seems to grasp the semantic meaning of the action very well. If the new model is embedded into the existing HMM, as described in Sec. 3.3, the lower part of Fig.14 presents the classification results of this HMM. The recognition rate of 62.5% is good considering that no data of the action sequences was used in the SVM training.

To further examine the inter-personal variance in motion primitives, we repeated the experiment such that now all persons, including the test person, were used in the training of the SVM. Thus, it was supposed that if the hypothesis of actions consisting of primitives is valid, the recognition rate of individual primitives would increase also for the unknown actions where known primitives are used in unknown contexts. The results of this experiment are shown in Fig. 15. The recognition rate for the *move side* action increased from 62.5% to 77.5%. This result can be considered remarkable because it suggests that to learn good models for complex actions for a wide variety of people, it is important to learn the

HMM	Total probability over 120 sequences					
log10(P(all people|model))	A-G-PS-RT	A-r-ps-rt	A-pf-ps-rt	A-ps-r-rt	A-ps-rt	A-ps-g-rt
Approach-grasp-move side-retreat	-55,3219	-58,3656	-59,6757	-61,1559	-62,3538	-62,5674
log10(P(all people|model))	A-ps-pf-rt	A-g-pf-rt	A-pf-g-rt	A-g-r-rt	A-r-g-rt	A-pf-r-rt
Approach-grasp-move side-retreat	-63,2876	-70,1639	-70,3123	-70,7719	-71,9952	-72,4476
log10(P(all people|model))	A-r-pf-rt	A-g-rt	A-r-rt	A-pf-rt		
Approach-grasp-move side-retreat	-72,8203	-72,9108	-76,6566	-77,2139		

HMM	push-forward	push-side	rotate	grasp	move-side
push-forward	84,1667	5,8333	1,6667	2,5	5,8333
push-side	11,6667	47,5	5	2,5	33,3333
rotate	0	0	95	0	5
grasp	3,3333	5	8,3333	76,6667	6,6667
move-side	5,8333	14,1667	10	7,5	62,5

Figure 14: Modeling new action: Best action, Classification in combined HMM.

individual ways of each person executing a certain primitive and that the sequences of primitives for particular semantic actions can be learned in general from data from other people demonstrating the same action.

SVM+HMM	push-forward	push-side	rotate	grasp	move side
push-forward	95,83334	0,83333	3,33333	0	0
push-side	4,16666	59,16667	0	0,83333	35,83334
rotate	0	0	99,16667	0	0,83333
grasp	0,83333	0	2,5	94,16667	2,5
move side	0	6,66666	10,83334	5	77,5

Figure 15: Modeling new action: Classification with personal learning of primitives.

6 DISCUSSION

In this paper, we have studied the recognition and understanding of manipulation actions performed by humans. While the literature in action recognition is large, there are not many extensive studies on the modeling of the manipulation actions, which have the characteristic of being typically very similar to each other. Similar to some other studies, we have considered a framework where the actions are composed of primitives. However, in contrast to others, we consider two alternative hypotheses: 1) individual actions can be considered manipulation primitives, and 2) manipulation actions should be broken down into primitives. Based on initial results, we have realized that even quite simple manipulation actions consist of several primitives, which, however, might be common with other actions. The idea of composite actions is thus result of initial evaluation of the model "actions as primitives". We have also considered assisting primitives, such as approaching the object, which might not serve directly in the recognition of the action but which still can be useful in segmenting the manipulation.

Rather than using generative models for the whole action, SVM based discriminative models have been used for the recognition of individual primitives. This is because our focus is on action recognition and understanding rather than action synthesis. It should be noted that although in this paper the classification is done each time instant, the considerations apply to the case when short time windows

are used instead of instants. Also, the ideas presented are by no means limited to a particular classifier (such as SVM) for the primitives.

The data for experiments was collected from 10 different demonstrators, each demonstrating the actions in several different conditions, and with only an oral explanation of the action given. Thus, the data had significant intra- and inter-personal variation. The most important findings of the experiments are that a) sequences of simple semantic primitives can be used in describing actions, b) inter-personal variations in primitives are significant, and c) actions learned as sequences of primitives from other demonstrators can be combined with knowledge of personal primitives to recognize new actions.

Future work will study what new actions can be modeled with our current primitives, and more importantly, what set of primitives would be appropriate to model a large variety of manipulation tasks typically performed by humans. Finally, we hope to study the model in the context of visual data.

REFERENCES

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding: CVIU*, 73(3):428–440, 1999.

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

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

[4] Sylvain Calinon, Aude Billard, and Florent Guenter. Discriminative and adaptative imitation in uni-manual and bi-manual tasks. In *Robotics and Autonomous Systems*, volume 54, 2005.

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

[6] D. Del Vecchio, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics-bbased primitives with application to drawing tasks. *Automatica*, 39(12):2085–2098, 2003.

[7] Staffan Ekvall and Danica Kragic. Grasp recognition for programming by demonstration tasks. In *IEEE International Conference on Robotics and Automation, ICRA'05*, pages 748 – 753, 2005.

[8] D. Newton et al. The objective basis of behavior unit. *Journal of Personality and Social Pshychology*, 35(12):847–862, 1977.

[9] Luciano Fadiga, Leonardo Fogassi, Vittorio Gallese, and Giacomo Rizzolatti. Visuomotor neurons: Ambiguity of the discharge or 'motor perception'? *International Journal of Psychophysiology*, 35(2-3):165–177, 2000.

[10] Steven E. Golowich and Don X. Sun. A support vector/hidden Markov model approach to phoneme recognition. In *ASA Proceedings of the Statistical Computing Section*, pages 125–130, 1998.

[11] Marco Iacoboni, Istvan Molnar-Szakacs, Vittorio Galles, Giovanni Buccino, John Mazziotta, and Giacomo Rizzolatti. Grasping the intentions of others with one's own mirror neuron system. *PLOS Biology*, 3(3), 2005.

[12] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[13] Odest Chadwicke Jenkins and Maja J. Mataric. Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion. *International Journal of Humanoid Robotics*, 1(2):237–288, Jun 2004.

[14] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching. In *IEEE Transactiond on Robotics and Automation*, volume 10(6), pages 799–822, 1994.

[15] Li Liao and William Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Journal of Computational Biology*, pages 857–868, 2003.

[16] Manuel Cabido Lopes and Jose santos Victor. Visual transformations in gesture imitation: What you see is what you do. In *IEEE International Conference on Robotics and Automation, ICRA04*, pages 2375– 2381, 2003.

[17] C. Lu and N. Ferrier. Repetitive motion analysis: Segmentation and event classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):258–263, 2004.

[18] Koichi Ogawara, Soshi Iba, Hiroshi Kimura, and Katsushi Ikeuchi. Recognition of human task by attention point analysis. In *IEEE International Conference on Intelligent Robot and Systems IROS'00*, pages 2121–2126, 2000.

[19] Koichi Ogawara, Soshi Iba, Hiroshi Kimura, and Katsushi Ikeuchi. Acquiring hand-action models by attention point analysis. In *IEEE International Conference on Robotics and Automation*, pages 465–470, 2001.

