2008 8th IEEE-RAS International Conference on Humanoid Robots
December 1 ~ 3, 2008 / Daejeon, Korea

Learning Primitive Actions through Object Exploration

Damir Omrčen, Aleš Ude, Andrej Kos

Jozef Stefan Institute, Slovenia, damir.omrcen@ijs.si, ales.ude@ijs.si, andrej.kos@ijs.si

***Abstract*— The goal of this paper is to investigate how to acquire useful action knowledge by observing the results of exploratory actions on objects. We focus on poking as a representative type of nonprehensile manipulation. Poking can be defined as a short term pushing action. Here we propose an explorative process that allows the robot to learn the relationship between the point of contact on the object boundary and the angle of poke and the actual response of an object. The robot acquires this knowledge without having any prior knowledge about the action. Initially, the robot was only able to move in random directions. Such self emergent processes are essential for the early cognition.**
The proposed process has been implemented and tested on the humanoid robot Hoap-3.

I. Introduction

The main motivation of this work is that many decades of research in the fields of robotics and artificial intelligence (AI) did not result in an intelligent "android like" robot. Why classical AI and robotics did not succeed in building an intelligent robot that can think like a human?

Traditional AI did not succeed due to the lack of a solid theoretical foundation as discussed in a very pointed way by Dennett [1], when he had introduced the "frame-problem". Additional weakness of the AI in the real world scenarios is the uncertainty in the world information, due to uncertain sensor information. However, in our opinion the most important drawback is that there is no self-emergence in classical AI. The instructor/user has to put more knowledge into the system than he gets it out of it. Nothing emerges by itself. In [2] Lungarella et al. present a survey on developmental robotic, which tries to solve the problems mentioned above.

In this work we investigate how to improve the self-emergence process when learning continuous object-action effects. Our research is part of an EU project PACO-PLUS, whose objective is to develop new methods to endow an artificial robotic system with the ability to give meaning to objects through perception, manipulation, and interaction with people. One of our guiding principles is that new object-action knowledge on a humanoid robot can emerge by exploring the external world. More specifically, by performing actions on different object, the robot can learn the results and preconditions of the actions.

We build cognition on a paradigm of Object-Action Complexes (OAC). Objects and Actions are inseparably intertwined and that categories are therefore determined (and also limited) by the action an agent can perform and by the attributes of the world it can perceive; the resulting, so-called Object-Action Complexes (OACs) are the entities on which cognition develops (action-centred cognition). Entities ("things") in the world of a robot (or human) will only become semantically useful "objects" through the action that the agent can/will perform on them. Objects are not just "things" upon which active agents act, but may be able to execute their own actions. Thus each active agent is just another instance of an OAC. This paradigm of OACs offers two novel key issues which will assure that a system with advanced cognitive properties can be developed.

Objects and actions cannot be separated, because objects can induce actions (cup → drink), while actions can redefine objects. While this paper is concerned with OACs at the level of early perception-action events, the project strives to provide a continuous path from such events to complex cognitive processes, where OACs are used as basic building blocks.

To acquire new primitive actions, the robot starts by randomly acting on various objects in its environment. The goal of this explorative process is to acquire new information that was not built into the system. As an example we study how to learn a relatively simple pushing behaviour. We also show how this knowledge can later be used to move (or to control) an object in a desired direction.

Pushing, poking, and rolling are examples of nonprehensile manipulation of objects, i.e. object manipulation without a grasp. This kind of manipulation is used when it is difficult to grasp an object, when an object is too large or too heavy, etc. In this paper we focus on poking as a representative type of nonprehensile manipulation. Poking can be defined as a short term pushing action. Conceptually, our goal is to investigate how to acquire useful action knowledge by observing the results of exploratory actions on objects. For this purpose we study how poking behaviour can be obtained both when the agent generates the exploratory pushes (pokes) and/or when the agent only observes poking actions, performed by other agent or human.

When poking an object, the object motion depends on the object's shape, weight distribution and on the support friction forces. A lot of work has already been done in the field of mechanics on controllability and planning of poking [5],[6]. Obviously, poking could easily be implemented by assuming a proper representation for the physics of the task, but such an approach relies on a priori knowledge about the action and therefore does not solve the complete learning problem. Additionally, it is sometimes difficult to obtain the model parameters using available sensors (e.g. it is very difficult to obtain friction between the object and the pusher using vision).

