

Fast Adaptation for Effect-aware Pushing.

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—In order to produce robots that are more capable of skilled manipulation tasks, they must learn meaningful knowledge of how objects behave to external stimulus. With this knowledge a robot can predict the outcome of an action, control the object to serve a particular purpose, and together with reasoning, create or modify robot plans. In this paper we 1) build a mathematical compact model for planar sliding motion of an object, 2) show how a robot acquires the parameters of such a model; then how this is used to 3) predict pushing actions; and 4) to move an object from any¹ position and orientation to another.

I. INTRODUCTION

Competent object manipulation often requires the parameterization of actions in terms of desired and undesired action outcomes. For example, the robot might be asked to pour fluid into a bowl without spilling anything, or to hold an object without breaking it. In order to carry out such outcome-based action specifications the robots must be equipped with models that relate action parameterizations with their respective effects.

The richness of action specifications and their execution becomes particularly apparent when we consider elementary actions with continuous parameter spaces. Pushing can be used to accomplish many tasks and to increase the robustness of other tasks such as grasping [1] [2]. This is because the effects of pushing can vary even qualitatively depending on where an object is exactly pushed, how hard, on the object properties, the surface properties, etc. Having an informative action-effect model the robot can push the ice tea without tilting, push the ice tea to turn it around, push the ice tea to a specified position, and so on.



Fig. 1. TUM-Rosie robot pushing a box

We believe that performing skilled tasks with everyday objects requires deep understanding of the object’s reactions to external stimuli. This knowledge can be represented with a mathematical model that relates actions executed on the object and reactions or outcomes that the object experiences. Prediction, control, knowledge, experience and reasoning all seem to

¹kinematically possible for the robot

combine to generate intelligent behavior when manipulating objects. In this paper we concentrate with the prediction, control and low level knowledge and experience related to a particular task to give the robot an additional manipulation skill.

Mathematical models are usually based on physical and mechanical descriptions of the object’s behavior. Such descriptions usually make use of concepts such as weight, friction and force, all of which have a high level meaning. This meaningfulness of the parameters allows us to interconnect the control and prediction to high level reasoning and planning.

As an example of how our system could be used together with high level system, the sentence “Draw the ice tea near without tilting it” can define an association map (Fig. 2) which can be useful for interconnecting the object model to the high level reasoning and planning. This suggests the use of a language to communicate with the object model system.

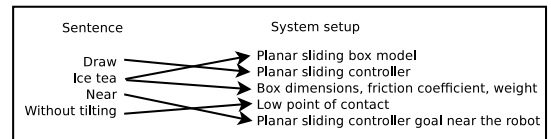


Fig. 2. Action language to system parameterization association

An important advantage of having a reasoning system, working together with prediction and control, is that it can allow a robot to recover or abort from unexpected situations (unpredicted behavior can be detected). Also the robot can reorganize the sequence of steps to realize more complicated tasks [3].

Because we desire to have the most accurate mathematical description (for better prediction), and yet only the most meaningful parameters to set (for high level reasoning), and a simple enough prediction and control system, a compromise between accuracy and simplicity of the model must be reached. For example, if the task is to slide an object from a particular position on a table to another, we desire to only have to set object parameters that the robot can measure or acquire, that are important to the task, and that can be used for reasoning. In our case, weight, dimensions and friction coefficients should be enough. We could derive a more complicated model that takes into account object deformation, liquid vibration, temperature, and so on. This may improve predictability (assuming the robot can acquire all the parameters), but will give no important improvement for the controlling and reasoning parts. In fact, it will most likely complicate things unnecessarily.

The model is constructed with respect to the object or in

object coordinates. A robot can use such a model by ways of a coordinate transformation from the robot motor and sensory space to the object's space. This is usually accomplished by inverse/forward kinematics of position, velocity and force. An advantage of this approach is that the object model can be used in any robot that such a transformation is available. This contrasts with other approaches, where raw or near-raw sensory/motor robot data is used while practicing the task to learn a model [4] [5]. This makes it more difficult to transfer this learnt task to another different robot. Our approach basically divides the problem in 1) the kinematics of the arm/hands/fingers, 2) the object model and 3) a high level reasoning and planning

Another positive feature of our system is that the model parameters are measured or estimated by robot actions. An important result of this is that, any future measurements and therefore prediction and control are calibrated or are relative to the original measured parameter's values, this in fact, physically grounds our model.

Our previous work [6] consisted on making a simple model that was able to predict if an object will slide or tilt given a force is applied to it. This paper continues in this line of research and now addresses the more complex behavior: object sliding (planar sliding). We want to answer the questions of "What happens after the object starts to slide?", "How would it slide?", "Which velocity is the object going to experiment while sliding, given an applied external force?", "How can we slide the object from a selectable position and orientation to another?".

In this paper we concentrate on the object model, parameter exploration, prediction and control. The rest of the document is organized as follows: section II discusses previous and related work on models of objects and previous planar sliding modelling results, section IV explains how our planar sliding model was derived, section III shows the experimental scenario and the different parts of our robot system, section V describes how we acquire the model parameters from robot experimentation, section VI shows how to use the model to predict action outcomes, section VII presents a control rule that slides the object to a general position and orientation and finally in section VIII we analyze our results and present how our system can be extended. The results of our work are explained in each corresponding section.