[20] Koichi Ogawara, Jun Takamatsu, Hiroshi Kimura, and Katsushi Ikeuchi. Modeling manipulation interactions by hidden Markov models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1096–1101, 2002.

[21] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[22] V.S. Ramachandran. Mirror neurons and imitation learning as the driving force behind the gerat leap forward in human evolution. *Edge*, 69, 2000.

[23] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[24] Hiroshi Shimodaira, Ken ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *Advances in Neural Information Processing Systems 14, NIPS2001*, pages 921–928, 2001.

[25] Dinoj Surendran and Gina-Anne Levow. Dialog act tagging with support vector machines and hidden Markov models. In *Interspeech 2006 — ICSLP*, Pittsburgh, PA, USA, 2006.

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

Learning and Recognition of Object Manipulation Actions Using Linear and Nonlinear Dimensionality Reduction

Isabel Serrano Vicente and Danica Kragic
Computational Vision and Active Perception Laboratory
Centre for Autonomous Systems
Royal Institute of Technology, Stockholm, Sweden
497600@celes.unizar.es, dani@kth.se

***Abstract*— Neuroscientific and physiological literature states that the core of developmental learning in humans is by watching another person performing a task. This has also motivated the research in the robotics area of learning by imitation and robot programming through demonstration. There is an extensive amount of work dealing with issues of *what*, *when* and *how* to imitate.**

In this work, we perform an extensive statistical evaluation for learning and recognition of object manipulation actions. We concentrate on single arm/hand actions but study the problem of modeling and dimensionality reduction for cases where actions are very similar to each other in terms of arm motions. For this purpose, we evaluate a linear and a nonlinear dimensionality reduction techniques: Principal Component Analysis and Spatio-Temporal Isomap. Classification of query sequences is based on different variants of Nearest Neighbor classification. We thoroughly describe and evaluate different parameters that affect the modeling strategies and perform the evaluation with a training set of 20 people.

I. Introduction

Human-computer interaction, surveillance, video retrieval are just a few example areas that require human activity recognition, [1]. In robotics, recognition of human activity has been used extensively for robot task learning through imitation and demonstration, [2], [3], [4], [5], [6], [7], [8], [9], [10]. For humans, one of the fundamentals of social behaviors is the understanding of each others intentions through perception and recognition of performed actions. However, the neural and functional mechanisms underlying this ability in human are still poorly understood, [11] which makes it difficult to develop the necessary models needed for designing a robot system that can learn just by observing a human or another robot performing an action. The recent discovery of *mirror neurons* in monkey's brain [12], [13] has nevertheless introduced new hypotheses and ideas about the process of imitation and its role in the evolution.

It has been shown in [14] that an action perceived by a human can be represented as a sequence of clearly segmented *action units*. This motivates the idea that the action recognition process may be considered as an interpretation of the continuous human behaviors which, in its turn, consists of a sequence of action primitives [8] such as *reaching*, *picking up*, *putting down*. In relation, learning *what* and *how* to imitate has been recognized as an important problem, [10]. It has been argued that the data used for imitation has statistical dependencies between the activities one wishes to model and that each activity has a rich set of features that can aid both the modeling and recognition process. While in the computer vision community, most work on modelling human motion has concentrated on cyclic motions such as walking or running, [1], examples in robotics consider mainly non-cyclic actions. In [5], [6], a framework for acquiring hand-action models by integrating multiple observations based on gesture spotting is proposed. [7] present a gesture imitation system where the focus is put on the coordinate system transformation (*View-Point Transformation*) so that the teacher induced gesture is transformed into the robot's egocentric system. This way the robot *observes* the gesture as it was generated by thye observer himself. [8] approaches the task learning problem by proposing a system for deriving behavior vocabularies or simple action models that can be used for more complex task extraction and learning. [10] presents a learning system for one and two-hand motions where the robot's body constraints are considered as a part of the optimal trajectory generation process. An interesting trend to note here is that most of the studies are based on a **single** user generated motion. A natural question to pose here is how the underlying modeling methods scale and apply for cases when the robot is supposed to learn from multiple teachers. The experimental evaluation conducted in our work is based on 20 people.

In robotics, many of the systems used for imitation are based on generative models such as Hidden Markov Models, [5], [10]. Generative models define a joint probability distribution over observations and state variables. For modeling of the observation process and enumerating all possible sequence of observations, it is commonly assumed that these are atomic and independent. This affects the inference problem which makes generative models intractable for multiple overlapping features of the observation or complex dependencies of observations at multiple time steps. One of the solutions to this problem may be the use of discriminative models such as Conditional Random Fields, [15].

In this work, we perform an extensive statistical evaluation for learning and recognition of object manipulation actions. Single arm/hand actions are considered with a specific focus on the problem of modeling and dimensionality reduction for cases where actions are very similar to each other in terms

of arm motions. For this purpose, we evaluate a linear and a nonlinear dimensionality reduction techniques: Principal Component Analysis and Spatio-Temporal Isomap. Classification of query sequences is based on a combination of clustering and different variants of Nearest Neighbor classifiers. For both methods, we thoroughly describe and evaluate different parameters that affect the modeling strategies and perform the evaluation with a training set of 20 people. To our knowledge, there are no examples in the field of robotics where such a large set of people was considered. Similar to [16], the results can be used to enable a more sophisticated probabilistic modeling and recognition of actions an provide a modeling basis for methods such as those presented in [8], [10].

Thus, the questions we wanted to answer with the current study were:

- What modeling strategies are suitable for action representation and recognition purposes?
- Is it possible to learn action when we do not have the knowledge of the task or the embodiment (kinematic structure) of the teacher?
- Is it possible to distinguish between very similar actions such as *pick up* and *push* an object?
- Is it enough to only observe the motion of the arm/hand or does the motion of the object have to be included in the modeling process?

This paper is organized as follows. In Section II we describe the experimental setting and collection of training data. In Section III we give a short overview of dimensionality reduction techniques and present details of their implementation in Section IV. Experimental evaluation is summarized in Section V and paper concluded in Section VI.

II. Data Collection and Preprocessing

We follow the classical approach to activity recognition through training and testing steps. Training step refers to a learning step where the data is collected, labelled and used to find an appropriate representation space for the data. The system learns a model for each activity which is then used for the classification of new actions in the testing step. The four activities considered in this work are:

1) Push forward an object placed on a table (P);
2) Rotate an object placed on table (R);
3) Pick up the object placed on the table (PU) and
4) Put down an object on a table (PD).

Notations P, R, PU, PD are used to denote different actions in the experimental evaluation in Section V.

Fig. 1 shows two example images stored during a push activity training - the activity is performed with the object being placed at two different heights. To motivate the choice of these activities, let us consider a robot being a part of a coffee drinking scenario. A *pick up* activity could be representing the fact of picking up the cup to take a swig of coffee; *put down* an object could represent leaving the cup of coffee after taking a swig, *rotate* an object would be similar to fold a napkin placed on the table, and finally, let

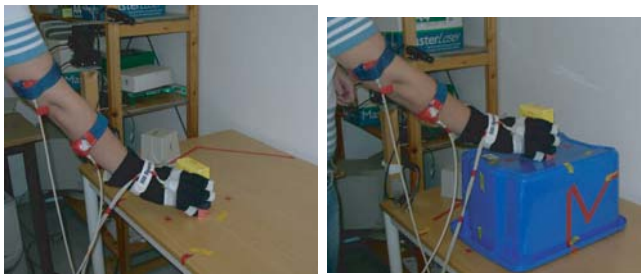


Fig. 1. Left) An example of pushing forward an object on the table and Right) An example of pushing forward an object on the box

us suppose that the person who sat down in front of you taking a coffee asks for the sugar bowl close to you and you *push* the bowl sliding over the table to bring it closer to him/her. The activities considered in this work are major buliding blocks of any similar task.