978-1-4244-2822-9/08/$25.00

If the physical model of the object and the action is not available, the robot has to experiment with different poking actions on the object. In this way the robot acquires new knowledge from exploration and human demonstration in the same way as infants learn their actions – performing actions on objects, i.e. playing with toys. While poking has been used to study cognitive processes before [4], our work focuses on different issues, that is learning complete controllers, whereas Fitzpatrick et al. were primarily concerned with extracting object properties associated with poking actions.

After learning, the robot can use the newly acquired knowledge in order to poke an object in a specified direction. The robot is able to reason how and where an object has to be pushed to move where desired. Our implementation can be divided in two parts. Firstly, the robot learns how an object moves when it is poked from a certain position and from a certain direction. This can be accomplished by experimenting with different poking actions, in which the robot pushes the object several times from different directions and at different locations on the object boundary. During this process the agent builds a knowledge base, which describes the relationship between the point and angle of push on the one side and the actual object movement on the other side. Secondly, the acquired poking knowledge is used to control the object movement, i. e. to push the object along a prespecified trajectory.

II. Methods

The method for learning poking action has been implemented on a humanoid robot Hoap-3 (Fig. 1). It is 60 cm tall, 9 kg heavy robot equipped with CCD cameras, microphone, foot load sensors and distance sensors. It has 6 DOF in each leg/foot, 1 DOF in the waist, 3 DOF in the neck and 5 DOF in each arm.

As already stated, the goal of the robot is to learn the result of a poking action. It starts by experimenting with different poking actions applied to different objects placed on a table. Afterwards the robot uses the acquired knowledge to reason about the object movement with respect to the performed action. The reasoning should be used later to find the right poking action in order to move the object as desired.

The scene (experiment) has been realized as follows (see Fig. 1). The robot stands at a table and uses a tool to poke the object on the table. The objects used in the experiments are planar polygonal objects. To simplify the environment only one object is placed on a table at a time. To realize one point poking actions, the robot holds a tool in its hand. It is a stick, which increases the robot's workspace. At the end of the tool we have mounted a cylinder, which assures a one point push. The part of the tool which has been used for pushing (the cylinder) will be denoted as a *pusher*. The robot uses only one arm in this experiment. Otherwise, the robot is fixed in the environment. To measure the position and the orientation of the object on the table, the robot uses stereo cameras mounted on its head.

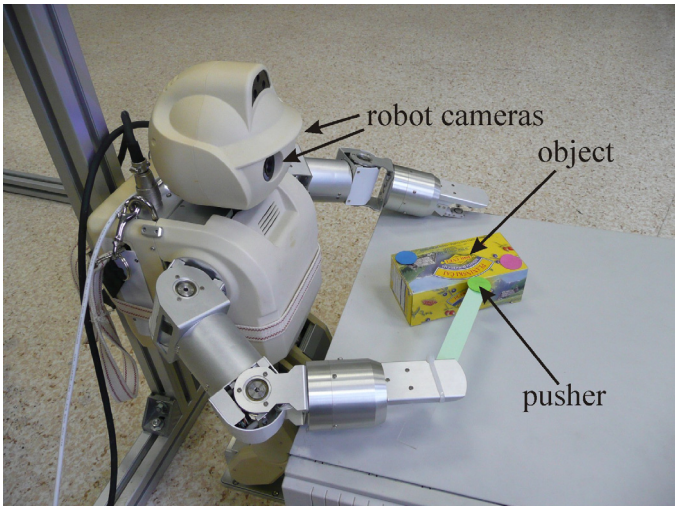

Fig. 1: Robot during pushing action

III. Vision system

All objects used for poking were placed on a table. The table is planar, which makes the design of a vision system much simpler. To acquire positions and orientations of the object on the table, it is sufficient to use one camera. We used colour markers to simplify vision processing. We placed two markers on each object in order to extract both position and orientation of the object. Additionally, we marked the pusher to enable visual servoing.