II. RELATED WORK

Our previous work [6] consisted of predicting if an object would slide or/and tilt when an external force was applied. The model used takes into consideration friction, weight and object table normal, and with this, it's able to predict if the object will tilt (also around which axis) and/or slide or not. The behavior during sliding is not described with this model. Our previous and current work complement each other, because they deal with different types of behavior, and together they generate a more complete object model. Current work is concentrated more on the moving behavior of the object (quasi-static),

dealing with velocities and forces, while previous work used static analysis to derive conclusions.

Previous works, where a robot pokes with an object that slides on a table, learn how the object reacts to these actions by means of machine learning algorithms [5], [4]. Some of these works refer to a concept so called "Object-Action-Complexes" [7], which states that objects and actions inter-depend. These systems can learn a very good mapping between actions and effects, but no way of generalizing the same learnt experience to other objects or robots is shown. In particular 1) A robot would have to practice again with every single different object to learn a correct map again, 2) A method to interconnect this learnt behavior to a reasoning system is missing, 3) Training sets for learning can also be big, 4) Because learning usually happens in retinal space (instead of 3D), changing the setting of the experiment (e.g. height of the table), can affect the performance of the system in unpredictable ways and 5) to be shown is how these systems deal with every day objects.

Model-based planar sliding has been a problem analyzed extensively. There exists a model that describe how the velocity direction relates to the applied force (applied in a particular position) [8]. With this model one can build a Limit Surface (LS) by sampling every center of rotation (COR) that the object could possibly have. Also, it's possible to find out the force vector that is necessary to start motion. To obtain the velocity direction from the LS, the surface normal must be extracted. One way to do this is using a computer graphics algorithm that extracts approximated normals from meshes, another way would be by using a parametric description of the LS [9] and then calculating the normal of it. The fit is not perfect, but good enough for most situations. The precision of the fit decreases when the contact surface is strongly discrete or strongly elongated.

Other work [10] explains the qualitative characteristics of pushing objects. In particular, it's described under which circumstances an object will turn to the right, to the left, if the finger will slide to the right or the left, and so on. We extend this by deriving a kinematic model to describe the same but quantitatively. An alternative quantitative derivation is available [11].

Using Goyal's models [8] (or simplifications of it), stable pushing tasks have been accomplished [12][13][14]. These solutions have in common that they try to exploit the friction between the actuator and the objects to achieve naturally stable pushing; then they use a motion planner to guide the objects to the desired goal. The planners are careful to maintain the system in this stable pushing configuration by respecting some minimum and maximum control values. Instability happens when the manipulator loses fixed relative position to the object. In [15] uncertainty and robustness are used to slide an object using robust stable "motion primitives", they use a simplified three-support-points friction model. The stable motion primitives always involve two (or more) pushing points (PP), and they explain that when there's only one PP, robustness is lost and motion becomes "unpredictable". Since we are using a single pushing point of contact, and only forces

and not torques to push the object, our system is working in an unstable configuration, for this reason a stabilizing feedback controller is needed (Sec. VII).

Previous work on feedback controllers for planar sliding with only one contact consist on 1) using a linearized model that only works for a particular PP and can do trajectory tracking control [16] and, 2) using a finger tip that can detect angle of touch and therefore orientation of the object, execute a “stable constant trajectory” controller, which basically means maintaining the angle between the object and the finger constant, but no direct control over the trajectory is explained [11]. We want to extend on this with a point stabilizing feedback controller.

III. EXPERIMENTAL SETUP AND SYSTEM DESCRIPTION

Our experimental scenario is a common kitchen, where various everyday objects are commonly used for experiments. For pushing, our type of objects are boxes of various dimensions. In particular we use an ice tea box (with liquid inside), a coffee package (full) and a cereal box (more than half empty) (Fig. 3)



Fig. 3. Experiment objects

The place where the pushing experiments take place, is a kitchen table, where the robot is able to slide the objects freely around. This table is supposed to be horizontal, and therefore, we can assume that gravity is affecting the objects only vertically.

Our robot (TUM-Rosie) has a set of Kuka-DLR arms and DLR-HIT hands, which both have impedance control. Torque values are available for each joint. We use the finger torque sensors to calculate the finger tip forces. A marker tracking system was used to detect the objects position, but we are planning to use a CAD+texture object tracker in the near future. Arm movements are generated with a close-loop inverse velocity kinematics with a potential field for obstacle avoidance and goal attraction [17]. The system developed integrates the marker perception, arm/finger motion generation, tip force estimation, our new prediction, control and exploration modules together.

IV. OBJECT MODEL

Our strategy for building the model is as follows: 1) Extract the normal to the LS surface (we will use an ellipsoid approximation of the LS [9]). This will describe the velocity-direction vs applied forces on the center of mass (COM or o): $\hat{\mathbf{n}}_o$ vs \mathbf{F}_o . 2) Use a wrench to translate the applied force from the pushing point (from now on c) to the COM: \mathbf{F}_c vs \mathbf{F}_o .

3) Combining step 2 with the LS ellipsoid equation we find a new LS with respect to c . If we take the normal of it we get the relation : $\hat{\mathbf{n}}_c$ vs \mathbf{F}_c . 4) Calculate the twist between c and COM. 5) Take the results from all previous steps to obtain the relation: \mathbf{v}_o vs \mathbf{v}_c .

The following assumptions will be made throughout the rest of the paper: 1) Friction surfaces are regular (uniform friction coefficient). 2) The finger tip is assumed to be a point (the robot continuously corrects the finger tip orientation to maintain a fixed pose with respect the the touching surface). 3) Object-table weight distribution is uniform. 4) Object-table surface shape is rectangular. 5) The object’s centroid and center of mass (COM) are the same and therefore the object’s (COF) is the projection of the object’s centroid in the frictional plane (we refer to COM as equal to COF). 6) Finger tip can’t apply torques, only forces.

