Online Adaptation of Robot Pushing Control to Object Properties

Senka Krivic and Justus Piater

***Abstract*— Pushing is a common task in robotic scenarios. In real-world environments, robots need to manipulate various unknown objects without previous experience. We propose a data-driven approach for learning local inverse models of robot-object interaction for push manipulation. The robot makes observations of the object behaviour on the fly and adapts its movement direction. The proposed model is probabilistic, and we update it using maximum a posteriori (MAP) estimation. We test our method by pushing objects with a holonomic mobile robot base. Validation of results over a diverse object set demonstrates a high degree of robustness and a high success rate in pushing objects towards a fixed target and along a path compared to previous methods. Moreover, based on learned inverse models, the robot can learn object properties and distinguish between different object behaviours when they are pushed from different sides.**

Fig. 1. A Robotino robot tries to push a can since the hand is occupied. Objects in real-world environments, such as a kindergarten room, are diverse making the pushing a challenging task.

I. Introduction

The pushing skill for a robot base can be of great utility in many real-world environments. A robot can deliver objects, or move them out of the way. Also, pushing motions can be used to inspect object properties, or as preparation for some other manipulation. Many researchers have studied nonprehensile manipulation by pushing with robotic manipulators or the robot base. This topic raised interest in different research areas which resulted in a variety of approaches and studies related to push-manipulation [1], [2], [3], [4].

In real-world environments, a robot has to manipulate various objects. Physical models of the environment and objects are usually not available to the robot. In complex robotic scenarios, such as tidying up a kindergarten room, there is a large number of objects (Fig. 1). Different object properties and behaviours manifest for different dynamics of pushing and environment conditions. For example, objects can rotate, slide or slip. Moreover, mass and friction distributions of objects are usually non-uniform. Thus, a robot should be able to adapt to different robot-object-environment dynamics. In this paper, we address the problem of pushing objects with a robot base when properties of the object and the environment are unknown and uncertain.

Physical models of object behaviour under pushing usually require identification of the object and the environment properties, such as friction and mass distributions. These properties can be unobservable directly and exhaustive experiments can be necessary to estimate them [5]. Also, designing an analytical model for push control can be very difficult [1]. Thus, we propose a data-driven approach for learning the inverse model of object-robot push-interaction online.

The authors are with the Department of Computer Science, Universität Innsbruck, 6020 Innsbruck, Austria.
{senka.krivic, justus.piater}@uibk.ac.at

Inverse models can calculate necessary feedforward motor commands from the desired trajectory information. Kawato et al. [6] studied models of human motor learning. They proposed that the brain acquires an inverse dynamics model of the object to be controlled through motor learning, after which motor control can be executed in a pure feedforward manner. Similarly, we propose an adaptive scheme which controls the object movement direction based on the observations. The inverse model produces the reference movement for the holonomic robot to achieve the desired motion of the object towards the target.

The proposed pushing controller produces input for a robot based on current poses of the robot, the object and the target. We define the inverse model of a robot-object push interaction by blending two motions, pushing and relocating around the object. We let the robot learn activation functions for pushing at a specific contact location based on the evidence of successful pushes at nearby contact locations. In this way, local models are assigned locally to particular object orientations, characterising different behaviours of a single object. Nearby contact locations can produce similar behaviours. Therefore, to predict the direction from which an object can be successfully pushed for a specific contact location, the model takes into account observations from nearby contact locations.

Data-driven approaches were part of several studies, which mostly focus on learning accurate models from exhaustive experience based on visual features [7], [8]. In contrast to that, we develop a model that captures local object behaviours under pushing. The acquired model is not an accurate and complete representation of the robot-object dynamics, but it is an estimate that enables successful pushes. In this way, the robot movement adapts based on a small amount

of observed data. As soon as there are some changes in the robot-object-environment interaction, the system adjusts itself.

Our contributions are as follows:

- We propose an inverse model of a push-interaction which captures local behaviours of the object.
- We also introduce an adaptive scheme for learning the inverse model on the fly, which enables pushing unknown objects without previous experience.
- The proposed approach enables spatial classification of object properties based on the object motion data gathered while pushing.

II. Related Work

Many researchers have studied push manipulation over the recent decades. This topic raised interest in different research areas which resulted in a variety of approaches and problem statements related to pushing. First attempts to solve the problem of pushing led to analysis of the dynamics of pushing. Mason and Lynch [2] presented an analytical model of the dynamics of pushing with a robotic manipulator. However, in practice, analytical models are very sensitive to changes of parameters that describe the geometry or friction. Salganicoff et al. [3] created one of the first systems that built a forward model of a planar object from acquired vision data.

Pushing was often used for learning object specific properties, dynamic models and behaviours by observing the results of a push [9], [10]. Several methods treated the problem of learning contact locations for successful pushes such as presented by Hermans et al. [8] or Kopicki et al. [7]. Most of the data-driven approaches focused on extracting the shape of an object and building a direct model for prediction of the object movement based on the contact point. Lau et al. [11] created mappings between pushing actions and their effects using a non-parametric regression method. Similarly, Walker et al. [12] modelled a map of the support friction using local regression. Agrawal et al. [13] proposed learning physics models with deep neural networks from experience for vision-based control. Meriçli et al. [14] created inverse models for pushing massive objects on caster wheels based on experience from self-exploration or demonstration via joy-sticking. In contrast to these methods, we do not take any shape features of the object and previously acquired experience into account. We estimate the model of push-interaction on the fly by observing the pose of the object.

Igarashi et al. [15] presented a dipole-field pushing controller that generates object motion by two primitive robot motions: pushing the object, and orbiting around it. Similarly, we introduced an adaptive feedback controller [16], as well based on the displacement angle of the robot from the desired object movement direction. We extend this work by defining an adaptive inverse model of local object behaviours while taking robot-object configurations into account.