To define the transformation (mapping) between the image coordinate system and the world coordinate system, where the robot is situated in, a calibration has to be performed. The mapping incorporates extrinsic (position and orientation of the camera) and intrinsic (focal lengths, pixel size, image centre) camera parameters. We could estimate the intrinsic and extrinsic camera parameters using other methods (kinematics, chess board…). However, in our case we rather used the robot to move the marker in front of the vision system. Using more than 100 measurements, we calculated the transformation matrix using least-square error methods.

The use of the robot in the calibration process makes the system much simpler and more flexible. Additionally, the result can be more precise, since same data is used during the calibration as well as during poking. So the same sensor uncertainty appears twice and the errors can cancel each other (e.g. kinematics data, vision data…). That means that the same kinematic error which appears during calibration, will also appear during the control – and that will already be included and handled in the calibration process.

The accuracy of the robot and the vision system is rather low. To improve the precision of motion, we had to use visual servoing techniques. To determine the position of the pusher using vision system we put a marker on top of it. Since the vision is calibrated only in one plane, the marker has to lie in that plane. This is true during poking; however, when the pusher is above of the object, the position is not totally correct any more. In this case the robot kinematics is more accurate. To solve this, we have implemented a continuous switch

between kinematics and vision information, where the amount of each depends on the distance from the calibrated plane.

IV. Learning

In the first phase of the process, the robot has to learn the behaviour of an object, when the object is poked from a certain direction and at a certain angle (see Fig. 2).

In this phase the robot experiments with different poking actions. The robot has to push an object from different sides of an object as well as under different angles. In the beginning of the process the robot (agent) has no knowledge about the poking action and the robot experiments with different poking actions completely randomly. Afterwards, the robot should only perform action at the points (or angles) where the knowledge (or the model) is not precise enough.

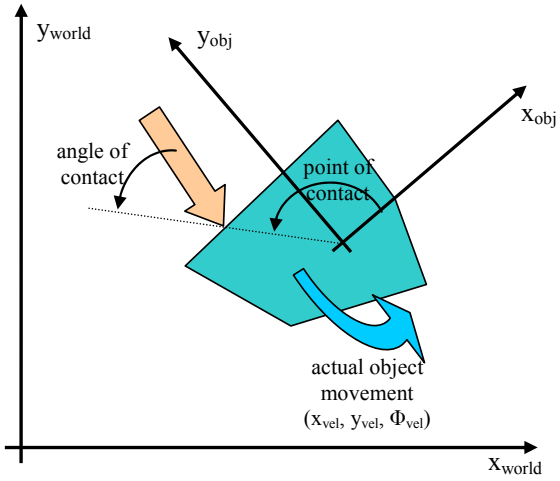

Fig. 2: Schematics of a poking action

After applying a poking action, an object accelerates and changes its position and orientation. Since the objects are very light and the friction between the object and the table is relatively high, we can neglect the dynamic properties of the motion. Typical response of the object is shown in Fig. 3. The object velocity settles in less that 200 ms. The reason for very noisy object velocity is that we have used vision to obtain the position of the markers. However, since training can be done off-line, the data can be filtered and processed before the use.

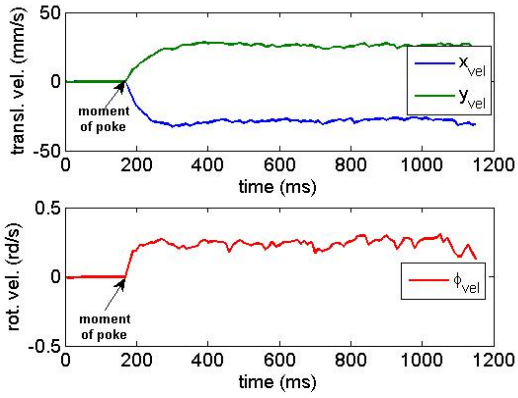

Fig. 3: Typical response (velocity) or an object after applying a poking action

Due to the fact that an object settles its response in a very short time, we can model object behaviour statically. We model the relationship between the displacements of the pusher and the displacement of an object. The displacements of the pusher are expressed by two parameters: the point and the angle of contact on the object boundary. The velocity of the pusher is kept constant. The point of contact is expressed as the angle between the line segment connecting the point of contact and the centre of the object and the x-axis of the object's coordinate system. Similarly, the angle of a contact is expressed as the angle between the pushing direction and the tangent at the point of contact.