For reference, the formulas for sampling the LS (1a), (1b) and the ellipsoid approximation (1c) [9] are:

$$F_x = \int_A f_{ax} da; \quad F_y = \int_A f_{ay} da \quad (1a)$$

$$M = \int_A (r_{ax} f_{ay} - r_{ay} f_{ax}) da \quad (1b)$$

$$1 = \frac{F_{xo}^2}{f_{max}^2} + \frac{F_{yo}^2}{f_{max}^2} + \frac{M_o^2}{m_{max}^2}; \quad \text{with } \mathbf{F}_o = [F_{xo}, F_{yo}, M_o] \quad (1c)$$

To find the maximum planar and rotation friction (f_{max} , m_{max}) we use the original LS equations (1a,1b) for pure translational motion (to get f_{max}) and afterwards for pure rotation motion (to get m_{max} using maxima), for a rectangular base shape:

$$m_{max} = \frac{\mu_{ot} f_n}{l_1 l_2^2} \left(l_2^3 \operatorname{arcsinh}\left(\frac{l_1}{l_2}\right) + l_1^3 \operatorname{arcsinh}\left(\frac{l_2}{l_1}\right) + 2 l_1 l_2 \sqrt{l_2^2 + l_1^2} \right) \quad (2a)$$

$$f_{max} = \mu_{ot} f_n \quad (2b)$$

Where l_1, l_2 are the box base dimensions, f_n the normal force ($f_n = \text{weight} * g$) and μ_{ot} the object-table friction coefficient.

1) *velocity direction vs applied forces in COM $\hat{\mathbf{n}}_o$ vs \mathbf{F}_o :* Calculate the normal to the ellipsoid surface (1c):

$$\mathbf{n}_o = \nabla f(F_{xo}, F_{yo}, M_o) = \left[\frac{2 F_{xo}}{f_{max}^2} \quad \frac{2 F_{yo}}{f_{max}^2} \quad \frac{2 M_o}{m_{max}^2} \right] \quad (3a)$$

$$\mathbf{n}_o = \mathbf{A} \mathbf{F}_o, \quad \mathbf{A} = \begin{bmatrix} \frac{2}{f_{max}^2} & 0 & 0 \\ 0 & \frac{2}{f_{max}^2} & 0 \\ 0 & 0 & \frac{2}{m_{max}^2} \end{bmatrix} \quad (3b)$$

$$\hat{\mathbf{n}}_o = \frac{\mathbf{n}_o}{\|\mathbf{n}_o\|} \quad (3c)$$

2) *Wrench to translate \mathbf{F}_c to \mathbf{F}_o :*

$$M_c = M_o - \mathbf{c}_{xy} \times \mathbf{F}_c \quad (4a)$$

$$\mathbf{F}_{o_{xy}} = \mathbf{F}_c \quad (4b)$$

\mathbf{F}_c is the (xy-component) force vector applied on c , M 's are the torques in c and COM. Since the finger tip can't apply torques:

$$M_c = 0 \Rightarrow M_o = \mathbf{c} \times \mathbf{F}_c \quad (5a)$$

$$M_o = c_x F_{c_y} - c_y F_{c_x} \quad (5b)$$

In matrix notation:

$$\mathbf{F}_o = \mathbf{C} \mathbf{F}_c; \text{ with } \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -c_y & c_x \end{bmatrix} \quad (6)$$

3) *velocity direction vs applied forces in c*: $\hat{\mathbf{n}}_c$ vs \mathbf{F}_c :
Combining eq.(5a) with eq. (1c):

$$M_o = m_{max} \sqrt{1 - \left(\frac{F_{ox}^2}{f_{max}^2} + \frac{F_{oy}^2}{f_{max}^2} \right)} = c_x F_{c_y} - c_y F_{c_x} \quad (7)$$

But since (4b), $F_{ox} = F_{cx} = F_x$ and $F_{oy} = F_{cy} = F_y$, then grouping F terms:

$$m_{max}^2 = \left(\frac{m_{max}^2}{f_{max}^2} + c_y^2 \right) F_x^2 + \left(\frac{m_{max}^2}{f_{max}^2} + c_x^2 \right) F_y^2 - 2 c_x c_y F_x F_y \quad (8)$$

This formula gives an analytical description of the LS on any particular pushing point. We effectively moved the LS from the COM to the c .

Calculating the normal of (8), we get \mathbf{F}_c vs \mathbf{n}_c :

$$\mathbf{n}_c = \begin{bmatrix} 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_y^2 \right) F_x - 2 c_x c_y F_y \\ -2 c_x c_y F_x + 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_x^2 \right) F_y \end{bmatrix} \quad (9)$$

Or in matrix notation:

$$\mathbf{n}_c = \mathbf{B} \mathbf{F}_c, \mathbf{B} = \begin{bmatrix} 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_y^2 \right) & -2 c_x c_y \\ -2 c_x c_y & 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_x^2 \right) \end{bmatrix} \quad (10a)$$

$$\hat{\mathbf{n}}_c = \frac{\mathbf{n}_c}{\|\mathbf{n}_c\|} \quad (10b)$$

4) *Twist between c and COM*:

$$\mathbf{v}_c = \mathbf{v}_o + \mathbf{c} \times \mathbf{w} \quad (11)$$

Where \mathbf{w} is the rotation velocity vector (equal in COM and c), \mathbf{v}_c and \mathbf{v}_o are the velocities in c and COM. In matrix notation:

$$\mathbf{v}_c = \mathbf{D} \mathbf{v}_o; \text{ with } \mathbf{D} = \begin{bmatrix} 1 & 0 & -c_y \\ 0 & 1 & c_x \end{bmatrix} \quad (12)$$