III. Problem Formulation

Push manipulation is part of a modular complex system that enables operation of a mobile robot in real-world scenarios. The robot's task is to push an object towards a fixed or movable target along a path. We assume that the global pose $\mathbf{r} = \begin{bmatrix} x_\mathbf{r} & y_\mathbf{r} & \theta_\mathbf{r} \end{bmatrix}^T$ of the robot and the global pose $\mathbf{o} = \begin{bmatrix} x_\mathbf{o} & y_\mathbf{o} & \theta_\mathbf{o} \end{bmatrix}^T$ of an object are available at each time step k (Fig. 2).

The robot needs to push unknown objects in different environments without previous experience. Let us assume that the dynamics of the object motion can be described with a nonlinear equation $\dot{\mathbf{o}} = f_\mathbf{o}(\mathbf{o}, \mathbf{r}, \dot{\mathbf{r}})$.

The model of the object motion $f_\mathbf{o}$ and its inverse are unknown to the robot, but it is assumed that they can be described by a probabilistic model. We formulate the problem as estimating an inverse model $\dot{\mathbf{r}} = \hat{f}_\mathbf{o}^{-1}(\mathbf{o}, \mathbf{r}, \dot{\mathbf{o}}^d)$ of the robot-object pushing interaction on the fly by observing the motion of the robot and the object. $\dot{\mathbf{o}}^d$ denotes a desired object movement defined by the current object pose $\mathbf{o}$ and the current target pose $\mathbf{t}$. The adaptive model is used to produce a reference signal for the robot.

IV. Learning Inverse Models of Push-Interactions

In this section, we describe the proposed approach which enables online adaptation of robot movement to object pushing behaviours. First, we introduce a motion controller that enables adaptive pushing based on inverse models of push-interaction (IV-A). Objects often have irregular shapes, mass and friction distributions. Different orientations of the object entail different contact locations while pushing, and can cause different object behaviours. Thus, the object orientation

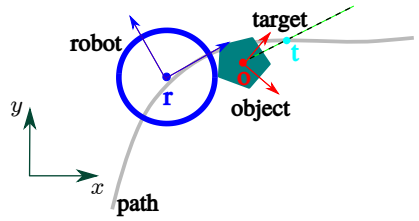

Fig. 2. The robot's task it to push an object towards a fixed target or a movable target on a path.

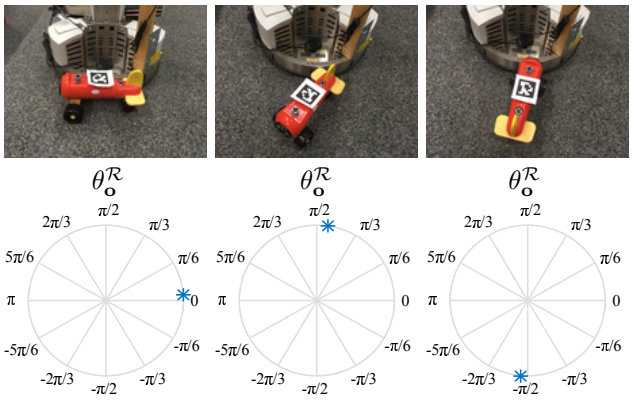

Fig. 3. Different configurations of the robot and the object while pushing. Blue marks denote the corresponding orientations of the object in the robot frame $\mathcal{R}$.

$\theta^{\mathcal{R}}_{\mathbf{o}}$ should be taken into account. $\mathcal{R}$ denotes a local frame assigned to the robot's current pose $\mathbf{r}$. Examples of various configurations of an object are shown in Fig. 3.

Models (Sec. IV-B) are assigned locally to different contact locations. Then, we describe online learning of local models (Sec. IV-C). For each contact location, the behaviour of the object when pushed at nearby contact locations is used to update a local model. At the end of this section (IV-D), we describe how nearby contact locations are chosen.

A. Learning scheme

To be able to push an object in the desired direction, we propose an adaptive movement scheme as shown in Fig. 4, which produces the control input $\mathbf{u}$ for the robot. It consists of an adaptive feedforward component for learning the inverse model of the robot-object push interaction $\hat{f}^{-1}_{\mathbf{o}}$ which produces the reference signal for the robot, and the feedback component altering it.

The proposed scheme controls the object movement direction $\theta_{\dot{\mathbf{o}}}$, while the velocity of the robot movement is kept constant. The error of the object movement direction is

$$\gamma = \theta^{d}_{\dot{\mathbf{o}}} - \theta_{\dot{\mathbf{o}}}$$

where $\theta^{d}_{\dot{\mathbf{o}}}$ is the desired direction of the object movement $\overrightarrow{\mathbf{ot}}$, as shown in Fig. 5**(a)**.

The robot control signal $\mathbf{u} = \begin{bmatrix} r_{\mathbf{u}} & \theta_{\mathbf{u}} \end{bmatrix}^T$ is defined in polar coordinates as

$$\begin{aligned} r_{\mathbf{u}} &= V \\ \theta_{\mathbf{u}} &= \theta_{\text{feedforward}} + \theta_{\text{feedback}} \end{aligned} \tag{1}$$

where V is the constant magnitude of the desired robot velocity, $\theta_{\text{feedforward}}$ is the movement direction defined by the inverse model $\hat{f}^{-1}_{\mathbf{o}}$, and θ_{feedback} is the output of a proportional-integral (PI) controller with saturated I component, also known as integral anti-windup.

At the beginning of the push, the robot has no knowledge about the object. The feedforward component will produce a reference signal corresponding to the pushing of a circular object with uniform mass and friction distribution. The feedback term reacts to the current error of the object movement direction, and provides a slow but reliable control input for the system as the adaptive feedforward component learns the inverse model of the current robot-object interaction.

