Prediction of action outcomes using an object model

Federico Ruiz-Ugalde, Gordon Cheng, Michael Beetz
Intelligent Autonomous Systems and Institute for Cognitive Systems, Technische Universität München
ruizf@cs.tum.edu, gordon@tum.de, beetz@cs.tum.edu

***Abstract*— When a robot wants to manipulate an object, it needs to know what action to execute to obtain the desired result. In most of the cases, the actions that can be applied to an object consist of exerting forces to it. If a robot is able to predict what will happen to an object when some force is applied to it, then it's possible to build a controller that solves the inverse problem of what force needs to be applied in order to get a desired result. To accomplish this, the first task is to build an object model and second to get the right parameters for it. The goals of this paper are 1) to demonstrate the use of an object model to predict outcomes of actions, and 2) to adapt this model to an specific object instance for a specific robot.**

I. Introduction

The simple task of pushing an object can produce a very rich set of different types of outcomes. An object will topple or slide or both, depending on object characteristics like shape, weight, friction coefficient(with other objects) and also on the forces exerted on the object. The object can also move for longer distances if pushed too hard. We can think of the object as a system that takes the forces exerted to it and produces outcomes, i.e. a control system. In the context of robot manipulation it is desired that a robot is able to accomplish a certain manipulation task where objects are involved, therefore the robot needs to be able to control the outcome of actions applied to this objects. The intent of our work is to build a control system that is able to control outcomes by exerting appropriate forces to the object.

Like with any other control problem, we need to describe the system to control, i.e. the object. We call this description, the object model. With this model the robot can predict what will be the outcomes of the actions applied to this object. Apart from only predicting the outcomes of actions, the object model determines how can the object be fully used, but the actual usefulness of the object is limited by the possible different ways the robot can act on the object, this is the input domain of actions to this object. This input domain will define a output domain of outcomes, i.e. what could possibly do the robot with this object. If the robot is aware of this input domain and can predict the corresponding outcomes, then we can say that the robot is aware of the affordances of this object [1].

The inputs exerted to the object will be in general forces. In case of a humanoid robot this forces can be produced using pushing actions with the fingers, hands or arms, or by grasping the object and then using the arm to exert the forces. This actions can be produce by different types of controllers in the robot, but from the object's perspective this actions will always be just some forces. This means that for object control, the arm or hand controllers are not relevant as long as it's able to execute the object control commanded action, although the arm and hand controllers will be part of the factors that will define the input domain of actions, which in turn is important for defining the affordances of the object. As an example, if we want the robot to apply a static rotation to the object, it may probably need to grab the object first and then rotate, but for pushing the object, it's enough to apply some force with only one finger.

Strong evidence suggests that language and actions are connected together [2], we believe that one way to make this connection is by mapping the input and output domain regions with words and sentences. Sentences like "push the ice tea box away without toppling it", "open the ice tea cup" or "pour the contents of the ice tea into the mug" can give an idea of how powerful for a robot a language can be. We refer to this as a language-action-outcome association This richness on how actions and outcomes can be expressed is reflected on the richness of an action-outcome language. (Fig. 1)

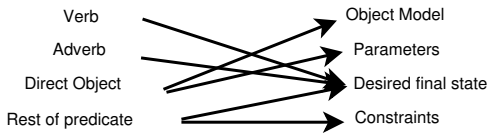

Fig. 1. Language-action-outcome association

Taking the first example sentence above one can find a mapping between the words and the object controller and it's parameters, e.g. the verb "push" would translate to an input domain region that uses a push controller to exert force, "ice tea box" determines the object model to use, "away" gives the desired outcome of the object (the object final position must change to "away"), " without toppling it" gives an extra constraint to the object controller (optimization principle) and partially defines the final outcome. (Fig. 2)

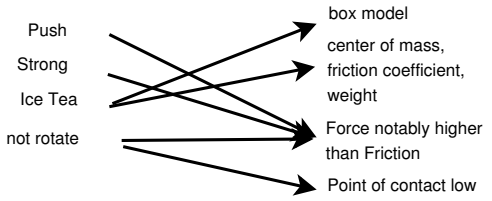

Fig. 2. Language-action-outcome association example

It's important to notice, that the sentences that involve actions manipulating objects, refer directly to the manipulated object and they usually don't need to mention what the arms or hands must do in order to accomplish the action. This reinforces the idea that language helps to specify the object controller parameters .