5) *Relation between velocity in c and velocity in COM*, \mathbf{v}_o vs \mathbf{v}_c : If we put eqs. (3b), (10a) and (6) together, we get:

$$\mathbf{n}_o = \mathbf{A} \mathbf{C} \mathbf{B}^{-1} \mathbf{n}_c \quad (13a)$$

$$\hat{\mathbf{n}}_o = \frac{\mathbf{n}_o}{\|\mathbf{n}_o\|} \quad (13b)$$

Now we need to find the factors relating \mathbf{v}_o with \mathbf{n}_o and \mathbf{v}_c with \mathbf{n}_c . Using eqs. (12) and (13):

$$\mathbf{v}_o = \alpha_1 * \mathbf{n}_o \quad (14a)$$

$$\mathbf{v}_c = \alpha_2 * \mathbf{n}_c \quad (14b)$$

$$\mathbf{v}_o = \frac{\alpha_1 \mathbf{A} \mathbf{C} \mathbf{B}^{-1} \mathbf{v}_c}{\alpha_2} \quad (14c)$$

$$\mathbf{v}_c = \frac{D \alpha_1 \mathbf{A} \mathbf{C}, \mathbf{B}^{-1} \mathbf{v}_c}{\alpha_2} \quad (14d)$$

$$\alpha = \frac{\alpha_2}{\alpha_1} = \frac{\|D \mathbf{A} \mathbf{C} \mathbf{B}^{-1} \mathbf{v}_c\|}{\|\mathbf{v}_c\|} \quad (14e)$$

Simplifying the right hand side of eq. (14e) (using maxima) the result is:

$$\alpha = \frac{1}{m_{max}^2} \quad (15)$$

Using eq. (14c), the kinematic model becomes:

$$\mathbf{v}_o = \mathbf{M} \mathbf{v}_c; \text{ with } \mathbf{M} = m_{max}^2 \mathbf{A} \mathbf{C} \mathbf{B}^{-1} \quad (16)$$

Which is a linear model if c doesn't change. The reference frame of this model is the object's COM. To get this model in table's reference frame, we rotate the resulting velocity by the current object orientation (ϕ):

$$\mathbf{v}_{o_{table}} = \mathbf{R} \mathbf{M} \mathbf{v}_c; \text{ with } \mathbf{R} = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (17)$$

This model relates the speed of c (in local object coordinates) to the object's COM velocity. It is non-linear and since $\mathbf{R} \mathbf{M}$ is a 3x2 matrix and the system is linearly independent on the control inputs, the system is classified as a "drift-less" or non-holonomic system [18]. Such systems have proven to be difficult to point stabilize (car parking problem).

How can we know if, during pushing, the finger slides? To answer this question we use the friction cone of the finger-object contact. To simplify the analysis, we assume there's no finger movement in the z direction (table normal), then we can define the possible forces that can be applied by the finger as a range between two limit vectors. If the robot moves the finger outside of this range, the finger will slide.

Then, if $F_{c\perp}$ is the perpendicular component (with respect to the lateral contact surface) of the force applied by the finger to the lateral contact surface of the box, then:

$$F_{c\parallel_{max,min}} = \pm \mu_{fo} F_{c\perp} \quad (18a)$$

$$\mathbf{F}_{c_{max,min}} = [F_{c\perp}, \pm \mu_{fo} F_{c\perp}] \quad (18b)$$

Where $\mathbf{F}_{c_{max,min}}$ define the extreme lateral forces that can be applied by the finger to the object. Substituting this equation in eq. (10a) yields:

$$\mathbf{n}_{c_{max,min}} = \begin{bmatrix} 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_y^2 \right) \mp 2 c_x c_y \mu_{fo} \\ -2 c_x c_y \pm 2 \left(\frac{m_{max}^2}{f_{max}^2} + c_x^2 \right) \mu_{fo} \end{bmatrix} F_{c\perp} \quad (19)$$

Using any reasonable value for $F_{c\perp}$ (e.g. 1), one can find the unit limit vectors $\hat{\mathbf{v}}_{c_{max,min}}$. $F_{c\perp}$ has a different value for each limit vector. To find this, one can use eq. (8) with

(18b), resolve for $F_{c\perp}$ and substitute in eq. (19). Using the perpendicular (to the box surface) component of $\mathbf{v}_{finger_{xy}}$, it's possible to scale $\mathbf{n}_{c_{max,min}}$ to get $\mathbf{v}_{c_{max,min}}$.

The system checks that $\mathbf{v}_{finger_{xy}}$ is not outside of the range delimited by $\mathbf{v}_{c_{max,min}}$, in this case $\mathbf{v}_c = \mathbf{v}_{finger}$, and eq. (17) is used. If it's outside, we choose \mathbf{v}_c to be $\mathbf{v}_{c_{max}}$ or $\mathbf{v}_{c_{min}}$, depending on the side $\mathbf{v}_{finger_{xy}}$ is. To obtain the sliding speed \mathbf{v}_{slide} :

$$\mathbf{v}_{finger_{xy}} - \mathbf{v}_c = \mathbf{v}_{slide_{xy}} \quad (20)$$

Eqs. (17), (19) and (20) together are used to predict object and finger sliding behavior.

