Project no.: **027657**

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

Project Acronym: **PACO-PLUS**

Deliverable no.: D2.2

Title of the deliverable: Software components necessary for robot sensing and control when learning object-action complexes

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, Kai Welke, Pedram Azad, Andrej Kos
Participants(s):	JSI, UniKarl
Work package contributing to the deliverable:	WP1, WP2, WP8
Nature:	R/D
Version:	1.0
Total number of pages:	6
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 report summarizes the software providing the basis for learning of object-action complexes. The software was tested on the PACO-PLUS hardware (humanoid at UniKarl, PA-10 arm at JSI and the computer architecture at both institutions). It consists of oculomotor control and the associated tracking and visual attention algorithms, human-like inverse kinematics, human motion mapping and reproduction, and of the control and sensing algorithms that can be used to manipulate unknown objects with a robot arm in order to acquire representations suitable for recognition and higher cognitive tasks. The software is included on the supplied CD. The CD also contains several videos demonstrating the functionality of the developed software.

Keyword list: Sensorimotor primitives, oculomotor control, visual attention, object learning

Table of Contents

Introduction

We have equipped the PACO-PLUS hardware with the following sensorimotor primitives that can be used to support object exploration and interaction with people:

- Oculomotor behaviours on a humanoid head (saccades and smooth pursuit + foveation).
- Distributed visual attention, integrated with robot head movements.
- Inverse kinematics to realize manipulation and explorative movements with redundant humanoid robot arms.
- Motion mapping and reproduction.
- Sensorimotor primitives for building object representations.

The software is available on the CD attached to this deliverable. Since running the code requires the availability of PACO-PLUS hardware, we supplemented the software with some videos showing the functionality of the provided code. The videos are located in the directories containing the code demonstrated by the video.

Oculomotor Control and Tracking

We developed a control system that is appropriate to smoothly pursue an object and enhances the appearance of the humanoid through mimicking aspects of human movement: human eyes follow object movement, but without head movement have a limited range; thus, the robot's control system supports its eye movements through head movements. Our humanoid head has seven degrees of freedom that are used to maintain the view of the object. The head is equipped with two eyes. The eyes have a common tilt and can pan independently. The visual system is mounted on a 4 DOF neck which is realized as a Pitch-Roll-Yaw-Pitch mechanism. 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 of each eye and the common tilt to keep the target near the centre of the corresponding view. 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 Therefore, tracking is interrupted as soon as the object of interest is not visible in both camera images. To continue the tracking procedure either the object of interest has to be moved into the robot's field of view or a higher-level module responsible for visual attention, which is explained in the following section, has to attract the robot's attention to the object again.

To aid in coordinating the joints, we assign a relaxation position to each joint and the target position. The relaxation position for the target is in the centre of the view, and the eyes' task is to bring the target to that position. The relaxation position for the three eye joints is to face forward, and the head's task is to bring the eyes to that position. Further, the four neck joints have a relaxation 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.

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 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.

Foveation is implemented by introducing a constant offset from the centre for the target position in the peripheral views. We have shown that this is sufficient to keep the object in the fovea as long as the object does not approach the eyes too closely.

The control algorithms are implemented in standard C and are included on the CD in the directory OculomotorControl.

Visual Attention

Before the robot can learn about new objects, it must get some hints about where to look for them. The theory of preattentive, bottom-up visual processing provides us with one possible way of how to initially direct the robot's focus of attention. Our implementation is based on the model proposed by Itti, Koch, and Niebur and feature integration theory, which postulates that bottom-up preattentive processing explores the images using a number of feature processors, whose output is integrated into a global saliency map. The focus of attention emerges through competition across features in the integrated saliency map. Since this a computationally intensive process, we decided to utilize distributed processing to implement it.