Figure 3 shows how the object controller fits into the complete action-outcome language system.

This paper concentrates on the object control problem. Specifically, we setup a simplified object model for a box, we show how the robot explores the object to determine the parameters of this object model for a specific box instance

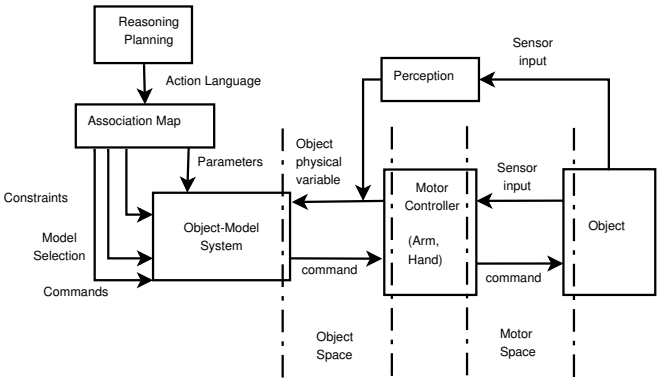

Fig. 3. Complete system

and we demonstrate the use of this object model to predict action outcomes. The model works in the 3D case, i.e. it can predict outcomes from forces in 3D space. Also instead of using vision as feedback, we are able to predict outcomes by only using the torque and position sensors from the fingers to feel movement of the object.

II. Related Work

A lot of work has been done in the direction of acquiring human motion capabilities. The use of learning from demonstration for acquiring movement primitives has been well know to provide good results in the context of imitation, and has given extraordinary motion capabilities to robots [3]. Advancements in robot technology like the DLR-III lightweight arms and DLR-HIT hands [4], allow robots to interact in more flexible ways with the environment. Since object control is more centered on the object, the arm and hands motion systems are not the main concern, but more central is the object and how to interact with it. This is much related to the concept of affordances [5] [1].

Affordances are action possibilities that are perceivable by an actor. The idea is that given an object and an actor, the affordances suggests how this object can be used by this actor. In this direction work has been done in providing robots with affordances. In [6], a robot observes movements executed by himself of pushing/pulling and poking of an object, then it associates this actions with the corresponding outcomes, and finally is able to execute commands to achieve a desired outcome. Vision is used to feedback the information of the generated movement, a histogram based learning algorithm is used to learn the model.

Also building on top of the affordances concept, is the notion of OAC (object-action-complexes) [7]. OACs state, that objects are constantly suggesting actions (to be realized with them or upon them) which will ultimately transform the object to a new state. It can also be interpreted as that the action transforms the usefulness of the object. One of the first examples of OAC applied to a robot is shown in [8], where a robot learns a pushing rule, i.e. how an object reacts if its pushed, and also learns the inverse pushing rule, i.e. how to push an object in order to accomplish some goal. They build the pushing rule with an artificial neural network. The problem is solved for the planar case (3 DOF), and also uses vision as the outcome feedback.

Our contributions are focused on using mechanical models to build the object models. This models are extremely compact and require determining very few model parameters. Also the exploration process for finding this parameters requires very few experiments, and the robot could even be able to initially just guess a value for this parameters or retrieve one from a knowledge database, and start right away to play with the object, already by touching once the object the robot could get an update to the value that is good enough to start predicting outcomes correctly.

III. Object Model

The object model describes how the object reacts to certain physical inputs. The inputs to the object are any physical quantities that could change the state of the object over time. For example forces on the object can change position, orientation, speed, acceleration and shape. The outputs of the object refer to physical values or properties of the object, like current position, speed, acceleration and temperature. The model describes the internal workings of the object, i.e. how inputs affects outputs. The model will be always an approximation of the real object behavior, how complete is this model depends on how good we are in measuring the outputs and producing inputs with our robot, or if we see it from another perspective, how much do our robot needs to model the object to fulfill some desired manipulation goals.