Looking carefully the eqs. (16), (3) and (2) where the terms f_{max} and m_{max} appear, μ_{ot} and f_n cancel out. This means that the object sliding movement doesn't depend on these two parameters. What these two parameters will determine is: at which force the object will start or stop sliding. For this, we first translate the force \mathbf{F}_c to COM (eq. (6)). Then using eq. (1c) we can deduce if movement will start or stop. When the object is still and we start to increase the applied force, the term on the right side will increase until is near to 1. In this moment, the object could start sliding (we use the static versions of μ_{ot} for calculating f_{max} and m_{max}). If the force is increased even more, the object would accelerate. On the other hand, if the object is moving and we start to decrease the applied force, and the right side becomes smaller than 1, object will stop at any moment (we use the dynamic versions of μ_{ot} for calculating f_{max} and m_{max}). For each case, an appropriate threshold is chosen in practice.

A simulation environment was programed to test the model. A simulated straight pushing motion is shown in Fig. 4.

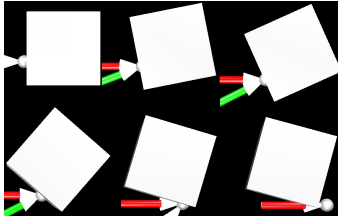


Fig. 4. Sliding sequence simulation. Red: finger speed, green: finger force.

V. EXPLORATION FOR MODEL ADAPTATION

Our robot needs the base shape, the weight, object-table and finger-object friction coefficient parameters to be able to use the model. These parameters can be either recalled from a database, or explored. Our robot explores for all except the base shape (we assume a vision system can give us that). We use the same methods used in our previous work [6], with the additions that, we now also explore for the kinetic friction coefficient and the finger-object friction coefficient. For the μ_k , we carefully look for the initial force spike. For $\mu_{finger-object}$, the robot first presses a lateral side of the object with the finger, and then slides it along this surface. See Fig. 5

Objects used were: an ice tea box, a coffee package, and a cereal box. For each, the experiments were repeated 10 times. The resulting parameters are shown in table I.



Fig. 5. Robot finding the model parameters

	Ice tea	Coffee	Cereal
weight (kg)	0.419	0.510	0.241
$\mu_{s_{object-table}}$	0.298	0.313	0.343
$\mu_{k_{object-table}}$	0.269	0.240	0.292
$\mu_{s_{finger-object}}$	0.693	0.885	0.938
$\mu_{k_{finger-object}}$	0.567	0.733	0.660

TABLE I
OBJECTS PARAMETERS

VI. PREDICTION

Initially, the predicted object position is set to the camera object estimated position. Because the camera position estimates are slower than the finger measurements (15 against 120 samples per second), then the predictor only uses the finger tip measurements (force and position) to estimate object positions while there are no camera estimates available. In this way the predictor it's believing the camera about the global position of the object, but when no camera estimate is available the predictor uses the finger to integrate object positions. We use the force together with the finger velocity and eq. (10) to be able to find out if the finger is slipping or not and how much. Internally the module calculates the velocity direction \mathbf{n}_c using eq. (10) and the current finger forces. The magnitude is calculated by scaling \mathbf{n}_c by a factor α that makes the \mathbf{n}_c component that is normal to the finger-object surface, equal to the same speed of the same component of the finger tip velocity.

$$v_{finger\perp_{surface}} = \alpha n_{c\perp_{surface}} \quad (21a)$$

$$\mathbf{v}_c = \alpha \mathbf{n}_c \quad (21b)$$

Then eqs. (17), (19) and (20) are used to calculate the object velocity and this is then integrated to get the next object position. Finger slipping is calculated with eq. (20). The finger tip velocity is obtained by calculating the derivative of the tip positions.

To decide if the finger is really pushing the object, we use a finger force threshold based on eq. (8). With this equation we can also establish if the object will start moving or stop moving.

Fig. 6 shows in red and green the object predicted pose. The jumps are the new camera pose updates. To observe more easily, how the prediction estimates start to drift away if we don't get global synchronizations often enough, we artificially slowed down the camera update (Figs. 6a, 6b). Real camera update rates results are shown in Figs. 6c and 6d. From the pictures we can notice, that the object predicted positions (that happen after the robot gets a camera pose update) (Fig. 6b), show some drift, but the prediction trajectory shows a proper direction of evolution. Some drift is expected since

the parameter estimation and the model are not perfect, and also because, after the robot gets a camera pose update, the robot starts to integrate the positions, and integrations in practice will always introduce drift. If the camera rate update is increased, the maximum drift is reduced. Another possible reason for the drift comes from the camera pose estimates, if the initial condition for the predictor is not perfect, errors will accumulate during integration. The precision of our camera pose estimate was of 1.5cm.