B. Inverse Model of Push-Interaction

To simplify the description, let us assign a local pushing frame $\mathcal{P}$ to the robot $\mathbf{r}$ such that the $x^{\mathcal{P}}$ axis coincides with $\overrightarrow{\mathbf{ro}}$. The success of a push depends on the robot-object-target configuration, which is characterized by the angle α, defined as in Fig. 5**(b)**, and the current object orientation $\theta^{\mathcal{R}}_{\mathbf{o}}$. Thus, the inverse model of push-interaction determines the robot movement direction based on the current α and $\theta^{\mathcal{R}}_{\mathbf{o}}$.

While interacting with an object, the robot has to push the object towards the target but also relocate to a position from where pushing is possible. To introduce the inverse model, we define two primitive movements, pushing and relocating. The direction of pushing $\hat{\mathbf{d}}_{\text{push}}$ equals $\overrightarrow{\mathbf{ro}}$, and the primitive relocate direction $\hat{\mathbf{d}}_{\text{push}}$ is perpendicular to it. We define the inverse model of robot-object pushing interaction by blending two primitive movements in the feedforward controller component, as shown in Fig. 5**(c)**. The output of the inverse model defines the desired direction of robot movement and is given by

$$\mathbf{u}^{\mathcal{P}}_{\text{feedforward}} = \psi_{\text{push}}(\alpha, \theta^{\mathcal{R}}_{\mathbf{o}})\hat{\mathbf{d}}^{\mathcal{P}}_{\text{push}} + \psi_{\text{relocate}}(\alpha, \theta^{\mathcal{R}}_{\mathbf{o}})\hat{\mathbf{d}}^{\mathcal{P}}_{\text{relocate}} \tag{2}$$

where ψ_{push} and ψ_{relocate} are activation functions which system adapts based on the observed object movement. This results in smooth transitions between pushing and relocating.

Push and relocate directions are defined in the local pushing frame as

$$\hat{\mathbf{d}}^{\mathcal{P}}_{\text{push}} = \text{sgn}(\cos\alpha)\hat{\boldsymbol{\imath}}^{\mathcal{P}} \tag{3}$$

$$\hat{\mathbf{d}}^{\mathcal{P}}_{\text{relocate}} = \text{sgn}(\sin(\alpha_p(\theta^{\mathcal{R}}_{\mathbf{o}}) - \alpha))\hat{\boldsymbol{\jmath}}^{\mathcal{P}} \tag{4}$$

where $\hat{\boldsymbol{\imath}}^{\mathcal{P}}$ and $\hat{\boldsymbol{\jmath}}^{\mathcal{P}}$ are unit vectors in the direction of the $x^{\mathcal{P}}$ and $y^{\mathcal{P}}$ axes respectively, and $\alpha_p(\theta^{\mathcal{R}}_{\mathbf{o}})$ is the value of α that enables pushing of the object directly towards the target for the current object orientation by moving the robot only into the direction $\hat{\mathbf{d}}^{\mathcal{P}}_{\text{push}}$. This value depends on the robot-object contact location which is characterized by the object orientation $\theta^{\mathcal{R}}_{\mathbf{o}}$. For a circular object with uniform mass and friction distribution this value is $\alpha_p = 0$. The system estimates values of $\alpha_p(\theta^{\mathcal{R}}_{\mathbf{o}})$ for a particular object on the fly. Non-zero values of α_p indicate misalignment of the center of mass from the center of geometry.

The sign function in Eq. (3) enables the robot relocation to the position from where it is possible to push in the case that $|\alpha| > \pi/2$. Blending the two movement directions drives the robot around the object such that $\alpha \to \alpha_p$.

We let the robot learn which angles α are good for pushing and estimate the push activation function ψ_{push}. Since primitive movements are perpendicular, we define the relocation activation function as

$$\psi_{\text{relocate}}(\alpha, \theta^{\mathcal{R}}_{\mathbf{o}}) = \sqrt{1 - \psi^2_{\text{push}}(\alpha, \theta^{\mathcal{R}}_{\mathbf{o}})}. \tag{5}$$

C. Online Adaptation of Inverse Models

To estimate the push activation function ψ_{push}, we let the robot learn which angles α are suitable for pushing. We characterize them as angles which lead to a decrease of the error of the object movement $|\dot{\gamma}| < 0$. We assume these angles are distributed according to a *von Mises* distribution

$$p(\alpha|\mu, \kappa) = \frac{\mathrm{e}^{\kappa\cos(\alpha-\mu)}}{2\pi I_0(\kappa)}, \tag{6}$$

a circular analogue of the normal distribution, where $-\pi < \mu \leq \pi$, $0 \leq \kappa \leq \infty$, and $I_0(\kappa)$ is the modified Bessel function of order zero. The mean direction of the distribution is given by μ, and κ is a concentration parameter with $\kappa = 0$ corresponding to a circular uniform distribution and $\kappa \to \infty$ to a point distribution.