Building the model.

There are three main options for making the object model: 1) learning, 2) system identification and 3) prior system definition plus parameter identification. In 1) a robot explores the object by exerting forces (inputs) to the object and measures the corresponding changes in state (outputs), then use an appropriate machine learning algorithm to learn the model function. This is the approach used in [6][8]. In 2) the object will be explored with different inputs, analyze the outputs and based on this, generate a transfer function that tries to imitate the registered behavior of the system as much as possible (and as needed), using differential equations. And 3) makes use of known mechanical and physical equations (equations of motion, rigid body, soft body) to model the objects manually and then the robot can do a more narrow search for the parameters of this models for each object instance.

Advantage of 1) is that the robot could be able to learn new models on the fly of novel objects, also the model learnt is adjusted to the robot itself, i.e. the generated model will never be more complicated than what can be described with what the robot manipulators can sense or actuate, the disadvantages are that the search space can be huge, which in turn translates to slow learning. That is usually the reason why is so difficult to scale this systems to 3D. In 2) the disadvantages are that doing system identification in a multi-variate system can be cumbersome specially when the system has a lot of variables, also probably a lot of manual work

could be necessary to be done. In 3) the main disadvantage is that the mathematical description of the object has to be done manually, which can be time consuming and difficult in some cases, on the other hand it is only done once for a class of objects. The advantages are: That a well made model can be general enough to apply to a large quantity of types of objects and still be a very compact model with few parameters. The exploration process is reduced to only search the model parameters for the object instance, this exploration could be in some cases automatically guided or encoded in the model. The model can be easily associated with objects in a knowledge database. Because of this advantages this is the method we choose to build the object model. Because this model requires that the robot explores the object to find the parameters to the model, it allows the robot to best fit the model and therefore the object to itself (to what he observes). Our model is designed to simulate a box, with the most popular instance: an ice tea box shown in Fig. 4.

For simplification, we make to following assumptions: The box is always in contact with another object with some associated friction coefficient (box-table) and with the robot hand. The object is not deformable or if it is, the deformation is negligible. We are constraining the robot to "pushing actions", but some other actions may be possible with the current model. We suppose the object is not moving (static case), we can predict if the box will slide or rotate and how strong depending on the applied force, but this prediction is only the instantaneous value, i.e. before any movement, this is fine for slow movements where inertia doesn't play such an important role. The box model consists of three parts: friction, contact and static equilibrium.

1) Friction: Between the object and the supporting object(table) we apply the coulomb friction model (Eq. 1). The relevant parameter for friction is static friction coefficient between the object and the supporting object. Fig. 4 shows the interacting forces during friction in the simplified 2d case (the model executed on the robot works in 3D).

$$F_f \leq \mu F_n \tag{1}$$

In the case of the box, F_n is the combined force (external force plus weight) projected on the normal vector of the table plane and μ is found out by robot exploration. F_f lies always in the contact plane and with direction against the applied force. Since we are dealing only with the static case our μ is static (μ_s).

Fig. 4. Friction forces.

2) Contact: This models how the object could rotate with an axis along the supporting object surface, if it rotates at all. The important parameter here is the contact surface shape.

The axis where the object will rotate depends on the shape of the supporting base of the object and on the applied torques. What needs to be found is in which vertices of the base of the object there can be a rotation which involves no vertical movement. For each vertex of the supporting base of the object a range of rotations that will produce vertical translation is computed using the other vertices. If a torque that is applied around the current vertex, produces a counteracting force from another point (that is part of the supporting base of the object), because this point is in contact with the table, then is said that this torque will produce a vertical movement on this vertex, and then we can add this torque to the set of torques that produces rotation with vertical movement. There is a continuous range of torques because there is a continuous set of points along the object that can cause counteracting forces. To find this continuous range, the normalized torques produced by arbitrary forces at every other vertex of the base of the object is calculated, then the biggest angle smaller than $180°$ between all the possible combinations of pair of angles gives the required range. Then this range is calculated for every vertex of the supporting base plane of the object and stored as part of the object model. This algorithm is shown in pseudocode 1. Chapter 27 of [9] contains more elaborate information on contact forces.

```
torque_ranges=[]
for pivot_vertex in box_vertices:
  torques=[]
  for vertex!=pivot_vertex in box_vertices:
    torque=vertex_to_pivot_vertex_torque(vertex,
        pivot_vertex)
    torques.append(torque)
  range=max_angle_smaller_than_180(torques)
  torque_ranges.append(range)
return(torque_ranges)
```