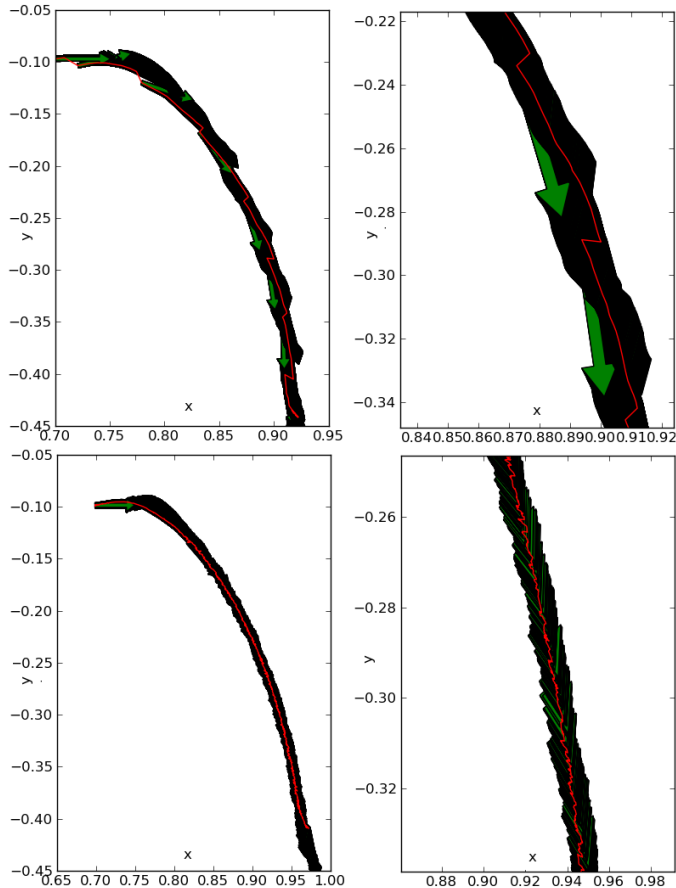


Fig. 6. Prediction. Red: object position, Green: object orientation. (goal is at (0.7,-0.1)). Units=meters a) top-left: simulated very slow camera frame rate, b) top-right: Close-up of a., c) bottom-left: Normal camera frame rate, d) bottom-right: Close-up of c.

We noticed that prediction accuracy is specially sensitive to offset errors in the camera estimated synchronization poses. This is expected since the planar sliding motion is very sensitive around pushing directions passing through the COF. A pushing vector that is passing slightly away of the COF causes a strong turning reaction (this also strongly depends on m_{max}).

VII. CONTROL

This section consists of A) deciding the initial c , and B) pushing the object towards the goal.

A. Initial pushing point

The strategy to find a initial c is based on trying to reduce control effort during pushing. We draw a line from the goal

through the object's COM until it touches the most far away (from the goal) lateral surface. c is the point where this line touches the lateral surface (Fig. 7).

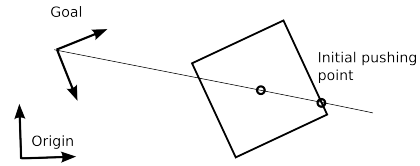


Fig. 7. Initial pushing point

B. Pushing towards the goal

As mentioned before, “drift-less” systems are difficult to control. There's a subset of these systems that can be converted to a so-called chained-form [19], but no general methods exists for this task. In our case we tried to derived a chained-form but no way was found to separate the variables to obtain a clean chained-form system. We opted for a more intuitive design approach to the control rule.

Our particular desire is to slide the object, in one single shot, from any pose to another. Usually, non-holonomic controllers do multiple iterations near the origin to converge, but they don't necessarily make their best effort to get to the goal in the first approach (e.g. [20]). We want to avoid this, because ideally we want to only push and not pull. Our intuition and practice tell us that by pushing the object to a previous position in front of the goal with a near-to-final orientation, then just pushing forward at the end would be enough to get to the goal. This is shown in Fig. 8a. An even better solution (Fig. 8b) consists of selecting a movement direction along an angle that is bigger than the angle formed between the axis x and the vector COM (if the goal is the origin). This angle is called $\phi_a = 1.5 * \eta$. As the object approaches the goal, this angle approaches the goal orientation ([21] for more details). We selected this solution.

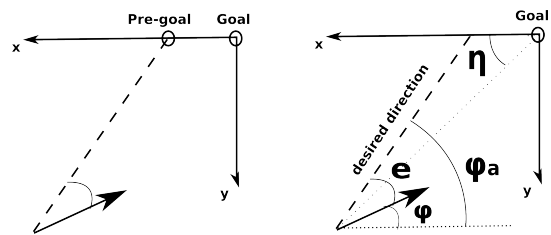


Fig. 8. Left: a) Intuitive control. Right: b) control method from [21]

It can be shown, that if we rotate our model reference frame in a way that the finger pushing point is always in a new local x' axis (Fig. 9), then our local model equations are simplified to:

$$v_o = M v_c, M = \begin{bmatrix} 1 & 0 \\ 0 & t_1 \\ 0 & t_2 \end{bmatrix} \quad (22a)$$

$$t_1 = \frac{1}{1 + \frac{f_{max}^2 c_x^2}{m_{max}^2}}, t_2 = \frac{c_x}{\frac{m_{max}^2}{f_{max}^2} + c_x^2} \quad (22b)$$

This particular configuration is useful, because it allows us to divide the control problem in, one of pushing (v_{for}) through the COM to get straight motion, and another (v_{lat}) one to turn the object (with an undesired but unavoidable additional translational lateral movement)