To generate the measurements for the training data, a Nest of Birds sensor was used, see Fig. 2 (right). The Nest of Birds simultaneously tracks the position and orientation of four sensors, referred to transmitter emitting pulsed DC magnetic field. The placement of the sensors is shown Fig. 1: thumb, hand, lower arm and upper arm. The persons involved in the study were not trained in any special way - each action started by having an arm in a relaxed, vertical position. Apart from the variation in their height and velocity with which an action was performed, the following variations were introduced to the training data:

- The objects were put on two different heights
- The person was standing at three different angles with respect to the table: 0, 30 and 60 degrees

Each action was performed three times for all combinations of the above heights and orientations resulting in total 18 training sequences per person and action thus 360 training sequences for each action.

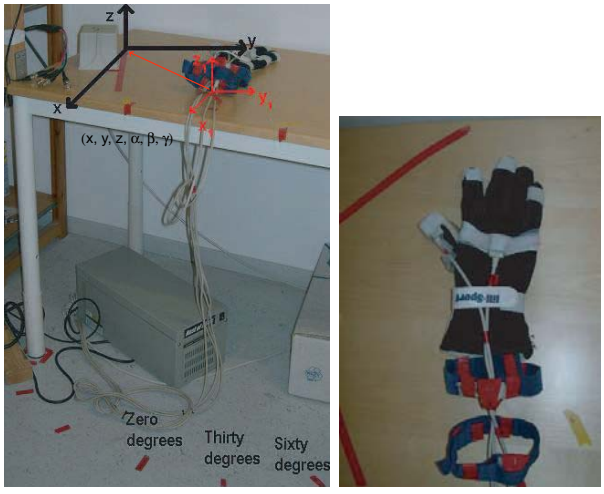


Fig. 2. Left) Nest of Birds sensor, and Right) Glove with the four sensors.

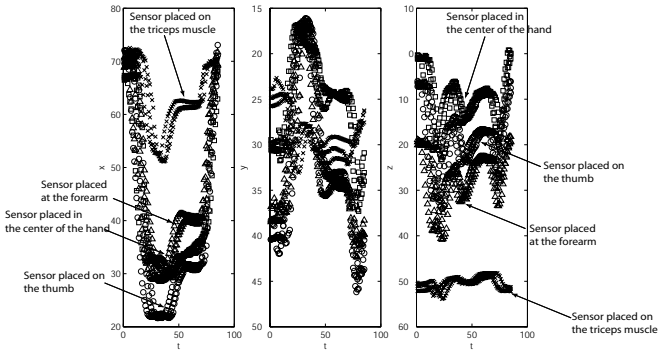


Fig. 3. Sensor measurements retrieved for three trials of a "rotate" activity.

III. Dimensionality Reduction

Finding low-dimensional data model hidden in the high-dimensional observations is one of the key problems in the area of activity modeling and recognition. In the current study, we have evaluated two dimensionality reduction methods. The first is the classical PCA which finds a low-dimensional embedding of the input data where the principal components are chosen such that they maximally explain the variance in the data. Since each data point is reconstructed by a suitable linear combination of the principal components, this method is applicable for cases where the assumption of linearity holds. However, for cases where the data represents essential nonlinear structures, PCA and similar techniques fail to detect the intrinsic dimensionality and model for the data. Therefore, we also evaluate a nonlinear dimensionality reduction approach proposed in [8] which is based on the isometric feature mapping or Isomap, [17].

A. Principal Component Analysis - PCA

PCA is commonly used for data dimensionality reduction, [18]. This method retains those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. The idea is that such low-order components often contain the "most important" aspects of the data and the high-order components often introduce more redundant information than new one. Therefore, the error introduced by ignoring the higher-order components is not significant if the assumption of linearity holds.

In relation to human motion modeling, the use of PCA for representation of temporal curves is common. It provides a statistical model of the variation present in the training set and can thus be used to construct a probabilistic prior for motion tracking based on Bayesian methods, [16].

B. Isometric Feature Mapping - Isomap

The main idea of Isomap, [17] is to find the intrinsic geometry of the data by computing the geodesic manifold distances between all pairs of data points. Once the geodesic distances are estimated, multidimensional scaling is applied which removes nonlinearities in the data and produces a coordinate space intrinsic to the underlying manifold.

Since the training data in our system are represented in a global coordinate system (robot centered), the system should be able to perform disambiguation of spatially proximal data that are structurally different (*pick up* and *put down*) as well as model the correspondence of spatially distal data points that share common structure (actions performed at different heights). An extension of the classical Isomap, the ST-Isomap, proposed in [8] is a method that satisfies the above requirements. Implementation details are presented in Section IV-C.

C. Clustering Methods

We have evaluated two clustering techniques in connection to PCA based action classification: *k*-means clustering and Gustafson-Kessel clustering. *k*-means clustering [18] is a partitioning method in which clusters are mutually exclusive (hard partitioning method). Clustering algorithms group sample points, $\mathbf{m}_j$ into c clusters. The set of cluster prototypes or centers is defined as $\mathbf{C} = \left[\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(c)}\right]$ where

$$\mathbf{c}^{(i)} = \frac{\sum_{j=1}^{d} u_{ij} \quad \mathbf{m}_j}{\sum_{j=1}^{d} u_{ij}} \qquad i = 1, 2, \ldots, c \tag{1}$$

where $u_{ij} \in \mathbf{U}$ denotes the membership of $\mathbf{m}_j$ in the ith cluster and $\mathbf{U}$ is known as the *partition matrix*.

For the classical *k*-means clustering, the hard partitioning space is defined as:

$$M_h = \{\mathbf{U} \in V_{cd} : u_{ij} \in \{0,1\}, \forall (i,j); \sum_{i=1}^{c} u_{ij} = 1; 0 < \sum_{i=1}^{d} u_{ij} < d, \forall i\} \tag{2}$$

The objective function we have to minimize is:

$$J_h(\mathbf{M};\mathbf{U},\mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{d} u_{ij} d_{\mathbf{A}}^2 \left(\mathbf{m}_j, \mathbf{c}^{(i)}\right) \tag{3}$$

where A is a norm-inducing matrix and $d_{\mathbf{A}}^2$ represents the distance measure

$$d_{\mathbf{A}}^2 = \left(\mathbf{m}_j, \mathbf{c}^{(i)}\right) = \|\mathbf{m}_j - \mathbf{c}^{(i)}\|_{\mathbf{A}}^2 = \left(\mathbf{m}_j - \mathbf{c}^{(i)}\right)^T \mathbf{A} \left(\mathbf{m}_j - \mathbf{c}^{(i)}\right) \tag{4}$$

The above condition of hard membership can be relaxed so that each sample point has some graded or "fuzzy" membership in a cluster. The incorporation of probabilities (or graded memberships) may improve the convergence of the clustering method compared to the classical *k*-means method. In addition, we do not have to assume anymore that the samples belong to spherical clusters.