Pseudocode 1. Calculation of range of torques for each vertex of the base of the box that produce rotations with vertical translation

3) Static equilibrium (Eq 2): This equation states that all forces and torques should sum up to zero. It helps to determine if the object will slide and/or rotate depending on the applied forces, the friction model and the contact model. Here the relevant parameter is the weight of the object.

$$\begin{aligned} \sum \overline{F} &= \overline{0} \\ \sum \overline{p} \times \overline{F} &= \overline{0} \end{aligned} \tag{2}$$

Fig. 5 shows the interacting forces, contact points and possible axis of rotation in the simplified 2d case, A and B are the contact points with the supporting object and also are the possible axis of rotation. Again the robot runs the 3d case.

Prediction using the model.

Using equation 1 and the force balance part of equation 2, the robot can determine if the box will slide or not. If the combined force (external force and weight) projected on the

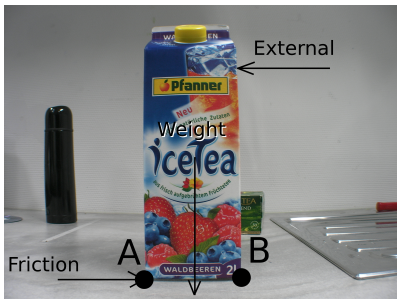

Fig. 5. Contact Model.

table plane, is bigger than the maximum friction force μF_n, then the object will slide. On the other hand, if this projected force is smaller or equal than the maximum friction force, then the object will not slide.

Apart from sliding, the object can also topple (or rotate in the vertical plane). This rotation can be around an axis defined by two of the four box base vertices, can be around an axis defined by a single vertex from the base and the produced total torque (external torque plus weight induced torque) around this point or no rotation at all. For finding this out, first the robot must compute the total applied torque (using the torque part of equation 2) projected on the table plane on each of the supporting vertices, then it must compare this torques against the torque ranges calculated using pseudo-code 1. If for one of the vertices, the applied torque lies inside of the torque range of such vertex, then this vertex will translate vertically. For the vertices where the applied torque lies outside of the torque range, then a score is given to this vertices. The score is high if the angular distance from the applied torque to the torque range area is high, the score will be lower as the distance turns smaller. The first two vertices with the highest scores are the vertices where rotation without vertical translation will occur. This means, that this two vertices will define the axis on the table where the object will rotate. If two of the three vertices have the same score, then it means that the object will rotate around the axis with the direction defined by the applied torque and passing through the the vertex that has the highest score. Rotation will not occur if the applied torque, lie inside of the torque range for all the vertices.

IV. Exploring for the parameters

In order to find the value of the parameters for the object model, the robot must do some exploration with the object, that is, the robot has to play with the object for a while. The fastest way to determine this parameters, is to have a formula that solves for this parameter directly taken from the object model, and then give the necessary inputs to the object and measure the corresponding outputs to calculate the parameter using such formula. One important thing is, that when we solve for this formula, we must try as much as possible to not depend on other unknown parameters. If we can solve the formula and still only depend on one unknown variable, then the experiment will be a simplified usage of the object which shows only a part if its behavior. Unfortunately this is not possible in most of the cases, what could be possible, is to perform first one experiment where only one parameter will be determined, and afterwards other experiments that depend in previews (but already discovered) parameters except for one. What becomes important is to find out the right order of experiments to find all the parameters one by one. Another more sophisticated way of finding the parameters is starting with a guess of all the parameters and then depending on the prediction errors adjust this parameters until all of them predict correctly the behavior of the object. This has the advantage that one doesn't have to compute the closed form solution for the model and that it can find the parameters even in situations where multiple parameters are tightly coupled together and the experiments to find this parameters will not provide each parameter alone. On another hand some parameters can be measured by other means, e.g. using vision to find out shape.