Given evidence of angles $\mathcal{D}_\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ that resulted in $|\dot{\gamma}| < 0$ coupled with evidence of corresponding object orientations $\mathcal{D}_\theta = \{\theta^{\mathcal{R}}_{\mathbf{o}\,1}, \theta^{\mathcal{R}}_{\mathbf{o}\,2}, ..., \theta^{\mathcal{R}}_{\mathbf{o}\,n}\}$, we estimate

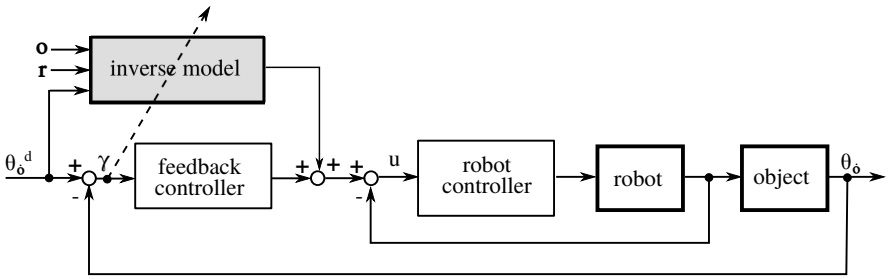

Fig. 4. The proposed control scheme for learning inverse models of the object-robot pushing interaction. The feedforward component, the inverse model, and the feedback component determine the robot movement direction $\theta_{\mathbf{u}}$.

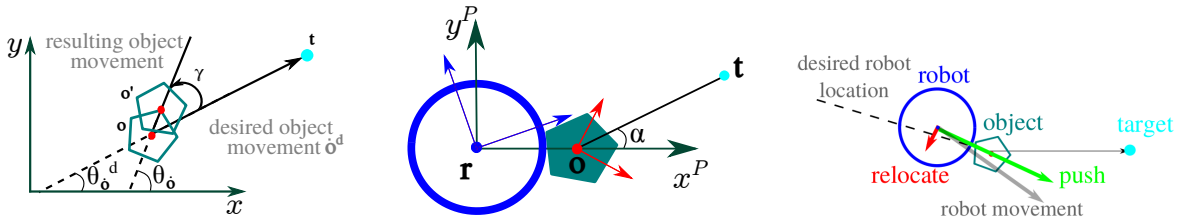

Fig. 5. **(a)** The error of the object movement γ. **(b)** The local push frame is determined by the current robot-object position. The current robot-object-target configuration is characterized by the angle α. **(c)** The robot movement direction is determined by blending push and relocate vectors.

the model parameters μ and κ using maximum a posteriori (MAP) estimation. The posterior distribution $p(\mu, \kappa|\mathcal{D}_\alpha^{\hat{\delta}})$ is obtained by combining the prior knowledge about the distribution $p(\alpha)$ and the knowledge about observed data $\mathcal{D}_\alpha^{\hat{\delta}}$ summarized by the likelihood function $p(\mathcal{D}_\alpha^{\hat{\delta}}|\mu, \kappa)$, where $\mathcal{D}_\alpha^{\hat{\delta}} \subset \mathcal{D}_\alpha$ represents the set of samples of α that correspond to orientations $\theta_{\mathbf{o}}^{\mathcal{R}}$ around the current one within the neighbourhood defined by $\hat{\delta}$ as shown in Fig. 6.

We can estimate the posterior using Bayes' theorem

$$p(\mu, \kappa|\mathcal{D}_\alpha^{\hat{\delta}}) = \frac{p(\mathcal{D}_\alpha^{\hat{\delta}}|\mu, \kappa)p(\mu, \kappa)}{p(\mathcal{D}_\alpha^{\hat{\delta}})}. \tag{7}$$

This is done using a bivariate conjugate prior for the von Mises distribution with unknown concentration and direction, originally developed by Guttorp and Lockhart [17] and revised by Fink [18]. The conjugate prior for the von Mises distribution is defined with hyperparameters R_0, c and ϕ as

$$p(\mu, \kappa|R_0, c, \phi) = \frac{1}{K} \frac{\mathrm{e}^{R_0 \kappa \cos(\phi - \mu)}}{2\pi I_0(\kappa)^c}$$

where $K = \int_0^\infty \frac{I_0(R_0\kappa)}{I_0(\kappa)^c} \mathrm{d}\kappa$ is a normalizing constant.

The prior hyperparameters are set to $R_0 = 5.84, c = 7.0$ and $\phi = 0.0$. We fitted these values such that they correspond to an object of circular shape and uniform mass and friction distributions.

With a model that describes which angles α result in $|\dot{\gamma}| < 0$ for a particular object orientation $\theta_{\mathbf{o}}^{\mathcal{R}}$, we define the push activation function as

$$\psi_{\text{push}}(\alpha, \theta_{\mathbf{o}}^{\mathcal{R}}) \propto p(\alpha|\hat{\mu}, \hat{\kappa}) \tag{8}$$

where $\hat{\mu}, \hat{\kappa}$ are MAP estimates of the model (6). This is a parametrization that maximizes the posterior probability (7) for the observed data:

$$(\hat{\mu}, \hat{\kappa}) = \underset{\hat{\mu}, \hat{\kappa}}{\operatorname{argmax}}\, p(\hat{\mu}, \hat{\kappa}|\mathcal{D}_\alpha^{\hat{\delta}})$$

Parameters $\hat{\mu}, \hat{\kappa}$ are mode values of the posterior distributions obtained in (7).

With the estimated distribution (6), which describes angles α suitable for pushing for the current $\theta_{\mathbf{o}}^{\mathcal{R}}$, we can determine $\alpha_p(\theta_{\mathbf{o}}^{\mathcal{R}}) = \hat{\mu}$ as its expected value.

D. Determination of the Contact Location Neighbourhood

With the assumption that pushing at nearby contact locations will result in similar pushing behaviour, we select samples of $\alpha \in \mathcal{D}_\alpha^\delta$ which correspond to the samples of $\theta_{\mathbf{o}}^{\mathcal{R}} \in \mathcal{D}_\theta^\delta$ such that

$$\theta_{\mathbf{o}\,k}^{\mathcal{R}} - \delta < \theta < \theta_{\mathbf{o}\,k}^{\mathcal{R}} + \delta, \theta \in \mathcal{D}_\theta,$$

where $\theta_{\mathbf{o}\,k}^{\mathcal{R}}$ is the object orientation at time step k.