The response of an object is represented by three parameters, i.e. the planar velocity of the object centre and the rotational velocity about the centre point on the object. The agent's view of the experiment is shown in Fig. 4.

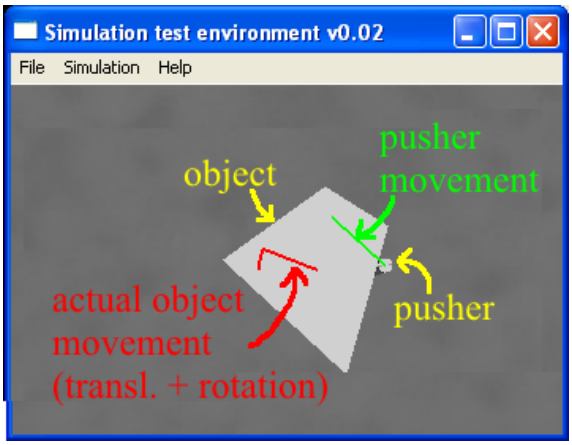

Fig. 4: Agent view of a scene during learning

To represent the relationship between the point and the angle of a contact and the object response, we used a neural network with two hidden layers. Based on the measurements we performed an optimization process and compared neural networks with different number of neurons in each layer. Since we could acquire quite a large set of data, one part of data has been used for learning while the other part of data has been used for verification of the neural network.

The result of the comparison of different neural networks showed that the most reasonable selection is to use different networks for different outputs. The resulting neural networks, which model the object behaviour satisfactory and are still simple enough, are shown in the following table:

	NN inputs	Number of neurons in 1st hid. layer	Number of neurons in 2nd hid. layer	NN output
x	position and angle	11	3	Velocity in x dir.
y	position and angle	12	6	Velocity in y dir.
Φ	position and angle	9	4	Velocity in Φ dir.

V. CONTROLLING

After the learning phase is completed, the robot can generate poking actions to move an object in the desired direction. The task of the robot in this phase is to perform a set of poking actions in order to bring an object where desired. Here, a higher-level motion planner should provide the desired movement of the object. The agent has to find out where and how to poke the object to achieve motion close to the desired one. During this process the agent has to use the knowledge acquired in the learning phase. Agent view of the poking scene is shown in Fig. 5.

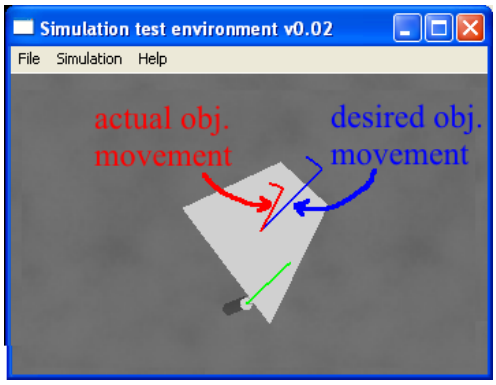

Fig. 5: Agent view of a scene during controlling object movement

Note that, the robot can not always achieve the desired velocity. The desired velocity is or can be defined in three directions (three DOFs); however, the robot controls only two input variables, the point and the angle of contact. Additionally, any arbitrary velocity vector can not be achieved due to the physical limitations of the action (this is still a nonprehensile action).

To achieve optimal motion in a given situation, the agent needs to optimize a criterion function with respect to the point and angle of push, e. g. the weighted square error between the desired motion and the predicted one. Thus we need to find a global minimum of the following function:

$$e = (\mathbf{W}(X_{des} - X_{pred}))^2, \qquad (1)$$

where X_{des} represents the desired motion in all three DOFs and X_{pred} represents the motion of the object which is predicted by the neural network, respectively. **W** is a weight specifying the importance of each direction.

It is easy to find a local minimum of a function defined in (1) using classical optimisation techniques. However, to find a better solution and to avoid falling into local minima, we run the optimization process several times with different initial values in order to find a better solution or even the global minimum. The solution, which might not be the globally optimal one, results in motion that is usually close to the desired motion. After applying the poking action, the object pose changes and the new point of contact and angle of push are determined, which can be better than the previous ones.

Note that only the directions of object movement are considered in the optimisation. The amplitudes of velocities can be modulated by stronger (faster) pushing action.