We shortly describe the method used in our work also known as Gustafson-Kessel (GK) clustering. First, we define a fuzzy partition space as:

$$M_f = \{\mathbf{U} \in V_{cd} : u_{ij} \in [0,1], \forall (i,j); \sum_{i=1}^{c} u_{ij} = 1; 0 < \sum_{i=1}^{d} u_{ij} < d, \forall i\} \tag{5}$$

Here, fuzzy objective function is a least-squares functional:

$$J_f(\mathbf{M};\mathbf{U},\mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{d} (u_{ij})^w d_{\mathbf{A}}^2 \left(\mathbf{m}_j, \mathbf{c}^{(i)}\right) \tag{6}$$

where w is a weighting factor $w = [1, \infty)$. Gustafson-Kessel method is a variation of fuzzy clustering algorithms which allows the samples to belong to several clusters simultaneously, with different degrees of membership. It employs an adaptive distance norm in order to detect clusters of different geometrical shapes in the data set. Specifically, each cluster has its own norm-inducing matrix $\mathbf{A}^{(i)}$:

$$d^2_{\mathbf{A}^{(i)}} = \left(\mathbf{c}_l^{(i)} - \mathbf{m}_j\right)^T \mathbf{A}^{(i)} \left(\mathbf{c}_l^{(i)} - \mathbf{m}_j\right) \tag{7}$$

where

$$\mathbf{A}^{(i)} = (|\mathbf{F}^{(i)}|)^{1/(r+1)} (\mathbf{F}^{(i)})^{-1} \tag{8}$$

and

$$\mathbf{F}^{(i)} = \frac{\sum_{j=1}^{d} (u_{ij})^w \left(\mathbf{m}_j - \mathbf{c}^{(i)}\right) \left(\mathbf{m}_j - \mathbf{c}^{(i)}\right)^T}{\sum_{j=1}^{d} (u_{ij})^w} \tag{9}$$

IV. Implementation

We give a short overview and implementation details for the methods used in this study.

A. PCA without temporal dependencies

The basic idea investigated here was that each action consists of a set of discrete poses that are represented in some high-dimensional space since. These action are gathered by 1,2,3 or 4 sensors where each sensor provides a full pose estimate (3 translations and 3 rotations). Since the sensor used to capture the training data provides Euler angles in the reference coordinate system, we represent each angle by its sine and cosine value resulting in 9 measurements in total per sensor. This then defines the dimension of the covariance matrix, estimated in the PCA process, [18].

Our reasoning here was that different actions will vary differently along different directions. If we are able to find this directions, each action may be represented only with those ones along which the data varies the most, precisely what PCA gives us. The implementation follows the classical PCA approach: we first estimate the mean of the data, subtract it from all the samples, estimate the covariance matrix and estimate its SVD, [18]. Finally, we keep only the eigenvectors that for which eigenvalues $\lambda_n > 0.005\lambda_{max}$. In our evaluation, dependant of the number of sensors used to measure an action, the dimensionality reduction was following: single sensor (from 9 to 3), two sensors (18 to 5), three sensors (27 to 6) and four sensors (36 to 7). These values are easy to understand due to the constraints posed by the kinematic structure of the arm. Once the basic set of eigenvectors is chosen, the training data is projected to this reduced action representation space. This is done for each action separately. To ease the classification, we cluster each action representation space. For this purpose, we have used k-means and GK clustering presented in Section III-C.

In the classification stage, each testing sequence is first projected to the reduced action representation space. For each sample point in an action, the distance to the closest cluster center is estimated and the classification is based on the minimum Euclidean distance sum.

B. PCA with Temporal Dependencies

We have also evaluated a PCA approach where, similar to the studies performed on cyclic motions, [1], we took into account the temporal dependencies of the activities. To be able to estimate the covariance matrix using whole sequences, we normalized them to equal length - 85 sample points per sequence. According to the procedure described in the previous section, the dimensionality reduction was following: single sensor (from 765 to 17), two sensors (1530 to 22), three sensors (2225 to 24) and four sensors (3060 to 26). Training sequences are then projected to separate decreased spaces where each represents one of the actions. Classification of a new sequence is performed based on the minimum Euclidean distance sum.

C. ST-Isomap

For the implementation of Isomap, we adopted the approach proposed in [8]. As in the case of temporal PCA, the sequences are first normalized to equal length of 85 sample points. We shortly explain the basic idea behind the method.

- Calculate a distance matrix D^l between N local neighbors using Euclidean distances. In the current implementation, $N = 10$. For each data sample, identify common temporal neighbors (CTN) and adjacent temporal neighbors (ATN). We refer to [8] and [19] for a more detailed definition of these.
- Reduce the distances in the original matrix taking into account spatio-temporal correspondences

$$D^0_{S_i,S_j} = \begin{cases} D^l_{S_i,S_j}/(c_{CTN}c_{ATN}) & \text{if } S_j \in CTN(S_i) \text{ and } j = i+1, \\ D^l_{S_i,S_j}/c_{CTN} & \text{if } S_j \in CTN(S_i), \\ D^l_{S_i,S_j}/c_{ATN} & \text{if } j = i+1, \\ penalty(S_i,S_j) & \text{otherwise.} \end{cases} \tag{10}$$

  where c_{ATN} and c_{CTN} are scalar parameters and $CTN()$ denotes common temporal neighbors. In the current implementation, we set $c_{ATN} = 1$ and varied value for $c_{CTN} = [2 \quad 5 \quad 10 \quad 100]$. Fig. 4 shows the effect of c_{CTN} parameter to the resulting embedding of the activities.
- Use D_0 to compute shortest path distance matrix D_g using Dijkstra's algorithm, [20]
- Use Multidimensional Scaling [21] to embed D_g to a lower dimensional space. We have evaluated the system for $[3 \quad 4 \quad 5 \quad 6]$ dimensions.

V. Experimental evaluation

We present the results for i) PCA without temporal dependencies, ii) PCA with temporal dependencies and iii) ST-Isomap.