A. Box dimensions and contact points (base shape).

Instead of letting the robot explore the shape of the object (which would be a more complicated precedure), the robot takes the shape of the object from a CAD model that is also used by the vision system [10] to detect the position of the object. In this way the object model gets the base shape parameters from the CAD model and the position and orientation of this base shape from the vision system. The center of mass is infered from the dimensions of the box assuming that the box is resting in one of the sides of the box.

B. Weight

Weight is necessary for the friction and force/torque balance part of the model. There are two ways to find the weight of the object. 1) A practical approach is to grab the object then lift it and then measure the force in the force sensing robotic hand/arm. 2) Another way to find the weight is to touch the object with the robot hand in a point that will most likely tilt the box but not slide it and then measure the necessary force to start tilting to box (detect the starting of tilting with movement detection from the hand/arm and/or with vision), then use the contact and force/torque model to calculate the weight. Because there is water in the box, the box must be lying in one of its sides to let us use a simple calculation for the center of mass and there can't be any suddent movement which would cause force oscilations. Method 2 is used because our robotic hand was not strong enough to lift the ice tea box with 1 liter of water inside. One of the fingers of our robot is used as the force sensing device. Since the same force sensing device is used in all the experiments involving force sensing, then we avoided the problem of calibration between different sensing devices.

C. Friction coefficient.

From equation 1 one can easily solve for the friction coefficient: $\mu = F_f/F_n$. Once the weight is known we can find out the static friction coefficient by applying an increasing force until the object starts sliding, the force just

before this happens is the maximum static friction force. In order to detect this precisely we can use the hand/arm to measure when the object starts to move. It is important that the object has low deformation to forces and that no tilting occurs. To assure that the object will only slide and not rotate, the robot pushes the object on a very low point of the object height.

V. Experimental Setup

Our experimental scenario is the assistive kitchen [11]. Our manipulation platform (Fig. 6) consists of a robot with an omni-directional base, two 7 DOF Kuka-DLR light-weight arms, two DLR-HIT-Schunk hands and two cameras. The hands and arms are equipped with torque sensors in all the actuated joints, and impedance control in joint space is used along all the experiments. The hands have 4 fingers each with 4 joints and 3 independent DOF and 3 torque sensors.

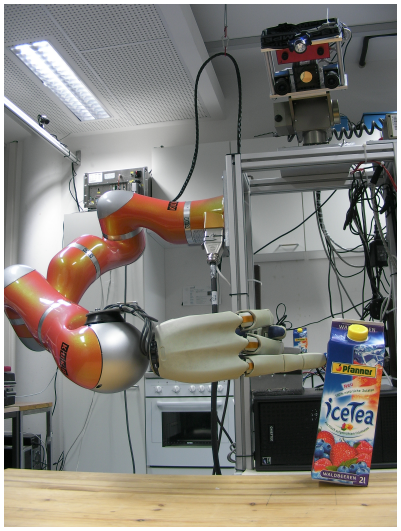

Fig. 6. Assistive kitchen robot

The Object.

We use an ice tea box that can hold up to 2 liters of liquid but during the experiments it was half full. We have a 3D CAD model of this ice tea box, but for the object model we only take the 8 vertices that represent the main box of the ice tea. This approximation will introduce some errors in our results, but as we will show they are not big. See Fig. 4

Motor Control.

In our motor control system we use the fingers as the torque sensing devices and the arms as the actuator devices. To move the hand to a specific location before touching the object, we use a point attractor system, where the arm is pulled by a velocity vector that is calculated in each control cycle. The hand is moved to a position before applying force to the object. From this position we start moving the arm using a force controlled system where a movement vector, a force magnitude and a maximum speed is given to the controller. The hand starts to accelerate in the given movement vector direction until the force magnitude or the maximum speed is reached.

