



Pushing corridors for delivering unknown objects with a mobile robot

Senka Krivic¹ · Justus Piater¹

Received: 7 December 2017 / Accepted: 12 September 2018 / Published online: 3 October 2018
© The Author(s) 2018

Abstract

This work addresses the problem of object delivery with a mobile robot in real-world environments. We introduce a multilayer, modular pushing skill that enables a robot to push unknown objects in such environments. We present a strategy that guarantees obstacle avoidance for object delivery by introducing the concept of a pushing corridor. This allows pushing objects in scattered and dynamic environments while exploiting the available collision-free area. Moreover, to push unknown objects, we propose an adaptive pushing controller that learns local inverse models of robot-object interaction on the fly. We performed exhaustive tests showing that our controller can adapt to various unknown objects with different mass and friction distributions. We show empirically that the proposed pushing skill leads towards successful pushes without prior knowledge and experience. The experimental results also demonstrate that the robot can successfully deliver objects in complex scenarios.

Keywords Nonprehensile manipulation · Pushing skill · Adaptation · Robot learning

1 Introduction

Nonprehensile manipulation such as pushing with a robot base can play a significant role in complex robotic scenarios. If a mobile robot does not have an arm or it is already holding an object, pushing with a robot base can be used for conveying objects or clearing paths. In this paper, we address the problem of object delivery in complex and dynamic scenarios. In contrast to predefined standard factory settings, robots in everyday environments face many challenges, such as uncertainty about the object's behaviour under pushing and the environment dynamics.

Generally, it is not straightforward to design an analytical model describing the behaviour of an object under manipulation (Mason 1986; Yu et al. 2016). Objects may have diverse, even anisotropic properties under pushing. Visual features can be non-informative, or even misleading, for the description of object behaviours while pushing (Ruiz-Ugalde

et al. 2011). Recently presented data-driven models (Bauzá and Rodriguez 2017; Agrawal et al. 2016) outperform analytical models. However, acquiring data for such empirical models that capture the accurate dynamics of pushing interactions can be expensive or impossible. There is a wide range of diversity of possible interactions between the robot, an object and the environment due to a variety of object and environment properties in real-world scenarios. Thus, we are interested in the feasibility of push-manipulation with data acquired on the fly. Since the tracking the 3 DoF pose can be challenging within the push manipulation, we are interested in producing a robust method which needs only 2 DoF location of the object.

For a given task, often, partial knowledge of the system dynamics is sufficient for controller design (Bansal et al. 2017; Cehajic et al. 2017). A dynamic model is iteratively updated, evaluating the controller in online methods (Deisenroth et al. 2015). Instead of learning accurate models of the physical interaction, we take a task-specific approach where learning is goal-driven. A robot adapts the inverse model of the robot-object push interaction online. The model combines pushing and relocating movements based on the angle between the approach line and the desired pushing direction. Additionally, we introduce a feedback component, altering the reference control and enabling the robot to adapt to the object behaviour changes gradually. In this way, a robot can push the object towards a movable target, actively using the sensor data.

Electronic supplementary material The online version of this article (<https://doi.org/10.1007/s10514-018-9804-8>) contains supplementary material, which is available to authorized users.

✉ Senka Krivic
senka.krivic@student.uibk.ac.at

Justus Piater
justus.piater@uibk.ac.at

¹ Department of Computer Science, Universität Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria

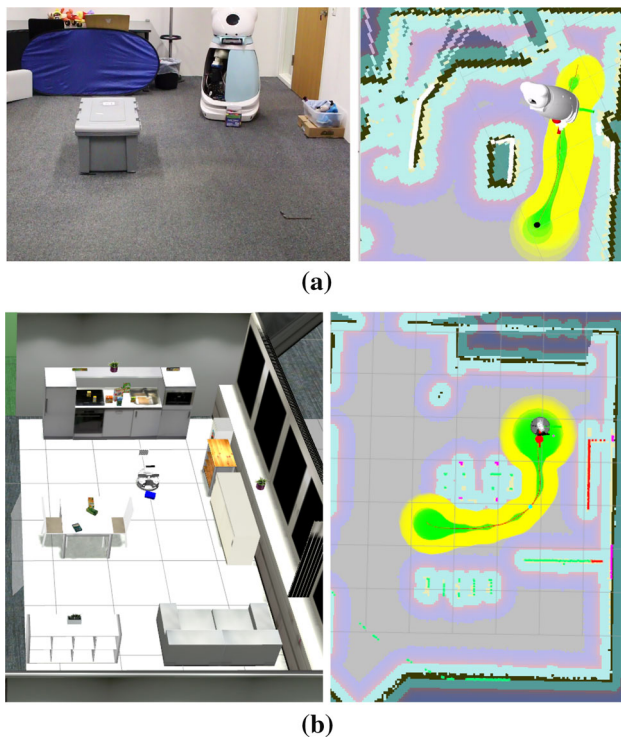


Fig. 1 Real and simulated experiments illustrating the pushing and object corridors. A collision-free path is obtained from a navigation system and determines the pushing corridor (yellow) and the object corridor (green). **a** Snapshots of the robot environment (left) and the visualisation tool RViz (right) taken during tests in a real-world scenario. Our robot “Roberta” based on a Festo Robotino platform was used in the experiments. **b** Example of a narrow pushing corridor in a kitchen environment. Snapshots of the Gazebo simulator (left) and the RViz visualisation (right) (Color figure online)

In various robotic scenarios, a robot has to operate in environments with scattered objects and tangled heaps on the floor, limiting its freedom to move. Unforeseen or movable obstacles might invalidate a robot’s plan. Thus, the pushing skill has to be reactive, both to changes in the environment and to the variety of situations which may occur while pushing an object.

Without exact models or exhaustive experience of the object behaviour, it is hard or even infeasible to push the object on a predefined path (Li and Zell 2006; Mericli et al. 2013). Building on our previous work (Krvic et al. 2016), where a robot was pushing in corridors with a fixed width, we introduce the concepts of *object* and *pushing corridors* (Fig. 1) as the representation of the available collision-free area between the robot and the pushing goal. We examine how pushing within corridors simplifies the object delivery in contrast to standard path tracking approaches and ensures collision avoidance.

In Sect. 2 we give an overview of the proposed approach and give details of it in Sects. 3, 4 and 5. We present experimental results in Sect. 6 and discuss them in Sect. 7. We give an overview of the relevant past work in Sect. 8 and conclude

Table 1 Mathematical symbols and notation

Symbol	Meaning
\mathcal{G}	Global frame
\mathcal{P}	Local pushing frame
$\hat{\mathbf{d}}$	Direction unit vector
$k \in \mathcal{N}$	Current time step
$\mathbf{r} \in \mathcal{R}^2$	Position of the robot in \mathcal{G}
$\mathbf{o} \in \mathcal{R}^2$	Position of the object in \mathcal{G}
$\mathbf{g} \in \mathcal{R}^2$	Desired goal position of the object in \mathcal{G}
$\mathbf{t} \in \mathcal{R}^2$	Intermediate pushing target
$\mathbf{v} \in \mathcal{R}^2$	Velocity vector
$\mathbf{u} \in \mathcal{R}^2$	Control command robot velocity vector
$\theta_{\mathbf{a}} \in \mathcal{R}$	Orientation of the vector \mathbf{a}
$r_{\mathbf{a}} \in \mathcal{R}$	Radius of the vector \mathbf{a}
$V \in \mathcal{R}$	Velocity magnitude
$d_r \in \mathcal{R}$	Robot diameter
$d_o \in \mathcal{R}$	Diameter of the object bounding blob
$\gamma \in \mathcal{R}$	Object movement error angle
$\alpha \in \mathcal{R}$	Orientation of the push-line in \mathcal{P}
$K \in \mathcal{R}^+$	Coefficient
\mathcal{M}	Map of environment
\mathbf{Q}	Collision-free pushing path
\mathbf{C}	Sequence of the path clearance
\mathbf{W}	Sequence of the corridor widths
E	Pushing corridor edge set of points
E_o	Object corridor edge set of points
A_k	Set of angles at the time step k
$d(\mathbf{a}, \mathbf{b})$	Euclidean distance between points \mathbf{a} and \mathbf{b}
$L_{\mathbf{Q}}(\mathbf{a}, \mathbf{b})$	Arc length of the path segment between points \mathbf{a} and \mathbf{b} on the path \mathbf{Q}
\mathbf{Q}_o	Object path
\mathbf{Q}_r	Robot path
$\psi(\cdot)$	Activation function

in Sect. 9. The notations given in Table 1 are used throughout this paper.