A. PCA without temporal dependencies

We have trained the system with 1, 5, 10 or 20 people. In case of a single person, we split the data in three possible combinations of two trials for training and one trial for evaluation. Similarly, this was done for the case of five and

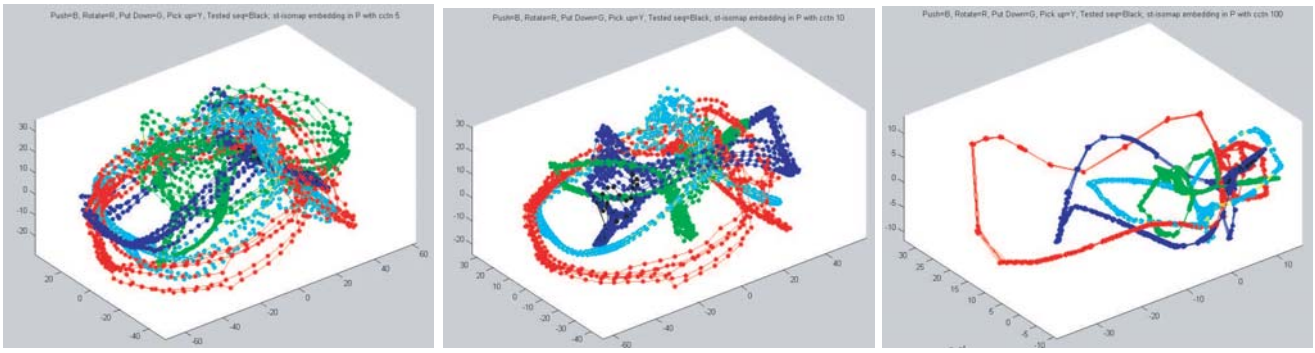


Fig. 4. Training data after estimating ST-Isomap and MDS embedding in 3 dimensions. The figures show the influence of the c_{CTN} parameter to the embedding: higher c_{CTN} brings sequences closer to each other.

ten people. For the case of twenty people, we split the trials in the three possible combinations of two for training the system and one for testing it, so we test the system three times with the demonstrations of all people. As we evaluated the system for each person three times and twenty people demonstrated the actions, it made a total of sixty tests.

Furthermore, in all cases we clustered the data using two both *k*-means and GK-clustering algorithm. We clustered the data to three, five and eight clusters in both cases. Here, we show only the resulting average of all the experiments and refer to [19] for a more detailed evaluation. In the forthcoming tables, the actions in the upper row are the tested sequences and the actions in the left column are the result of the classification. The results are expressed in percentage.

As explained in Section II, for each action, we have varied the position of the object (two heights) and the relative orientation of the person with respect to the table. The first experimental evaluation considered only two actions (push and rotate) where training and testing was performed on sequences captured under the same conditions (same orientation and height of the object). The average results considering different number of people in the training set as well as different numbers of sensors are summarized in Table I. We note here that we present the results for 5 clusters in more detail since it gave the highest classification rate on average. It can be seen that for only two actions, a classification rate of close to 90% is achieved. The presented results use are based on *k*-means clustering. GK- clustering gave approximately the same classification rate.

The second experiment to conduct was to consider all four actions, again considering the same conditions for training and testing. Due to the limited space, we show only the average classification rates for all four actions. In Table II we show how the size of the training set affects the rate given that the number of clusters is kept constant. In Table III we show how the number of clusters affect the classification rate given that the training set consist of all 20 people. Compared to the previous experiment, we can see that by adding two additional actions, the recognition rate is 30% lower on average. Again, similar results are obtained for both clustering methods.

Finally, we have evaluated the method considering all the variance in the data, namely that each action was performed on two different heights and in three orientations. The results are summarized in Table IV. It is obvious that, with the the recognition rates of about 40%, the simple approach considered here is not able to scale accordingly with the variation in the data. The next section presents the results of

5 clusters

	push	rot	push	rot	push	rot	push	rot
1pers	1s		2s		3s		4s	
push	**91.8**	1.6	**90.5**	1.3	**91.5**	2	**92.1**	2
rot	8.2	**98.4**	9.5	**98.7**	8.5	**98**	7.8	**98**
5pers	1s		2s		3s		4s	
push	**80**	34.4	**83.3**	27.8	**83.3**	19	**87.8**	30
rot	20	**65.6**	16.7	**72.2**	16.7	**81**	12.2	**70**
12pers	1s		2s		3s		4s	
push	**79.6**	18.5	**74.5**	14.8	**82.4**	18	**82.4**	14.8
rot	20.4	**81.5**	25.5	**85.2**	17.6	**82**	17.6	**85.2**
20pers	1s		2s		3s		4s	
push	**83**	14.7	**91.4**	16.7	**92.5**	11.9	**93.1**	10.8
rot	17	**85.3**	8.6	**83.3**	7.5	**88.1**	6.9	**89.2**

3 clusters

20pers	1s		2s		3s		4s	
push	**89.7**	28.9	**88.6**	21.7	**93.1**	26.4	**91.7**	21.1
rot	10.3	**71.1**	11.4	**78.3**	6.9	**73.6**	8.3	**78.9**

8 clusters

20pers	1s		2s		3s		4s	
push	**88.1**	15.6	**86.9**	12.5	**90.6**	10.6	**91.1**	8.9
rot	11.9	**84.5**	13.1	**87.5**	9.4	**89.4**	8.9	**91.1**

TABLE I

CLASSIFICATION RATES FOR TWO ACTIONS (PUSH, ROTATE) WHEN THE TRAINING AND TESTING WAS DONE UNDER SAME CONDITIONS (OBJECT HEIGHT, PERSONS ORIENTATION) USING *k*-MEANS CLUSTERING.

1 pers	1s	2s	3s	4s
average	**91.4**	**91.1**	**90.2**	**90**
5 pers	1s	2s	3s	4s
average	**61.9**	**65**	**68.6**	**61.1**
12 pers	1s	2s	3s	4s
average	**60.8**	**60.8**	**63.1**	**61.7**

TABLE II

CLASSIFICATION RATES FOR FOUR ACTIONS TRAINED AND TESTED IN SAME CONDITIONS (HEIGHT AND ORIENTATION), WITH VARYING SIZE OF THE TRAINING SET.THE NUMBER OF CLUSTERS IN *k*-MEANS IS 5.

3 clusters	1s	2s	3s	4s
average	**59.4**	**61.4**	**62.2**	**64.1**
5 clusters	1s	2s	3s	4s
average	**64.7**	**68.4**	**70.6**	**69.8**
8 clusters	1s	2s	3s	4s
average	**66.5**	**68**	**68.9**	**70**

TABLE III

CLASSIFICATION RATES FOR FOUR ACTIONS AND 20 PEOPLE TRAINED AND TESTED IN THE SAME CONDITIONS /HEIGHT AND ORIENTATION), WITH VARYING NUMBER OF CLUSTERS.

1 pers	1s	2s	3s	4s
average	**37.5**	**30.6**	**37.5**	**37.5**
5 pers	1s	2s	3s	4s
average	**34.7**	**33.9**	**38.1**	**38.9**
12 pers	1s	2s	3s	4s
average	**34.3**	**33.7**	**37.5**	**35.6**
20 pers	1s	2s	3s	4s
average	**35.4**	**37.2**	**37.3**	**37.4**

TABLE IV

CLASSIFICATION RATES FOR FOUR ACTIONS TRAINED AND TESTED IN DIFFERENT CONDITIONS, WITH VARYING SIZE OF THE TRAINING SET. THE NUMBER OF CLUSTERS USED IN k-MEANS IS FIXED TO FIVE.

the method where temporal dependencies between the data points are taken into account.

1 pers	1s	2s	3s	4s
average	**41.7**	**36.1**	**38.9**	**27.8**
5 pers	1s	2s	3s	4s
average	**35.8**	**35.8**	**33.6**	**36.9**
12 pers	1s	2s	3s	4s
average	**35.2**	**38.1**	**40**	**40.1**
20 pers	1s	2s	3s	4s
average	**41**	**34.3**	**36.7**	**36.3**

TABLE V

CLASSIFICATION RATES FOR FOUR ACTIONS TRAINED AND TESTED IN DIFFERENT CONDITIONS, WITH VARYING SIZE OF THE TRAINING SET. THE NUMBER OF CLUSTERS USED IN GK CLUSTERING IS FIXED TO FIVE.