VI. RESULTS

The proposed approach has been implemented on a Mitsubishi Pa-10 industrial type robot and on a humanoid robot HOAP-3. The accuracy and the workspace of the Mitsubishi robot are much larger than in the case of the Hoap robot; therefore, it has been much more straightforward to perform and to verify the learning and the control process.

On the other hand the experiments performed on a Hoap robot took us much more time and effort. In the experiments we used only the right arm, which has five DOFs. Technically, to achieve a pushing action with a cylinder (pusher), five DOFs are necessary. Three DOFs are needed to control the position of the pusher and two DOF are needed to control the rotations. One DOF of rotation about the cylinder axis is not important and therefore does not need to be controlled. The robot's right arm also has five DOFs.

Since the robot is rather small and there are no redundant DOFs very small workspace can be achieved. To improve that, we have treated the orientations as less significant and have controlled them in the null space. Additionally, we have used two tools in the same robot hand. One tool has been mounded in such a way that the robot achieved points near the body, while the other tool enables achieving points more far away. At the ends of both tools two cylinders has been mounted, which were used for pushing.

To control the robot we used a velocity based task controller and a quaternion control in the null space for both orientations.

We performed the learning process on a set of different planar objects shown in Fig. 6. Fig. 7 shows the response of a square object in all three directions with respect to the point and angle of contact.

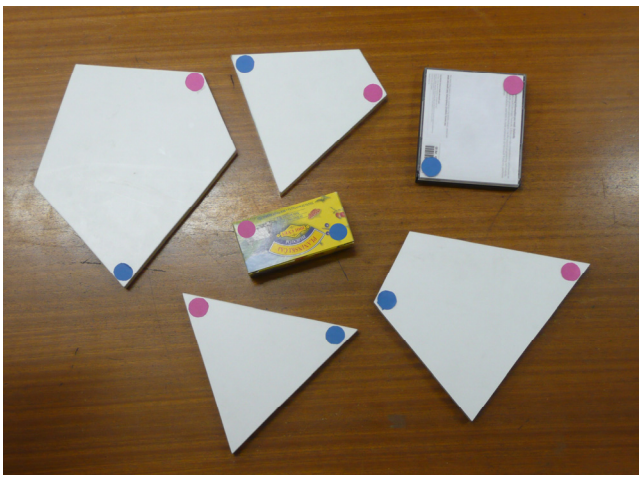

Fig. 6: A set of objects that were used for learning

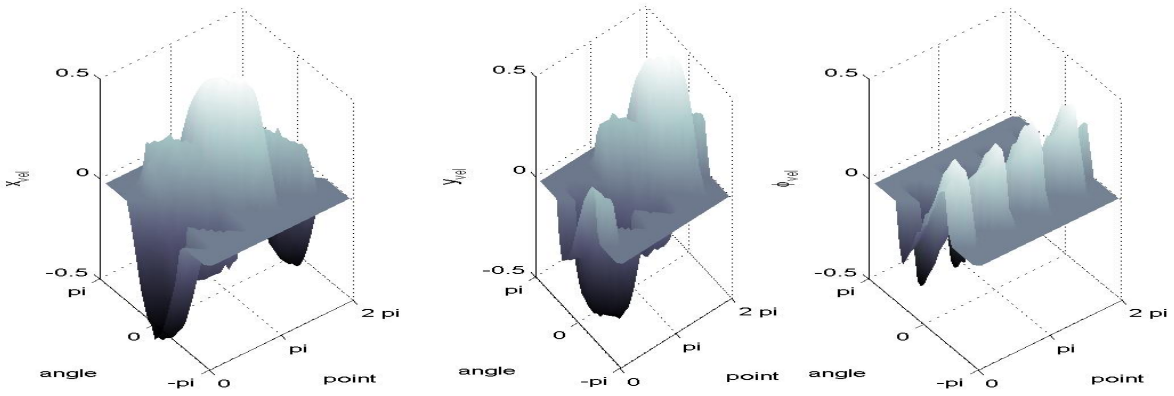

Fig. 7: Relationship between point and angle of contact and object response

In the learning process the robot generated random poking action from all sides of objects (point of contact on the boundary travels from 0 to 2π) and from different angles in the range from – π to π. The robot also generated actions that do not result in any object motion (for all angles that are less than – π/2 or more than + π/2), where the motion of the pusher is directed away from the object. Based on our experiences we know that such actions do not result in any motion. The learning would be much faster if we provide as much knowledge as possible; however, we wanted that the agent learns this rule by itself, without any hard coding. The goal of our work is to develop a system which could develop cognitive ability of the robot - a system where a robot could evolve in a more intelligent machine. Therefore, such things should not be hardcoded.