2 Reactive pushing skill

Given the challenges of the object delivery, described in Sect. 1, we introduce a modular pushing skill, as shown in Fig. 2.

Free space reasoning provides an abstraction of the free space in the environment where pushing can be achieved. It defines reconfigurable pushing and object corridors. This layer reacts to changes in the environment and updates the corridors. Section 3 introduces pushing corridors and describes this layer of the proposed pushing skill.

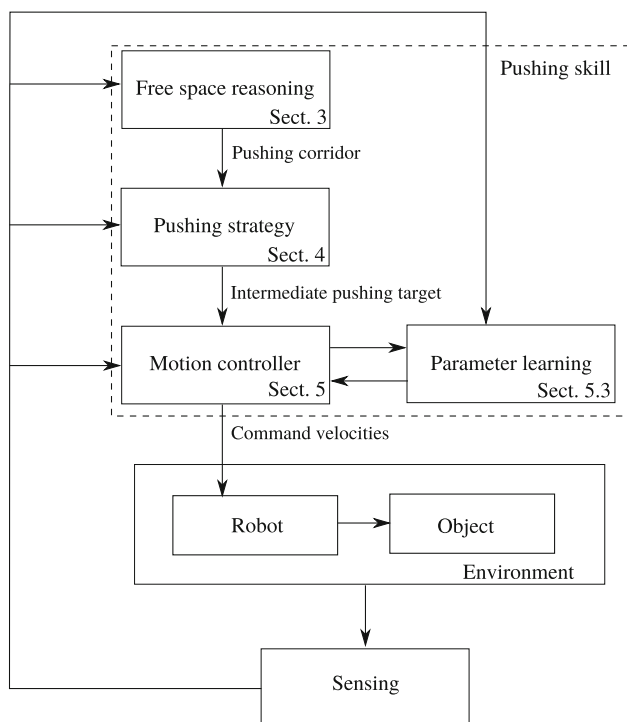


Fig. 2 The proposed structure of the pushing skill

Pushing strategy determines optimal intermediate pushing targets within the corridor. It ensures that the robot can push an object freely without violating the corridor. Section 4 introduces optimisation routines defining pushing strategies.

Adaptive pushing of an object towards a movable target is enabled through two modules which interact:

- The robot motion controller, which combines feedforward and feedback components, based on the desired pushing direction.
- The learning module which collects statistics about the exhibited object behaviour and adapts parameters of the motion controller.

Section 5 describes the details of the motion controller and the parameter learning module.

A skill like this can be incorporated into more complex modular robotic systems and used by a high-level planner, e.g. Konidaris et al. (2014), Hangl et al. (2016).

3 Corridors for pushing

Delivering objects in cluttered environments requires spatial reasoning. We bound the shapes of an object and a robot with circles. We assume that the size of a robot is determined by its bounding diameter d_r , and that the system can approximate

the diameter d_o of the bounding circle around the object to be pushed. The robot's task is to deliver an object from the current position \mathbf{o} to the goal position \mathbf{g} . We assume that the global position \mathbf{r} of the robot and the global position of an object \mathbf{o} , which represent the center of bounding circles, are available at each time step k . In addition, we assume that the map of the environment \mathcal{M} is available and that the robot is equipped with sensors to collect information for the map update at each time step.

The system is able to determine an optimal collision-free path \mathbf{Q} from the object's current position \mathbf{o} to the desired goal position \mathbf{g} . The collision-free path is given by a sequence of points $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \dots, \mathbf{q}_N)$, where $\mathbf{q}_N = \mathbf{g}$.

With the knowledge of the current map \mathcal{M} , the system is able to obtain the sequence of path clearances $\mathbf{C} = (c_1, c_2, c_3, \dots)$. Scalar values c_i correspond to the points $\mathbf{q}_i \in \mathbf{Q}$, representing the distance to the closest obstacle from each point.