To find the probability distribution which best represents the activation function ψ_{push}, we choose the set of samples that maximizes the certainty of the model. Therefore, the set of samples $\mathcal{D}_\alpha^{\hat{\delta}}$ is chosen such that it minimizes the entropy of $\mathcal{D}_\alpha^{\hat{\delta}} \subset \mathcal{D}_\alpha$ for the model (6), and is defined as

$$\hat{\delta} = \arg \min_{\delta \in 0,2\pi} H(\mathcal{D}_\alpha^\delta) \tag{9}$$

where $H(\mathcal{D}_\alpha^\delta) = \ln(2\pi I_0(\kappa)) - \kappa \frac{I_1(\kappa)}{I_0(\kappa)}$ is the information entropy of the angles $\alpha \in \mathcal{D}_\alpha^\delta$. In this way, only configurations of the object which are similar to the current one are taken into account. An example of a chosen subset is shown in Fig. 6.

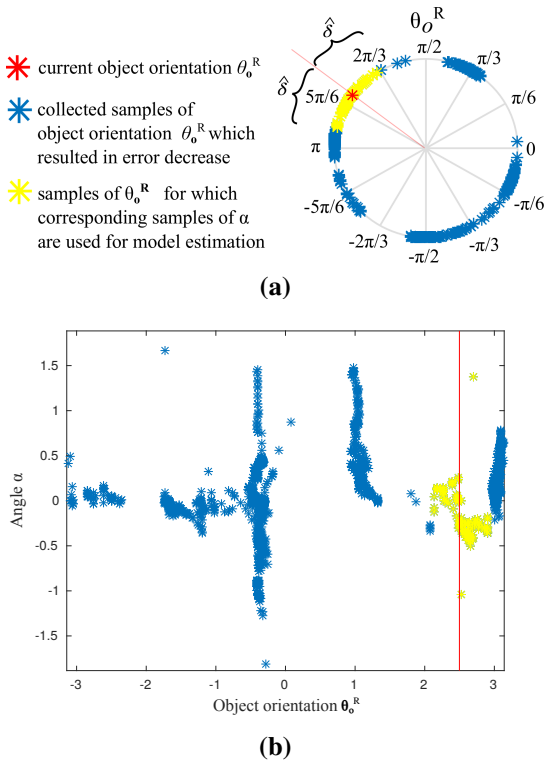

Fig. 6. **(a)** An example of samples which resulted in $|\dot{\gamma}| < 0$ collected for the object *'plane'* in pushing experiments. We illustrate the subset of samples (yellow) that result in a pushing behaviour similar to the current object orientation (red). $\hat{\delta}$ denotes the neighbourhood width. **(b)** Distribution of collected samples of α corresponding to the collected samples of $\theta_o^{\mathcal{R}}$.

V. Experimental Evaluation

A. Experimental Setup

We used a holonomic robot with a circular base, Robotino[1] by Festo, to evaluate the proposed approach. We use the additional degree of freedom to control the robot's orientation for object tracking since the camera is placed in a fixed position on the robot. AR tags were used for more accurate object tracking. For this purpose, we used the ARToolKitPlus[2] software library for the Robot Operating System (ROS). At each time step, the pose of the object and the robot were collected using odometry and the tracking system. The parameters of the PI controller were set to $K_P = 0.1$, $K_I = 0.05$ and integral term limits to $[-0.1, 0.1]$. The desired robot velocity was set to $V = 0.1\mathrm{m/s}$ in all experiments.

We examine the success of pushing various objects and adaptation to different object behaviours. For this purpose, we performed tests of pushing to a fixed target, as well as pushing along a path. We implemented the strategy of path tracking as pushing with a fixed lookahead distance. At each time step k a new pushing target on the path is chosen at a distance of $0.1\mathrm{m}$ from the point on the path that is the closest to the current object position. To test the effectiveness of the method for various object behaviours, we chose the set of objects shown in Fig. 7. The set contains objects of various shapes, sizes, friction properties, and mass distributions. However, we did not include objects that roll because of the pose tracking difficulties.

B. Effectiveness

In the first set of experiments, we demonstrate the superior capabilities of our proposed controller by comparing its performance with two other controllers. For this comparison, we designed two paths for pushing: a straight-line path of length 2.5 m, and a sine-wave path defined as $y = 0.3\sin(\pi x)$.

We compare our controller with the *dipole-field* controller by Igarashi [15]. The robot command for this controller is defined by $\mathbf{u}^{\mathcal{P}} = \left[\begin{smallmatrix}\cos\alpha\\ -\sin\alpha\end{smallmatrix}\right] V$ where α is defined the same as for our controller, as in Fig. 5(b), and V is the velocity amplitude. This controller was designed to push a circular object with a circular robot. Without any observations of the object error, using only the prior, our reference model (2) will produce values identical to this controller.

We also compare our controller with the *centroid* alignment controller proposed by Hermans et al.[19]. This is a proportional controller with two components, correcting the robot displacement error and the object position error. The robot command is defined as $\mathbf{u} = K_g e_g + K_c e_c$ where e_g is the distance of the object to the target, and e_c is the robot displacement from the push line. $K_g = 0.2$ and $K_c = 0.4$ were used in all experiments, with a velocity amplitude saturation of $0.2\,\mathrm{m/s}$. This controller does not take the object orientation into account.

We performed three pushes with each method for all objects in the used object set. The starting object orientation is randomly chosen in each test. We define a push to be successful if the object reached the end of the path to within $0.05\,\mathrm{m}$. A test is considered a failure if an object deviates more than $0.4\,\mathrm{m}$ from the path.