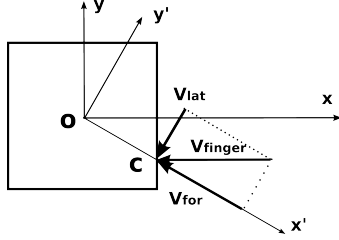


Fig. 9. New reference frame for the controller

Calculating the variables d , ξ (which we refer as e), ϕ , ϕ_a , η (Fig. 8) and using $b = 1.5$ (refer to [21] for a complete explanation of these parameters), we can compute our control values with the following equations:

$$v_{for} = v_{ref}\alpha \quad (23a)$$

$$v_{lat} = r_e + r_\phi \quad (23b)$$

$$r_e = s_1 e v_{for} d_{extra}^2 \quad (23c)$$

$$r_\phi = \frac{s_2 \phi v_{for}}{\exp \frac{t_2 * d}{t_1}} \quad (23d)$$

In our case $s_1 = 830$ and $s_2 = 45$ gave in practice an appropriate robust behavior. The values are high because the control system is working in a similar way as a sliding mode controller. v_{for} , v_{lat} must be rotated to the global reference frame and sent to the finger. d_{extra} is d (distance to goal) saturated to a particular value. α is a modulation value that slows down all the movements when we approach the goal.

Eq. 23a makes the object always go forward to maintain movement, otherwise the finger will detach from the object. v_{ref} is a constant maximum reference speed for the forward movement.

v_{lat} depends on the control signals r_e and r_ϕ . The first one tries to correct the orientation of the object to go aligned to the auxiliary directional angle ϕ_a . This is to prepare the box to be near the final orientation before reaching the goal. Alone, this control signal should be enough, but the system easily behaves incorrectly near the goal: η and therefore ϕ_a angle can change abruptly from small to high values, making the control rule change the orientation too quickly (Fig. 10b).

To avoid this, near the goal, we reduce the effect of r_e with d_{extra}^2 and $\exp \frac{t_2 * d}{t_1}$, and increase the effect of r_ϕ . r_ϕ tries to correct the object orientation to the final desired orientation. Since, near the goal, the finger-object vector should already be pointing towards the goal, this should give a proper behavior. The control rule depends on the objects parameters through t_1 and t_2 . We have to point out that, since our controller has a sliding-mode behavior, it's quiet insensible to parameter errors.

The control correction that r_e provides it's imperative. And the e error must be maintained very near to zero to have a good

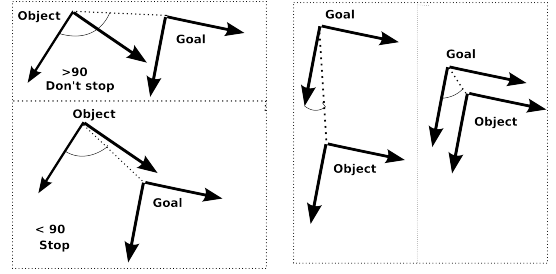


Fig. 10. Situation near goal. a) Left: when to stop movement. b) Right: η unstable near goal

final goal precision. To achieve this, the s_1 constant must be very high. This creates a form of sliding mode control, where the system slides along this condition ($e = 0$) until is near the goal (when r_ϕ becomes more important). Using s_1 very high, also proved to be important for robustness to object parameter errors and, to some degree, to camera object detection offsets.

v_{for} and v_{lat} are further processed to take into account the finger friction cone, and they are saturated to be sure the finger doesn't get disconnected from the object (near the object corners or in any other place). Again we use eq. (10) and (20) and some maximum safe v_{slide} to limit v_{lat} . Dependency in eq. 20, creates a dependency in μ_{fo} which is a explored parameter. The controller is less dependent on μ_{ot} and weight. They are used, together with eq. 8), to know when to start the pushing controller. If the right side value of eq. 8 is near the left side value, control can start, otherwise it could mean that we are not touching the correct object. This part of the predictor is used to trigger the pushing controller.

To stop the controlling action (near the goal), we calculate the angle between the object's x axis and the vector position from the goal, if this angle becomes smaller than 90 degrees, the control stops (Fig. 10a).

Typical control trajectories are shown in Figs. 11a, 11b and 11c. In the first pictures (Ice tea and Coffee), the trajectory is especially smooth. While sliding, these objects rarely got "stuck". On the other hand, the third picture (cereal) shows a more abrupt trajectory: during experimentation, the cereal got "stuck" often (probably because the cereal box was very thin, therefore more prone to oscillate). Since the gains of the controller are very high, the robot is quick to react to irregular movement from the object, and is still able to approach to goal to within 2cm. The controller considers this irregular behavior, perturbations.

The Fig. 11d shows the reaction of the system to perturbations. (Human poking with the ice tea while robot pushes the box towards the goal). Fast reaction and correction, as well as, good goal reaching precision during perturbation experiments were observed consistently.

VIII. DISCUSSION AND FUTURE WORK

As a result of this work we managed to 1) give a robot a predictive model of sliding objects in a plane, using a compact, yet precise, model that requires few parameters, 2) deal with the difficult problem of pushing with only one contact towards a selectable position and orientation robustly,