3.1 Pushing and object corridors

The path \mathbf{Q} and the path clearance \mathbf{C} allow us to introduce the concept of a *pushing corridor*.

Definition 1 The pushing corridor is the ordered pair $(\mathbf{Q}, \mathbf{W}_p)$, representing the collision-free area around the path \mathbf{Q} from the start \mathbf{o} to the object's goal position \mathbf{g} , given by the sequence of corridor widths \mathbf{W}_p .

Corridor widths \mathbf{W}_p correspond to the clearance \mathbf{C} reduced by the robot's radius $d_r/2$. In this way they determine the set of points defining the edges of the pushing corridor E . Neither the robot position \mathbf{r} nor the object position \mathbf{o} , referring to their respective centroid, should violate the pushing corridor. To ensure this, the object has to move within the inner part of the corridor, always leaving enough space for the robot to correct its movements. Therefore, we also introduce the concept of an *object corridor*.

Definition 2 The object corridor is the ordered pair $(\mathbf{Q}, \mathbf{W}_o)$, representing the area the object is not allowed to leave, given by the path \mathbf{Q} from the start \mathbf{o} to the object's goal position \mathbf{g} , paired with the sequence of object corridor widths \mathbf{W}_o .

The object corridor edges are curves parallel to the pushing corridor edges, with an offset of $d_r/2 + d_o/2$, allowing the robot to move freely around the object. E_o is the set of points defining the edges of the object corridor, as shown in Fig. 3.

3.2 Corridors in dynamic environments

The configuration of the real-world environment changes often. The map \mathcal{M} is updated only for recently visited areas of the environment.

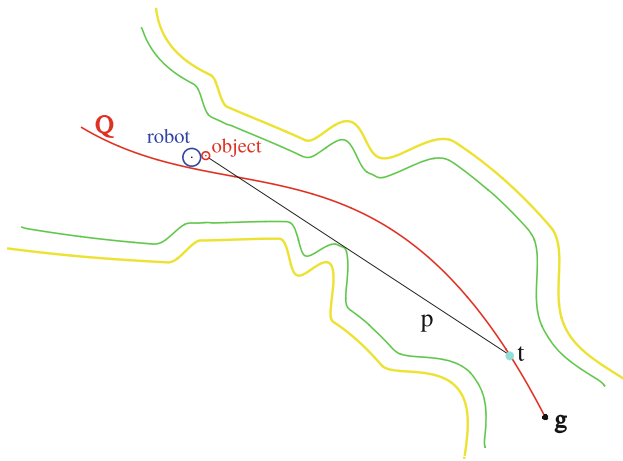


Fig. 3 Determining the next target point \mathbf{t} for pushing depends on the free space and current locations of the robot and the object. The red line denotes the pushing path \mathbf{Q} . The current object position \mathbf{o} and \mathbf{t} define the push-line p . The object corridor (green) is defined such that it keeps a distance of $d_r/2 + d_o/2$ from the pushing corridor (yellow), allowing the robot to move freely around the object (Color figure online)

Algorithm 1: UPDATE CORRIDOR

Input : Robot position \mathbf{r} , object position \mathbf{o} , sensor data S , current map \mathcal{M}

Output: Pushing corridor $(\mathbf{Q}', \mathbf{C}')$

```

1  $d_{obstacle} \leftarrow \text{DistanceRobotClosestObstacle}(S)$ ;
2  $d_{edge} \leftarrow \text{DistanceRobotCorridorEdge}((\mathbf{Q}, \mathbf{C}), \mathbf{r})$ ;
3 if  $d_{obstacle} \neq d_{edge}$  then
4   |  $\mathbf{Q}' \leftarrow \text{PathPlanning}(\mathcal{M}, \mathbf{o}, \mathbf{g}, d_r, d_o)$ 
5   |  $\mathbf{C}' \leftarrow \text{DistancePathObstacles}(\mathbf{Q}', \mathcal{M})$ ;
6 else
7   |  $(\mathbf{Q}', \mathbf{C}') \leftarrow (\mathbf{Q}, \mathbf{C})$ ;
8 end
9 return  $(\mathbf{Q}', \mathbf{C}')$ ;

```

The robot needs to accommodate possible changes in the environment and unforeseen objects while pushing. Thus, at each time step, the system checks if current sensor readings correspond to the current corridor (\mathbf{Q}, \mathbf{C}) . If changes occur, the set of widths \mathbf{C} is no longer valid and a new corridor is obtained, based on the current updated map of the environment. Algorithm 1 summarizes the procedure that is performed in the top layer of the pushing skill (Fig. 2).