Examples of pushing paths and executed trajectories of the robot and the object are shown in Fig. 8. The success rate is summarized in Tab. I, where the proposed method is labelled as *adaptive*. Examples from this experiment, are also given in the the accompanying video.

Another useful metric to measure the performance is the tracking accuracy. We define the tracking error at each time step k as the shortest Euclidean distance of the object to the path. In Fig. 9, we compare box plots of errors over all successful pushes. Note that the set of samples for the adaptive method is much larger than for the others due to larger proportion of successful pushes. For the proposed method we also compare errors for each object in Fig. 10.

It can be seen that the proposed method demonstrates a high success rate (Tab. I) as well as a higher tracking accuracy (Fig. 9), compared to the other two methods. Objects 'plane' and 'cloud' appear to be harder to push compared to other objects as can be seen in Tab. I. The object 'plane' manifests more complex behaviours, while the object 'cloud' is significantly larger than the others, making it easier to reach the error limit of $0.4\mathrm{m}$. The sine-wave

[1] http://www.festo-didactic.com/int-en/services/robotino
[2] http://wiki.ros.org/v4r_artoolkitplus

‘dino’ ‘plane’ ‘cloud’ ‘drill’ ‘frenchs’

Fig. 7. The set of objects used in the experiments.

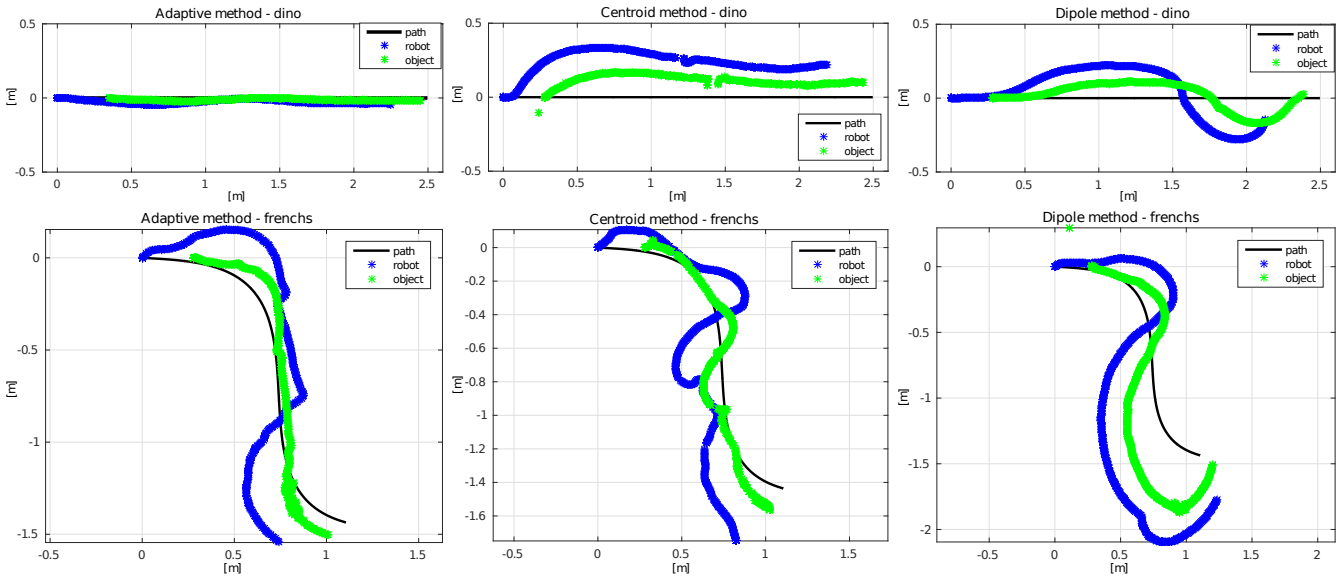

Fig. 8. Examples of pushes in path tracking tests.

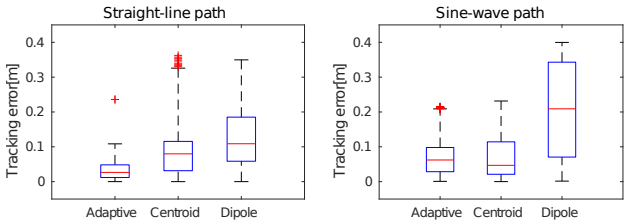

Fig. 9. Comparison of trajectory tracking errors calculated at each time step k over all successful pushes.

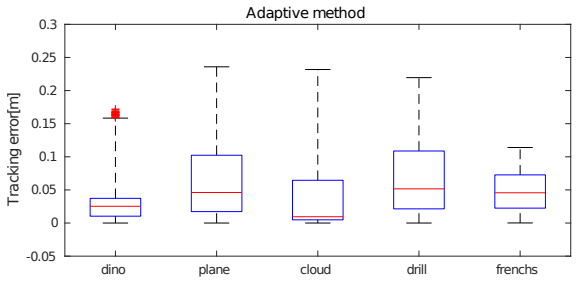

Fig. 10. Comparison of trajectory tracking errors over all successful pushes for each object using the proposed adaptive method.