To validate the learned controller, we defined a task of consecutive point-to-point movements, where the object orientation was not important. In case of Mitsubishi robot the trajectory has been more complex. The object had to move between the corners of the square of size 30 cm x 30 cm (see Fig. 8). In Fig. 8, points are marked by small circles. Fig. 8 also shows the actual movement of the object (blue line). The object starts from initial position an moves to point P1, then moves through P2, P3 and to P4, and finally returns to P1. The movement of the object is not very precise because the action learning has not been perfect. In any case, we cannot expect that a nonprehensile action would result in a movement with the same precision as an action with a grasp (with full control over an object). Nevertheless, the learned poking action is precise enough to keep the object within a few centimetres of the desired path.

In the case of the Hoap robot the trajectory has been much simpler. The robot had to move a smaller object to a point in space (marked with a red circle in Fig. 9). The trajectory has been defined in such a way that the robot needed to use both tools in order to be able to achieve the task.

Fig. 10 shows the rotation of the object during the whole movement cycle. Since the rotation of the object has not been controlled, it is changing randomly. This was achieved by not including the object rotation in the process of searching the most appropriate point for pushing. The weight **W** was, therefore, set to:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad (2)$$

where 0 as the last diagonal element correspond to the object rotation. Fig. 11 shows the points of contact and the angles of contact during the whole cycle. It can be seen that point and angle of contract change significantly depending on a current object state.

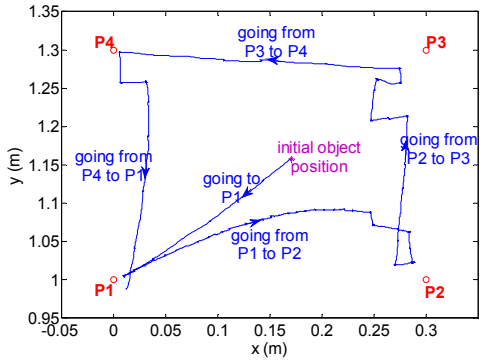

Fig. 8: Object positions during point–to-point movement between the corners of a square (experiments on a Mitsubishi robot)

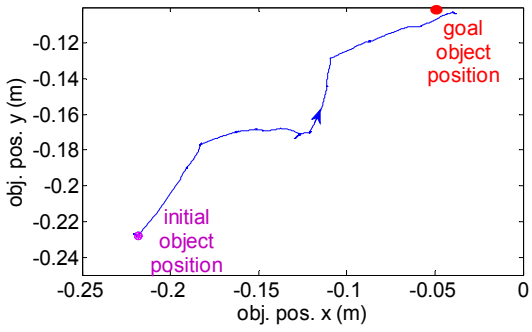

Fig. 9: Object positions during point–to-point movement (experiments on a Hoap-3 robot)