4 Pushing strategy

Collisions can be avoided by making sure both the robot and the object follow the path \mathbf{Q} exactly. However, this requirement is not always necessary. Moreover, it poses strict, even infeasible constraints on the task of pushing an unknown object. Pushing corridors are reconfigurable, as described in the previous section, and can be of various shapes and sizes. To exploit the collision-free area, we utilise the concept of the *intermediate pushing target*.

At each time step, the robot pushes along the corridor by choosing the next intermediate target point \mathbf{t} (Fig. 3). The intermediate target and the object's current position form a *push-line* $\overline{\mathbf{o}\mathbf{t}}$, which represents a new desired pushing direction. Such a strategy allows straight-line pushes, while constraints of the corridor are not violated, simplifying the pushing path.

This procedure is similar to the concept of the look-ahead distance in navigation introduced by Coulter (1992), which determines how far along the path a robot should look from the current location to compute the velocity commands. In a pushing scenario, a short lookahead distance ensures better path tracking but might lead to difficulties in object movement control. As a consequence, this can produce excessive robot movements while correcting the object motion. By contrast, large lookahead distances enable long object trajectories but raise the possibility of corridor violations.

4.1 The intermediate pushing target

The choice of the intermediate pushing target determines the trajectory of an object movement. In an ideal case, with no obstacles, a robot could push the object directly towards the pushing goal \mathbf{g} . Otherwise, the object has to be pushed in a way that it advances towards the goal while avoiding obstacles. Thus, the system chooses intermediate targets which simplify and shorten the object movement trajectory.

However, the choice of pushing targets should not drive objects to unreachable configurations for push-manipulation. To keep the object far from the pushing corridor edges, an intermediate pushing target is chosen such that it lies on the path \mathbf{Q} . Moreover, the system chooses target points such that push-lines are also entirely within the object corridor.

Since objects are unknown to the robot, the resulting object movement directions can be unforeseen. We make the following assumptions of the object behaviour.

Assumption 1 The movement of the center of the bounding circle of the object \mathbf{o} does not deviate more than 90° from the pushing direction (Fig. 4a).

Assumption 2 The object has quasi-static properties.

These assumptions guarantee that a robot maintains contact with the object while pushing it and the object movement is incremental at each time step.

In a worst case scenario, the object is exactly at the edge of the object corridor. With the uncertainty of the expected object movement defined with Assumption 1, the push-line is chosen perpendicular to l , the tangent to the edge line E_o at the point closest to the object. Figure 4b demonstrates examples of push-lines where the object reached the object-corridor edge.