TABLE I

COMPARISON OF PUSHING SUCCESS FOR EACH OBJECT OUT OF 3 TRIALS

Controller	*Adaptive*	*Centroid*	*Dipole*
Path: Line			
‘dino’	3	2	2
‘plane’	2	1	0
‘cloud’	3	1	1
‘drill’	3	1	1
‘frenchs’	3	2	2
Path: Sine			
‘dino’	3	1	1
‘plane’	2	1	0
‘cloud’	2	0	0
‘drill’	2	1	1
‘frenchs’	3	2	1

pushing path is more difficult for the *Centroid* controller because of the changes in the desired object displacement directions. Looking at the success rates in Tab. I and box plots in Fig. 9, we can see that *Dipole* controller performs the poorest. This is a consequence of the fact that this controller is designed in a pure feedforward manner. The single failure for pushing the object ‘drill’ with the *Adaptive* method was due to the start pose, where the drill chuck was the first contact location. This was caused by a lack of evidence

about object behaviour since the push failed already at the beginning of the path. Experimental tests over the object set shows high robustness of the proposed method. However, the prior which corresponds to a circular object can be misleading in cases such as the robot starting to push at the drill chuck. For this reason, the controller will not be suitable for objects that differ from the prior extremely in shape and mass distribution, e.g. a long, thin stick.

C. Adaptation

In a second set of experiments, we examine the adaptation effects and the benefits of using local models. Here, we compare two strategies: the proposed one, where the object orientation is taken into account by choosing subsets $\mathcal{D}^{\delta}$, and one where a single model (7) is estimated over the whole set $\mathcal{D}$. We let the robot push an object towards a fixed target at an initial distance of $2.5\,\mathrm{m}$ (Fig. 11). For each strategy, we performed five pushes.

To evaluate these results we calculated the pushing effort as the arc length $L_{\mathbf{o}} = \sum_{i=1}^{N} |\mathbf{o}(t_i) - \mathbf{o}(t_{i-1})|$ of the object trajectory. We compare the pushing effort for each object in Fig. 12. The proposed method demonstrated better pushing performance than the approach where the likelihood was calculated from the whole set $\mathcal{D}$. It is evident that the proposed controller with local models gives better results in the case of objects with more complex properties such as

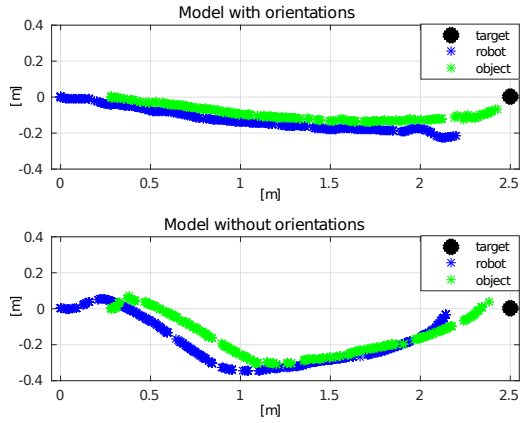

Fig. 11. Examples of trajectories in tests of pushing towards a fixed target.

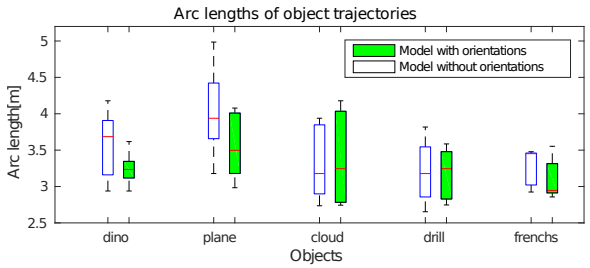

Fig. 12. Arc lengths of object trajectories are compared for the proposed inverse model of the robot-object pushing interactions taking into account orientations of the object and the inverse model ignoring the orientation of the object.

'plane', 'dino' and 'drill'. Contrary to that, in the case of the 'cloud' object with its fairly uniform mass and friction distribution, the performance is slightly worse than for the controller that uses a single model.

D. Control effort

We demonstrate an example of control signals produced by feedforward and feedback components together with object and robot trajectories in Fig. 13. Together, they define the robot's control signal $\theta_{\mathbf{u}}$ (1), As previously stated, at the beginning of a push the object is unknown, the feedforward component produces control signals suitable for a circular object, and the feedback component intervenes to correct the error. As the feedforward component adapts, it can be seen that the effect of the feedback component is reduced and the feedforward component takes over (e.g. at 5–15s).

E. Object properties

To shed further light on the effectiveness of the proposed controller for each object (Figs. 10 and 12), we examined how informative the local models obtained in the experiments were. For this purpose, we analysed offline inverse models with the complete evidence collected during all experiments. We clustered orientations based on subsets D^{δ} and their overlapping areas. This enables us to create object property maps consisting of clustered contact orientations as shown in Fig. 14. For the object 'cloud', it appears that the model was estimated from all or most of the collected samples. This can be justified by the nature of the object since it is lightweight with fairly uniform properties. It appears that the number of clusters indicates the complexity of the mass and friction distributions of an object, or its shape, such as for 'plane'.

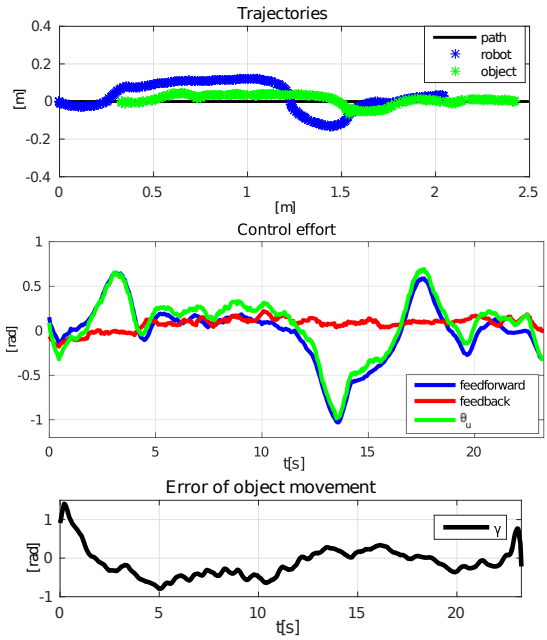

Fig. 13. Example of pushing object 'dino' along a straight-line path.