The system starts by acquiring a video stream, which is sent across a network of computers for further processing. Each computer in this first processing line is responsible for one visual cue and computes a retinal feature map in a pyramidal scheme. This results in a number of feature maps given at various scales. The following cues are incorporated into the system at the moment: intensity, colour, motion, and edges. The edge feature processor is implemented in a biologically motivated way by combining the results of sixteen Gabor kernels given at different orientations and scales. Within each feature processor, maps at different scales are combined to generate a global conspicuity map that emphasizes locations that stand out from their surroundings. The conspicuity maps are sent to the next node in the cluster for integration. This node takes care of integrating the incoming visual streams arriving with different latencies and with different frame rates. Our synchronization approach combines feature maps with different time stamps so that the time difference between the simultaneously processed maps is minimal. Properly synchronized conspicuity maps are combined into a global saliency map, which encodes the saliency of each pixel in the image based on the results of all feature processors. The global saliency map is sent to the next node in the cluster that is responsible for time integration of the incoming saliency maps. We implemented a winner-take-all network to compute the most salient area in the image. Finally, the humanoid's eyes perform a saccade to focus on the identified area.

The described system was implemented in C++ and is included in the VisualAttention directory of the attached CD.

Human-Like Inverse Kinematics for Pointing and Reaching

For the execution of manipulation and exploration tasks we have developed and implemented several methods for solving the inverse kinematics of redundant humanoid robot arms. This is

necessary because most of manipulation tasks are specified in terms of the object trajectories. Kinematics redundancy is used to avoid mechanical joint limits, to minimize the reconfiguration of the arm, and to generate human-like manipulation motions.

The use of redundancy for the generation of human-like robot arm motion has been already proposed in the literature and a variety of hypothetical cost functions has been suggested to explain principles of the human arm movements. In our work, we consider the problem of generating human-like motion from the kinematics' point of view. We have developed and implemented algorithms that incorporate physiological observation of human arm movements into a closed-form solution of the inverse kinematics problem to generate natural looking arm postures. The generation of human-like manipulation motion has been implemented and tested successfully for the 7-DOF arm of the humanoid robot ARMAR-III. The result is a real-time control scheme that eliminates a major part of unnatural arm configurations during the execution of pointing and reaching tasks.

The control algorithms described above are implemented in C++ in the software framework MCA2 (www.mca2.org), which we use to control the robot and are included in the ArmControl directory of the attached CD.

Motion Mapping and Reproduction

For the reproduction of captured movements on the robot we have developed two modules. One module is the so-called joint mapper, which is responsible for the conversion of the marker data that was captured with the VICON system to the M3 format (see Deliverable 8.2.1). The conversion of the trajectories acquired with the vision-based markerless human motion capture system does not need a separate module, since it can directly produce the M3 format due to the similar kinematics. The second module converts the data from the M3 format to the kinematics of ARMAR III.

The joint mapper module uses inverse kinematics formulas to derive joint angle values from the positions of neighbouring markers. This processing is performed offline; the module receives the files with the marker data as input and produces files containing joint angle trajectories in the M3 format.

The conversion module for the reproduction of movements on ARMAR III is implemented in a flexible way. It can be configured by an XML file, which describes the process of conversion. Therefore, it is possible to produce output for basically any target demonstrator, once the configuration file is defined properly. By now, we have specified the conversion to the ARMAR III platform only. The conversion is then performed online, producing trajectories in terms of the robot's kinematics on the fly. The input can be switched: it can be either a file containing data specified in the M3 format, or it can receive live data from a real-time human motion capture system.

This conversion is built in into the MCA2 software framework. This allows us to switch from simulation to reproduction on the real robot without any additional effort. The entire source code is included in the directory MotionMapping of the attached CD.

Learning object representations by manipulation

We have implemented the following sensorimotor primitives that can support learning object representations:

- A movement that allows the robot to determine the relationship between the image and world coordinate system at the given configuration of the eyes. The accompanying hand tracking process provides the data that is needed for calculations.
- An explorative sensorimotor 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 centre and have appropriate size for learning, i. e. it will cover significant portion of the image while being away from the image boundary. A Bayesian process that allows the robot to discern the object from the background based on knowledge about the robot movement, background models and other cues is integrated with the motor control part of the primitive.
- 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 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. This motor primitive is integrated with the same Bayesian technique that allows the robot to discern the object from the background. After the robot observed the objects from all relevant viewing directions, a nonlinear multi-class support vector machine is learned to enable for 3-D object recognition.

The techniques used to implement these processes are described in D2.1 and the attached papers. The motor control part was implemented using Matlab, while the sensor data processing part was implemented in C++ to ensure real-time operation. The software is available in the directory "ObjectLearning" on the supplied CD.