B. Temporal PCA

We present here only the results with all four actions, where the training and testing was performed given all 20 people and actions performed in all combinations of orientations and heights. As above, as each action sequence was performed three times in all conditions, we evaluated the system taken all combinations of two testing and one training action sets.

Table VI summarizes the results for one (1s, hand), two (2s, thumb and hand), three (3s, thumb, hand, forearm) and all four (4s) sensors considered. The important thing to note is that the recognition rate is somewhat higher compared to the results in the previous section but the system still has the difficulty of discriminating between some of the actions. We believe that this is an important result. Implementing PCA with temporal dependencies requires aligned and equal length sequences which may be difficult to obtain in an

	1s				2s			
	P	R	PD	PU	P	R	PD	PU
P	**50.1**	42.5	12.5	29.2	**50**	43.3	12.5	32.5
R	8.3	**33.3**	3.3	10	9.2	**35**	3.3	15
PD	15	3.3	**69.2**	22.5	15	5.8	**69.2**	20
PU	25.8	20.8	15	**38.3**	25.8	15.8	15	**32.5**
	3s				4s			
	P	R	PD	PU	P	R	PD	PU
P	**51.7**	42.5	12.5	30	**51.7**	42.5	12.5	29.2
R	7.5	**35**	3.3	1.5	8.3	**30**	3.3	11.7
PD	14.2	5	**69.2**	21.7	14.2	4.2	**66.7**	25
PU	26.7	17.5	15	**35.8**	25.8	23.3	17.5	**34.2**

TABLE VI

CLASSIFICATION RATES FOR PCA WITH TEMPORAL DEPENDENCIES FOR FOUR ACTIONS AND 20 PEOPLE IN THE TRAINING SET.

online process where we would like to perform recognition during and not after an action has been executed. A simple "voting" approach presented in the previous section may be as suitable. Another issue that we have investigated was if the number of sensors affects the classification rate. The results are summarized in Fig. 5. We note that the difference is only marginal and that almost equal results are obtained with a single or all four sensors. This means that for actions which are very similar in arm motion placing only a single sensor on the hand or tracking only the position and orientation of the hand may be sufficient.

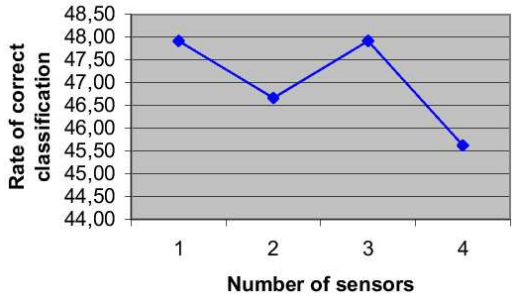
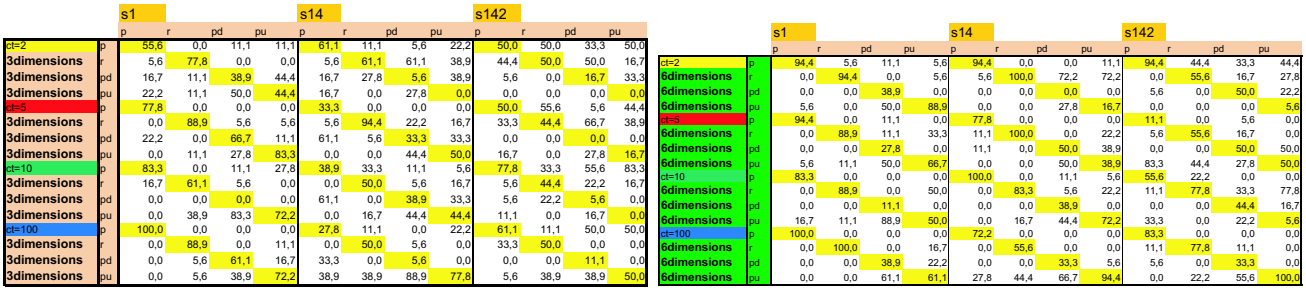


Fig. 5. The effect of number of sensors used to the classification rate.

C. ISOMAP

A non-linear dimension reduction, ST-Isomap was applied to extract a low dimensional representation for the activities. Shepard interpolation [22] was used map a query sequence to the estimated embedding. For classification, minimum Euclidean distance sum between the query samples and samples representing each activity in the embedding was used. From the training set of 20 people, we formed subsets of one, two and three persons. For each person, all four activities were considered using three trials for all combinations of three orientations and two heights. The classification was performed with the queries not included in the training set.

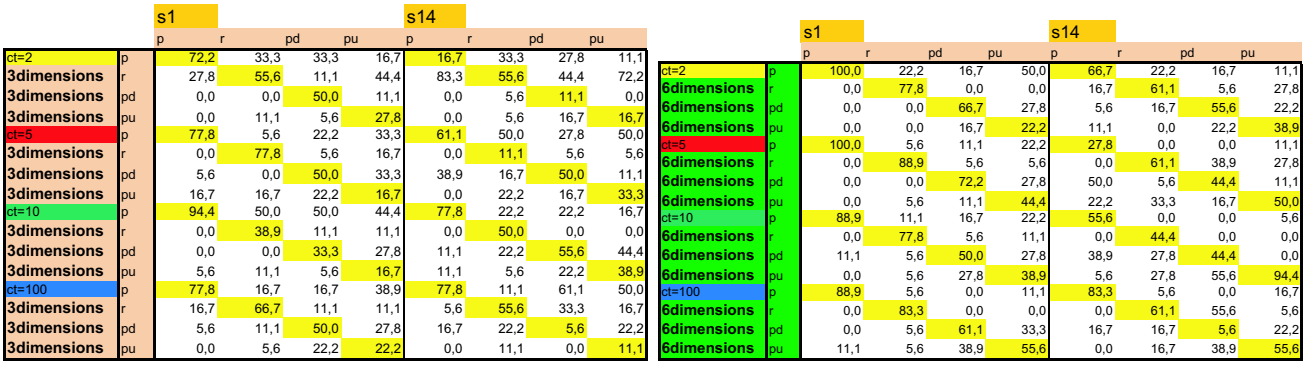
We have evaluated to the system with different numbers and sensors placements. In the forthcoming tables, this is denoted as: sensors placed on the i) hand (s1), ii) hand and thumb (s14), iii) hand, thumb and forearm (s142).