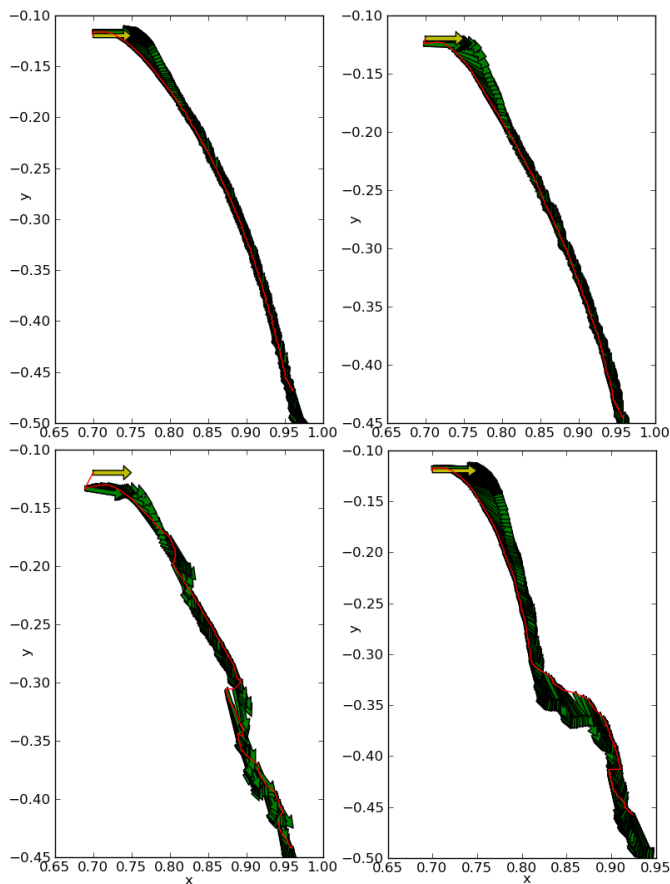


Fig. 11. Feedback controller object trajectories executed on the real robot. Green: object camera pose, yellow: goal=(0.7,-0.12), units in meters. a) Top-left: Ice tea, b) top-right: coffee, c) bottom-left: cereal, d) bottom-right: Ice tea with perturbations

and 3) consider in the prediction and control, finger-object slipping, 4) in a real world scenario, using regular kitchen objects with a humanoid robot. The behavior shown by the robot, demonstrates a skilled level of manipulation, which has the potential to be easily parameterized and commanded by high level instructions. Furthermore, since we use a model-based system, other robots with similar sensing and actuation capabilities, could run the same system even with the same parameters (if the force sensing is calibrated).

Possible future improvements/uses: 1) The finger slipping part of the model can be used in objects that don't have flat lateral surfaces, it only needs to know the normal vector of the lateral surface of the object where the finger is touching. 2) Use the same model even with arbitrary object-table surface shapes. For this, we would try to best-fit a rectangle to the surface and apply the model. 3) A better strategy for selecting the initial PP should take into account the robot's kinematics at the beginning and during the movement. 4) Using feedback optimal control, may allow us to derive a more powerful control law. 5) A control engineering stability analysis must be done on the control rule to identify more clearly the stability properties of the system. 6) We could use the object predicted positions to improve controller performance. 7) Use the prediction errors to estimate the object's COM offset

during control. 8) More in-depth analysis in the prediction capabilities.

Note: For a high resolution demonstration video please visit: <http://flexo.informatik.tu-muenchen.de/pushing.avi>

REFERENCES

- [1] L. Y. Chang, G. J. Zeglin, and N. S. Pollard, "Preparatory object rotation as a human-inspired grasping strategy," in *International Conference on Humanoid Robots*, 2008.
- [2] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.
- [3] L. Mosenlechner, N. Demmel, and M. Beetz, "Becoming Action-aware through Reasoning about Logged Plan Execution Traces," in *Submitted to the IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2010.
- [4] 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.
- [5] 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.
- [6] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, "Prediction of action outcomes using an object model," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22 2010.
- [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] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit surface and moment function descriptions of planar sliding," in *International Conference on Robotics and Automation*, 2009.
- [9] R. D. Howe and M. R. Cutkosky, "Practical force-motion models for sliding manipulation," *International Journal of Robotic Research*, vol. 15, pp. 557–572, 1996.
- [10] M. T. Mason, *Mechanics of Robotic Manipulation*. MIT Press, 2001.
- [11] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1992, pp. 416–421.
- [12] K. Lynch and M. Mason, "Stable pushing: Mechanics, controllability, and planning," *International Journal of Robotics Research*, vol. 15, no. 1, pp. 533–556, December 1996.
- [13] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *International Journal of Robotics Research*, vol. 18, pp. 64–92, 1998.
- [14] J. Bernheisel and K. Lynch, "Stable transport of assemblies by pushing," *Robotics, IEEE Transactions on*, vol. 22, no. 4, pp. 740–750, aug. 2006.
- [15] V. K. Peng Cheng, Jonathan Fink, "Abstractions and algorithms for cooperative multiple robot planar manipulation," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, 2008.
- [16] H. Alexander and H. Lakhani, "Robotic control of sliding object motion and orientation," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 4, apr 1996.
- [17] M. Beetz, F. Stulp, P. Esden-Tempski, A. Fedrizzi, U. Klank, I. Kresse, A. Maldonado, and F. Ruiz, "Generality and legibility in mobile manipulation," *Autonomous Robots Journal (Special Issue on Mobile Manipulation)*, vol. 28, no. 1, pp. 21–44, 2010.
- [18] A. Isidori, *Nonlinear Control Systems*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1995.
- [19] F. Matsuno and J. Tsurusaki, "Chained form transformation algorithm for a class of 3-states and 2-inputs nonholonomic systems and attitude control of a space robot," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, 1999, pp. 2126–2131 vol.3.
- [20] A. D. Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot."
- [21] F. Pourboghrat, "Exponential stabilization of nonholonomic mobile robots," *Computers and Electrical Engineering*, vol. 28, no. 5, 2002.