	Weight	μ_s
Mean	0.9952 kg	0.3780
Std. Dev.	0.0375 $kg^{1/2}$	0.0689
Std. Dev. %	3.7721%	18.219%
Min	0.9367 kg	0.3127
Max	1.0583 kg	0.5533

TABLE I
Parameters table

Experiments.

Determining the object model parameters: The robot starts by pushing the ice tea box applying a force with the moving direction normal to one of the lateral sides of the box. This force is applied in a high position (1cm below the height of the box) to assure that the ice tea will always tilt. The robot starts moving until the finger touches the ice tea, the robot detects this situation by calculating the speed of its hand and comparing it with a low threshold speed. Once the hand is not moving anymore, the robot proceeds to inspect its speed again and constantly stores the observed forces from the finger. Then the robot starts to gradually increase the applied force. When the robot detects movement again, meaning that the ice tea started to topple, then the robot stores the last force value before moving and restarts the experiment. This experiment is repeated ten times, and the goal is to estimate the weight of the ice tea box by solving Eq. 2 for the weight torque given the external torque. The second experiment is exactly the same as the first one, only that the force is applied in the lowest height possible to be sure that the ice tea will only slide and not rotate. By solving Eq. 1 for μ and using the resulting weight calculated from the last experiment, the robot is able to estimate the static friction coefficient. This experiment also runs 10 times.

Action outcomes: In order to demonstrate the object model prediction, the robot starts to apply forces along one whole lateral side surface of the ice tea. In every point where force is applied, this force is started low, to be sure there is no sliding or rotation and then this force is gradually incremented until movement is detected by the robot. This is the only moment where an external observer intervenes to give feedback to the robot about the outcome. The robot can detect automatically if the ice tea is not moving or moving (using the hands/arms position sensors) but it can not tell if the object is sliding or toppling (there is work in progress in this direction to get this system integrated with vision using the system described in [10]).

VI. Results.

Table I shows the results for 10 repetitions of the experiments for finding the parameters μ_s and the weight.

The ice tea was previously filled with approximately one liter of water. The estimated value is also near to one liter of water (1kg). The standard deviation is low, which indicates that the procedure to find this parameter is very stable. The same can not be said about the friction coefficient. It shows a lot of variability. The reason for this may be, because of not modeled complicated behavior of the friction

surface (deformation), also the surface of the table is not completely regular and it may have regions with different friction coefficients. Its important to notice that with our box model the similarity of the estimated weight value compared to the value measured with a calibrated scale is not of such importance as the repeatability of the values measured. The reason of this is that we want the robot to be able to predict outcomes and is not our main concern to get exact physical quantities. If all the model parameters and constants are given under the same scaling factor (because they are measured by the same sensors) then this model should be able to correctly predict qualitative outcomes.

A comparison between the model predictions and actual action execution is shown in Fig. 7. The Figures show in colors the different outcomes when the box is pushed hard enough to produce a movement. As the figures show the object model is very good in predicting correct outcomes, so it's easy for the robot to determine which regions are highly possible to get the desired outcome, but it's not so accurate about the minimum forces needed in order to slide it. Most probably this is related to the fact that the static friction coefficient showed great variability, posibly because of not modeled complicated phenomena, but nevertheless this prediction is still useful if the robot want to work away from the threshold point between sliding and not sliding, which for most of the applications it's the case.