		s1				s14				s142			
		p	r	pd	pu	p	r	pd	pu	p	r	pd	pu
ct=2	p	55,6	0,0	11,1	11,1	61,1	11,1	5,6	22,2	50,0	50,0	33,3	50,0
3dimensions	r	5,6	77,8	0,0	0,0	5,6	61,1	61,1	38,9	44,4	50,0	50,0	16,7
3dimensions	pd	16,7	11,1	38,9	44,4	16,7	27,8	5,6	38,9	5,6	0,0	16,7	33,3
3dimensions	pu	22,2	11,1	50,0	44,4	16,7	0,0	27,8	0,0	0,0	0,0	0,0	0,0
ct=5	p	77,8	0,0	0,0	0,0	33,3	0,0	0,0	0,0	50,0	55,6	5,6	44,4
3dimensions	r	0,0	88,9	5,6	5,6	5,6	94,4	22,2	16,7	33,3	44,4	66,7	38,9
3dimensions	pd	22,2	0,0	66,7	11,1	61,1	5,6	33,3	33,3	0,0	0,0	0,0	0,0
3dimensions	pu	0,0	11,1	27,8	83,3	0,0	0,0	44,4	50,0	16,7	0,0	27,8	16,7
ct=10	p	83,3	0,0	11,1	27,8	38,9	33,3	11,1	5,6	77,8	33,3	55,6	83,3
3dimensions	r	16,7	61,1	5,6	0,0	0,0	50,0	5,6	16,7	5,6	44,4	22,2	16,7
3dimensions	pd	0,0	0,0	0,0	0,0	61,1	0,0	38,9	33,3	5,6	22,2	5,6	0,0
3dimensions	pu	0,0	38,9	83,3	72,2	0,0	16,7	44,4	44,4	11,1	0,0	16,7	0,0
ct=100	p	100,0	0,0	0,0	0,0	27,8	11,1	0,0	22,2	61,1	11,1	50,0	50,0
3dimensions	r	0,0	88,9	0,0	11,1	0,0	50,0	5,6	0,0	33,3	50,0	0,0	0,0
3dimensions	pd	0,0	5,6	61,1	16,7	33,3	0,0	5,6	0,0	0,0	0,0	11,1	0,0
3dimensions	pu	0,0	5,6	38,9	72,2	38,9	38,9	88,9	77,8	5,6	38,9	38,9	50,0

		s1				s14				s142			
		p	r	pd	pu	p	r	pd	pu	p	r	pd	pu
ct=2	p	94,4	5,6	11,1	5,6	94,4	0,0	0,0	11,1	94,4	44,4	33,3	44,4
6dimensions	r	0,0	94,4	0,0	5,6	5,6	100,0	72,2	72,2	0,0	55,6	16,7	27,8
6dimensions	pd	0,0	0,0	38,9	0,0	0,0	0,0	0,0	0,0	5,6	0,0	50,0	22,2
6dimensions	pu	5,6	0,0	50,0	88,9	0,0	0,0	27,8	16,7	0,0	0,0	0,0	5,6
ct=5	p	94,4	0,0	11,1	0,0	77,8	0,0	0,0	0,0	11,1	0,0	5,6	0,0
6dimensions	r	0,0	88,9	11,1	33,3	11,1	100,0	0,0	22,2	5,6	55,6	16,7	0,0
6dimensions	pd	0,0	0,0	27,8	0,0	11,1	0,0	50,0	38,9	0,0	0,0	50,0	50,0
6dimensions	pu	5,6	11,1	50,0	66,7	0,0	0,0	50,0	38,9	83,3	44,4	27,8	50,0
ct=10	p	83,3	0,0	0,0	0,0	100,0	0,0	11,1	5,6	55,6	22,2	0,0	0,0
6dimensions	r	0,0	88,9	0,0	50,0	0,0	83,3	5,6	22,2	11,1	77,8	33,3	77,8
6dimensions	pd	0,0	0,0	11,1	0,0	0,0	0,0	38,9	0,0	0,0	0,0	44,4	16,7
6dimensions	pu	16,7	11,1	88,9	50,0	0,0	16,7	44,4	72,2	33,3	0,0	22,2	5,6
ct=100	p	100,0	0,0	0,0	0,0	72,2	0,0	0,0	0,0	83,3	0,0	0,0	0,0
6dimensions	r	0,0	100,0	0,0	16,7	0,0	55,6	0,0	0,0	11,1	77,8	11,1	0,0
6dimensions	pd	0,0	0,0	38,9	22,2	0,0	0,0	33,3	5,6	5,6	0,0	33,3	0,0
6dimensions	pu	0,0	0,0	61,1	61,1	27,8	44,4	66,7	94,4	0,0	22,2	55,6	100,0

Fig. 6. ST-Isomap results with training based on one person and testing it with another one. The results show how the dimension of the embedding and sensor number affect the classification result.



		s1				s14			
		p	r	pd	pu	p	r	pd	pu
ct=2	p	72,2	33,3	33,3	16,7	16,7	33,3	27,8	11,1
3dimensions	r	27,8	55,6	11,1	44,4	83,3	55,6	44,4	72,2
3dimensions	pd	0,0	0,0	50,0	11,1	0,0	5,6	11,1	0,0
3dimensions	pu	0,0	11,1	5,6	27,8	0,0	5,6	16,7	16,7
ct=5	p	77,8	5,6	22,2	33,3	61,1	50,0	27,8	50,0
3dimensions	r	0,0	77,8	5,6	16,7	0,0	11,1	5,6	5,6
3dimensions	pd	5,6	0,0	50,0	33,3	38,9	16,7	50,0	11,1
3dimensions	pu	16,7	16,7	22,2	16,7	0,0	22,2	16,7	33,3
ct=10	p	94,4	50,0	50,0	44,4	77,8	22,2	22,2	16,7
3dimensions	r	0,0	38,9	11,1	11,1	0,0	50,0	0,0	0,0
3dimensions	pd	0,0	0,0	33,3	27,8	11,1	22,2	55,6	44,4
3dimensions	pu	5,6	11,1	5,6	16,7	11,1	5,6	22,2	38,9
ct=100	p	77,8	16,7	16,7	38,9	77,8	11,1	61,1	50,0
3dimensions	r	16,7	66,7	11,1	11,1	5,6	55,6	33,3	16,7
3dimensions	pd	5,6	11,1	50,0	27,8	16,7	22,2	5,6	22,2
3dimensions	pu	0,0	5,6	22,2	22,2	0,0	11,1	0,0	11,1

		s1				s14			
		p	r	pd	pu	p	r	pd	pu
ct=2	p	100,0	22,2	16,7	50,0	66,7	22,2	16,7	11,1
6dimensions	r	0,0	77,8	0,0	0,0	16,7	61,1	5,6	27,8
6dimensions	pd	0,0	0,0	66,7	27,8	5,6	16,7	55,6	22,2
6dimensions	pu	0,0	0,0	16,7	22,2	11,1	0,0	22,2	38,9
ct=5	p	100,0	5,6	11,1	22,2	27,8	0,0	0,0	11,1
6dimensions	r	0,0	88,9	5,6	5,6	0,0	61,1	38,9	27,8
6dimensions	pd	0,0	0,0	72,2	27,8	50,0	5,6	44,4	11,1
6dimensions	pu	0,0	5,6	11,1	44,4	22,2	33,3	16,7	50,0
ct=10	p	88,9	11,1	16,7	22,2	55,6	0,0	0,0	5,6
6dimensions	r	0,0	77,8	5,6	11,1	0,0	44,4	0,0	0,0
6dimensions	pd	11,1	5,6	50,0	27,8	38,9	27,8	44,4	0,0
6dimensions	pu	0,0	5,6	27,8	38,9	5,6	27,8	55,6	94,4
ct=100	p	88,9	5,6	0,0	11,1	83,3	5,6	0,0	16,7
6dimensions	r	0,0	83,3	0,0	0,0	0,0	61,1	55,6	5,6
6dimensions	pd	0,0	5,6	61,1	33,3	16,7	16,7	5,6	22,2
6dimensions	pu	11,1	5,6	38,9	55,6	0,0	16,7	38,9	55,6

Fig. 7. ST-Isomap results with training based on 3 persons and testing it with another one. The results show how the dimension of the embedding and sensor number affect the classification result.