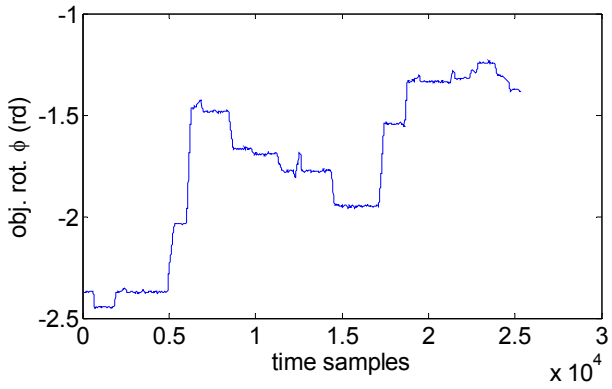

Fig. 10: Object rotation during point-to-point movement

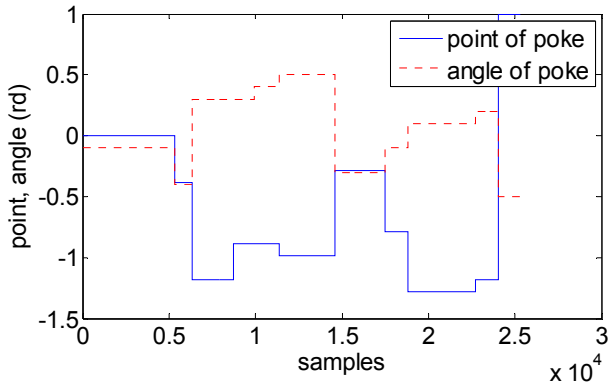

Fig. 11: Point and angle of contact during point to point movement

VII. DISCUSSION AND FUTURE WORK

In this paper we described how to learn the relationship between a point and an angle of a poke and the response of an object. The robot acquired this knowledge by exploration without having any prior knowledge about the action. While very precise learning of pushing actions can take a very long time, the agent learns a rough but reasonable approximation of the action already after a few explorative pushes. This initial knowledge can already be used for a rather rough control of the object movement. Next, while controlling the motion the robot can update its knowledge base by observing the actual movement of the object. Thus the relationship between the desired and the actual object motion gradually becomes more accurate and the control of the object movement direction improves. Additionally, to make the learning of poking actions more optimal, human instructor can demonstrate the most representative pokes (e.g. perpendicular pokes from a few different sides).

However, the knowledge, that the robot obtained by exploration, is useful only for the object that was used for training. Currently, for each new object exploration has to start from the beginning, thus it takes a long time before a satisfactory large object library is built. There is no generalisation. Our plan for the future is to learn more general pushing controllers instead of learning the behaviour of every object. The generalisation can be achieved by performing many different pushing actions on different objects. The actions and object has to differ in relevant characteristics in order to identify the general pushing rule. To solve such problems, some authors use the recurrent neural networks with parametric bias [7],[8]. In these works, static images of objects are linked to dynamic features of objects.

Using such general laws, people can predict the movement. However, when the actual movement of the object differs from the predicted one, humans include the feedback loop and adapt their actions in order to achieve the desired motion of the object. In the same way closed loop control has been used in our work. The robot/agent can predict only the approximate behaviour of the object. Due to the object properties that has not been modelled or cannot be measured, e.g. friction, mass distribution, etc., the actual motion differs and the robot has to adapt its motion to improve the motion of an object.

In summary, we realized the process of associating object-action events through an explorative, self emergent process. Such processes are of great importance for the early cognition. No knowledge about pushing was provided to the robot. We only provided rules about how to explore the environment and the robot obtained the controller by itself.

ACKNOWLEDGMENT

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657).

REFERENCES

[1] Dennett, D. C., "Cognitive wheels: The frame problem of AI," In Hook-way, C., editor, Minds, machines and evolution, 129–151. Cambridge University Press, 1984.

[2] M. Lungarella, G. Metta, R. Pfeifer and G. Sandini, "Developmental robotics: a survey," Connection Science, vol. 15, no. 4, pp. 151 – 190, 2003

[3] Integrated project proposal "Percepetion, Action, and Cognition Through Learning of Object-Action Complexes (PACO-PLUS), Cognitive Systems, FP6-2004-IST-4-2.4.8.

[4] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning About Objects Through Action - Initial Steps Towards Artificial Cognition", P roc. 2003 IEEE Int. Conf. Robotics and Automation, Taipei, Taiwan, pp. 3140-3145, September 2003.

[5] K. M. Lynch and M. T. Mason, "Stable pushing: mechanics, controllability, and planning," The international journal of robotics research, vol. 15, no. 6., pp. 533-556, 1996

[6] Q. Li and S. Payandeh, "Manipulation of convex objects via two-agent point-contact push,", The international journal of robotics research, vol. 26, no. 4, pp. 377-403, 2007

[7] S. Nishide, T. Ogata, J. Tani, K. Komatani and H. G. Okuno, "Predicting Object Dynamics from Visual Images through Active Sensing Experiences," IEEE Int. conf. on Rob. and Autom., pp. 2501-2506, Italy, 2007

[8] J. Tani and M. Ito, "Self-Organization of Behavioral Primitives as Multiple Attractor Dynamics: A Robot Experiment," IEEE Trans. on SMC Part A, Vol. 33, No. 4, pp. 481-488. 2003