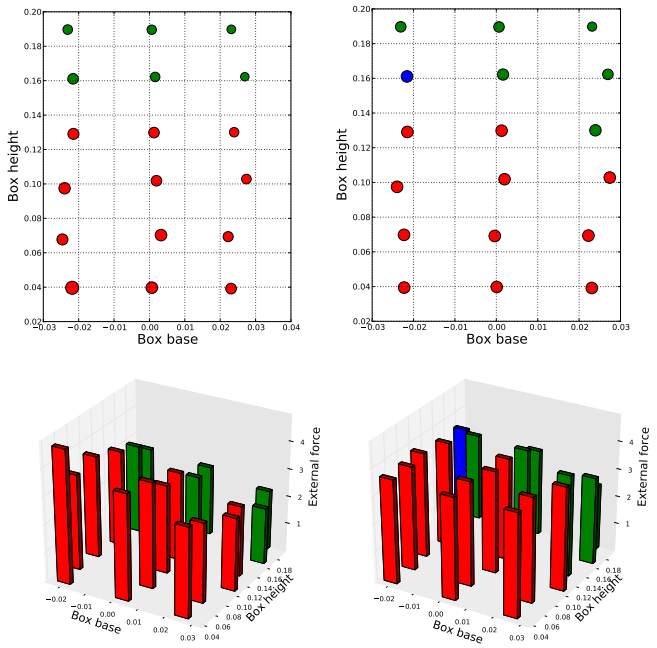

Fig. 7. Points of contact where the box slides or rotates if a force high enough is applied. Right: prediction model, left: experimentation. In the 2D pictures, the circle area indicate the force magnitude, while in the 3D pictures this force is indicated by the third coordinate. Red, green and blue indicate slide, rotation and sliding/rotation respectivaly.

VII. Discussion and Future Work

We were able to demonstrate the use of an object model to predict different types of outcomes when forces are exerted to an object. Our setup takes forces in 3D and is able to calculate outcomes without the continuous need of visual feedback. Also the task of determining the parameters using the robot itself proved to be very successful. It also showed us that there are some parameters that are specially difficult to estimate precisely, in our case the friction coefficient showed to have a high variance between measurements, indicating that possibly small variations in other physical variables could be changing the behavior of the object during parameter estimation, ultimately affecting the estimation process. Nevertheless, the robot is able to predict outcomes correctly under this circumstances, and it should be able to give useful results even having a minimal quantity of parameter estimation repetitions. While the object and the task of pushing seem to be simple, it proved to be challenging specially in the quality of the force signals from the robot and also in dealing with situations that the object model didn't consider completely. Expanding the system to allow the robot to interact with more objects, giving the robot the capability to find out the parameters without a closed solution to them, and then coupling this object manipulation capabilities with an action-outcome language are our next research goals.

References

[1] D. A. Norman, "Affordance, conventions, and design," *Interactions*, vol. 6, no. 3, pp. 38–43, 1999.

[2] R. J. Porter and J. F. Lubker, "Rapid reproduction of vowelvowel sequences: evidence for a fast and direct acousticmotoric linkage in speech," *Journal of Speech and Hearing Research*, 1980.

[3] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proceedings of the International Conference on Robotics and Automation (icra2009)*, 2009.

[4] H. Liu, P. Meusel, G. Hirzinger, M. Jin, Y. Liu, and Z. Xie, "The modular multisensory dlr-hit-hand: Hardware and software architecture," in *IEEE Transactions on Mechatronics*, 2008.

[5] J. J. Gibson, *The Theory of Affordances*. John Wiley & Sons, 1977.

[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, G. Sandini, and G. S, "Learning about objects through action - initial steps towards artificial cognition," in *In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA*, 2003, pp. 3140–3145.

[7] F. Wrgtter, A. Agostini, N. Krger, N. Shylo, and B. Porr, "Cognitive agents - a procedural perspective relying on the predictability of object-action-complexes (oacs)," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 420–432, 2009.

[8] D. Omrcen, C. Bge, T. Asfour, A. Ude, and R. Dillmann, "Autonomous acquisition of pushing actions to support object grasping with a humanoid robot," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2009.

[9] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30301-5

[10] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time cad model matching for mobile manipulation and grasping," in *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 7-10 2009.

[11] M. Beetz, F. Stulp, B. Radig, J. Bandouch, N. Blodow, M. Dolha, A. Fedrizzi, D. Jain, U. Klank, I. Kresse, A. Maldonado, Z. Marton, L. Mösenlechner, F. Ruiz, R. B. Rusu, and M. Tenorth, "The assistive kitchen — a demonstration scenario for cognitive technical systems," in *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, 2008, invited paper.