Thorough experimental evaluation with different values for c_{CTN} parameter and dimensionality of the embedding space was conducted.

Fig.6 shows the results obtained by ST-Isomap with training based on a single person. The results show how the dimension of the embedding and sensor number affect the classification result. Here, parameter $c_{CTN} = 2$. Fig.7 shows a similar experiment, but here the size of the training set was three. It is interesting to notice that best results are obtained based on the sensor placed on the hand. For the future, this would motivate that only the position of the user's hand and not the complete arm joint motion is needed to recognize object manipulation sequences when ST-Isomap is used. The effect of changing the values of parameter c_{CTN} is shown in Table V-C. On average, the best results are obtained with $c_{CTN} = 5$ and the average values per action are shown in Fig. 8. From the above results, it can be seen that, compared to the PCA, ST-Isomap gives better classification results.

VI. Conclusion

In this work, we have performed an initial study on recognition of four object manipulation actions: pick up, put down, rotate and push. Training and testing was performed with 20 people where the manipulated object was placed on two different heights and people performing the actions multiple times at three different orientations. We believe that this study is important and shows how the variation in the training data affects the recognition rate. Most of the current systems that utilize robot imitation learning use a single person to train or teach tasks to the robot. Since the intention for the future is that robots will be able to learn from observing *different* and *multiple* people that perform same actions, we believe that it is important to study how different methods scale with respect to this.

In this work, we have concentrated on evaluation of dimensionality reduction using linear and nonlinear techniques. We have shown how the number of sensors and different parameters affect the classification rate. We are aware of the

		push	rot	pd	pu
ct = 2	push	**88.9**	22.2	29.2	36.1
ct = 2	rot	11.1	**70.9**	8.3	16.7
ct = 2	pd	0	0	**51.4**	16.7
ct = 2	pu	0	6.9	11.1	**30.5**
ct = 5	push	**88.9**	6.9	19.4	25
ct = 5	rot	0	**79.2**	4.2	9.7
ct = 5	pd	1.3	0	**62.5**	27.8
ct = 5	pu	9.7	13.9	13.9	**37.5**
ct = 10	push	**90.3**	18.1	25	29.2
ct = 10	rot	1.4	**72.2**	8.3	13.9
ct = 10	pd	2.8	2.8	**50**	30.5
ct = 10	pu	5.5	6.9	16.7	**26.4**
ct = 100	push	**84.7**	8.3	6.9	23.6
ct = 100	rot	5.6	**80.6**	5.6	4.2
ct = 100	pd	2.8	6.9	**65.3**	36.1
ct = 100	pu	6.9	4.2	22.2	**36.1**

TABLE VII

Classification results using a single sensor placed on the hand. Training was performed with 3 persons. The recognition rates show the dependency on the parameter c_{CTN}.

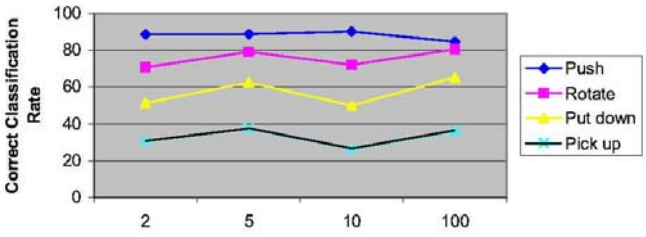


Fig. 8. Analysis of the recognition results when changing $c_{CTN} = 2, 5, 10, 100$ with a single sensor placed on the hand and training with three persons.

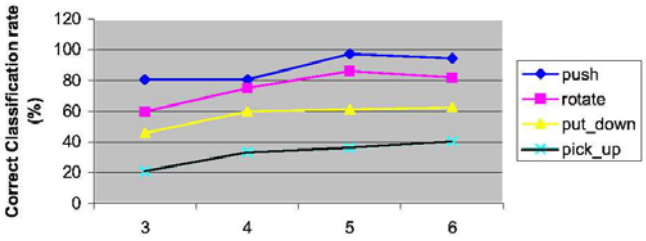


Fig. 9. Analysis of the recognition results when changing the dimension of the embedding space (3,4,5,6) with a single sensor placed on the hand and training with three persons.

fact that PCA and nearest neighbor classification are very simple techniques but we hope that our future work and work of other we evaluate more advanced techniques on the same data (which will be soon available for public access) and compare it to the results obtained in this work. We also believe that this data and evaluation follows the current trend of designing different benchmarking criteria in robotics.

Regarding the four questions posed in Section I we believe that for recognition of actions that are very similar, dimensionality reduction has to be performed with significant care in order to preserve the true variance in the data. We also believe that using the explicit knowledge of kinematic chains (arm model) may not be necessary in order to achieve satisfactory recognition rates. Finally, for some actions it is enough to provide only the measurements of the hand motions while distinguishing between *pick-up* and *put-down* would gain from including the motion of the object as well.

ACKNOWLEDGMENT

This work has been supported by EU through the project PACO-PLUS, FP6-2004-IST-4-27657. We also thank Odest Chadwicke Jenkins for the valuable input on the implementation of ST-Isomap.

REFERENCES

[1] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding: CVIU*, vol. 73, no. 3, pp. 428–440, 1999.

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

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

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

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

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

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

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

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

[10] 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.

[11] M. Iacoboni, I. Molnar-Szakacs, V. Galles, G. Buccino, J. Mazziotta, and G. Rizzolatti, "Grasping the intentions of others with one's own mirror neuron system," *PLOS Biology*, vol. 3, no. 3, 2005.

[12] V. Ramachandran, "Mirror neurons and imitation learning as the driving force behind the gerat leap forward in human evolution," *Edge*, vol. 69, 2000.

[13] L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti, "Visuomotor neurons: Ambiguity of the discharge or 'motor perception'?, volume = 35, year = 2000," *International Journal of Psychophysiology*, no. 2-3, pp. 165–177.

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

[15] C. Sminchiescu, A. Kanaujia, Z. Li, and D. Metaxas, "Conditional models for contextual human motion recognition," in *International Conference on Computer Vision, ICCV'05*, pp. 1808–1815, 2005.

[16]

[17] J. Tenenbaum, V. de Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[18] R. Duda, P. Hart, and D. Stork, *Pattern classification*. New York: Wiley-Interscience, 2001.

[19] I. S. Vicente, *Human action recognition based on linear and nonlinear dimensionality reduction using PCA and Isomap*. KTH, Stockholm, Sweden: Master thesis, 2006, http://cogvis.nada.kth.se/ danik/Isabel.pdf.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Second Edition. MIT Press and McGraw-Hill, 2001.

[21] M. Cox and M. Cox, *Multidimensional Scaling*. Chapman and Hall, 2001.

[22] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proc. 23rd National Conference ACM*, pp. 517–524, 1968.