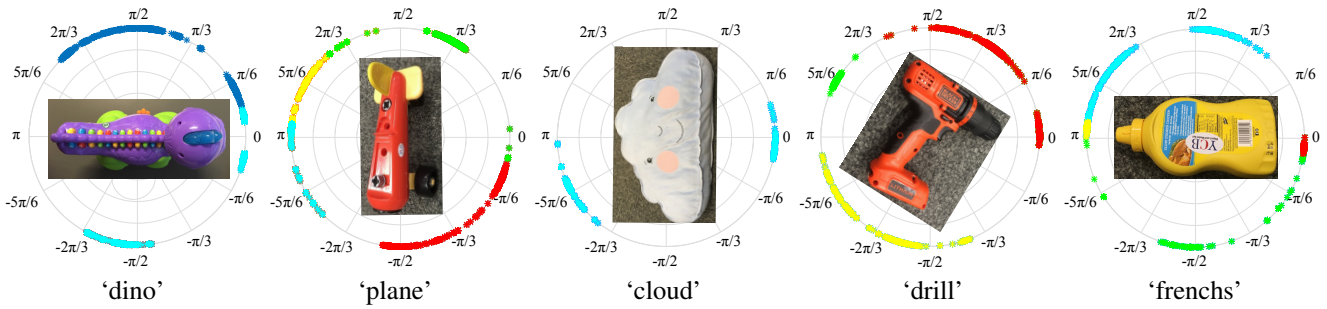

Fig. 14. Clustered object peripheral points based on the common subsets D^{δ}. Different colours belong to different observed object orientations for which the robot was able to push the object towards the target.

VI. Conclusion and Future Work

We presented a data-driven approach for online learning of inverse models of robot-object interactions for pushing unknown objects. The model is learned based on observed robot and object movements. We demonstrated the high success rate and tracking accuracy of the proposed controller, providing comparisons with other controllers in the literature. Also, results demonstrate that the robot can acquire knowledge of object behaviours when pushed at different contact points.

While performing exhaustive experiments, we did not observe any behaviour that lead to instability. A rigorous theoretical study of stability of the proposed control loop is part of planned future work.

Our method does not rely on any visual features of an object, but only on the observed motion of it. Thus, it would be interesting to combine our results with vision-based approaches for pushing, such as that by Hermans et al. [8], or with approaches for learning models of push affordances, e.g. Ridge et al. [20]. Also, the future will involve analysis of object rotations.

Acknowledgement

The research leading to these results has received funding from the European Community’s Seventh Framework Programme FP7/2007-2013 under grant agreement no. 610532, SQUIRREL, and a scholarship granted to S.K. by the University of Innsbruck, Vice Rectorate for Research.

References

[1] M. T. Mason, “Mechanics and planning of manipulator pushing operations,” *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986. [Online]. Available: http://ijr.sagepub.com/content/5/3/53.abstract

[2] 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. [Online]. Available: http://ijr.sagepub.com/content/15/6/533.abstract

[3] M. Salganicoff, G. Metta, A. Oddera, G. Sandini, M. Salganico, G. Metta, A. Oddera, and G. Sandini, “A vision-based learning method for pushing manipulation,” in *In AAAI Fall Symposium Series: Machine Learning in Vision: What Why and*, 1993.

[4] Q. Li and S. Payandeh, “Manipulation of convex objects via two-agent point-contact push,” *Int. J. Rob. Res.*, vol. 26, no. 4, pp. 377–403, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1177/0278364907076819

[5] K. T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, “More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 30–37.

[6] M. Kawato, “Internal models for motor control and trajectory planning,” *Current opinion in neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[7] M. Kopicki, S. Zurek, R. Stolkin, T. Morwald, and J. Wyatt, “Learning to predict how rigid objects behave under simple manipulation,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 5722–5729.

[8] T. Hermans, F. Li, J. Rehg, and A. Bobick, “Learning contact locations for pushing and orienting unknown objects,” in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Oct 2013, pp. 435–442.

[9] D. Katz and O. Brock, “Manipulating articulated objects with interactive perception,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 272–277.

[10] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, “Fast adaptation for effect-aware pushing,” in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, Oct 2011, pp. 614–621.

[11] M. Lau, J. Mitani, and T. Igarashi, “Automatic learning of pushing strategy for delivery of irregular-shaped objects,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3733–3738.

[12] S. Walker and J. Salisbury, “Pushing using learned manipulation maps,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 3808–3813.

[13] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, “Learning to poke by poking: Experiential learning of intuitive physics,” *CoRR*, vol. abs/1606.07419, 2016. [Online]. Available: http://arxiv.org/abs/1606.07419

[14] T. Meriçli, M. Veloso, and H. Akın, “Push-manipulation of complex passive mobile objects using experimentally acquired motion models,” *Autonomous Robots*, vol. 38, no. 3, pp. 317–329, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-014-9414-z

[15] T. Igarashi, Y. Kamiyama, and M. Inami, “A dipole field for object delivery by pushing on a flat surface.” in *ICRA*, 2010, pp. 5114–5119.

[16] S. Krivic, E. Ugur, and J. Piater, “A robust pushing skill for object delivery between obstacles,” in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug 2016, pp. 1184–1189.

[17] P. Guttorp and R. A. Lockhart, “Finding the location of a signal: A bayesian analysis,” *Journal of the American Statistical Association*, vol. 83, no. 402, pp. 322–330, 1988.

[18] D. Fink, “A compendium of conjugate priors,” 1997.

[19] T. Hermans, J. Rehg, and A. Bobick, “Decoupling behavior, perception, and control for autonomous learning of affordances,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 4989–4996.

[20] B. Ridge, D. Skocaj, and A. Leonardis, “Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 5047–5054.