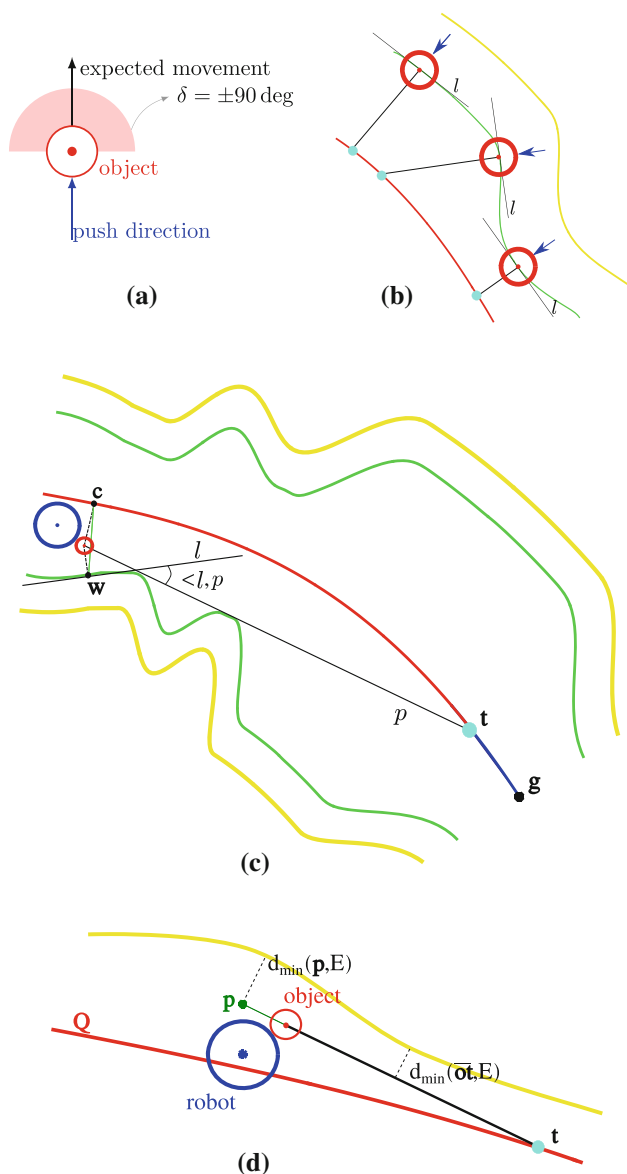


Fig. 4 **a** The actual object movement should not deviate more than $\delta = \pm 90^\circ$ from its desired movement. **b** Examples of push-lines that guarantee collision avoidance where the object reached the object-corridor edge; l denotes the tangent to the corridor edge at the point closest to the object. **c** Quantities used by the optimisation procedure to determine the pushing target. **d** The pushing strategy for narrow corridors. Point \mathbf{p} is the desired pushing position for the robot

These prerequisites allow us to define an optimisation procedure for determining the intermediate pushing target \mathbf{t} . As shown in Fig. 4c, at each time step k the intermediate pushing target can be defined as

$$\begin{aligned}
 \mathbf{t} &= \arg \min_{\mathbf{q}_i \in Q} L_Q(\mathbf{q}_i, \mathbf{g}) & (1) \\
 \text{s.t.} \quad \sin \angle \overline{\mathbf{ot}}, l &\geq \frac{d(\mathbf{o}, \mathbf{c})}{d(\mathbf{w}, \mathbf{c})} & (2) \\
 d_{\min}(\overline{\mathbf{ot}}, E_o) &> 0 & (3)
 \end{aligned}$$

where

- $L_Q(\mathbf{q}, \mathbf{g})$ is the arc length of the pushing path Q from the point \mathbf{q} to the goal point \mathbf{g} ,
- $\mathbf{w} \in E_o$ is the closest point on an object-corridor edge to the current object position \mathbf{o} ,
- \mathbf{c} is the closest point on the path Q to the current object position \mathbf{o} ,
- $\angle \overline{\mathbf{ot}}, l$ is the angle formed by the push-line $\overline{\mathbf{ot}}$ and the tangent l to the object corridor edge.
- $d_{\min}(\overline{\mathbf{ot}}, E_o)$ is the minimal distance between the push-line segment $\overline{\mathbf{ot}}$ and the edges of the object corridor.

The cost function (1) encourages the object to progress towards the goal, and promotes long push-lines. Condition (2) guarantees that neither the robot nor the object violate the corridor, even in cases of high path curvatures or the relocation of the robot. Finally, condition (3) ensures that the push-line does not touch the pushing-corridor edge.

Corridor non-violation guarantees follow from condition (2). In case the object advances towards the object corridor edge ($\mathbf{o} \rightarrow \mathbf{w}_j$), then condition (2) becomes $\sin \angle p, l \geq 1$. This enforces $\mathbf{t} = \mathbf{q}_i : l \perp \overline{\mathbf{oq}_i}$. Pushed towards this intermediate target and with Assumptions 1 and 2, the object will remain within the corridor.

4.2 Pushing in narrow corridors

Narrow corridors, where $\exists c_i \in C : c_i < d_r/2 + d_o/2$, often appear in real-world scenarios. An example of such a corridor is given in Fig. 1b. The previously presented strategy (1)–(3) imposes strict conditions on the robot movement for such corridors. To enable pushing in narrow corridors, we relax conditions (2) and (3).

The pushing direction has to be chosen in a way that a robot is able to fit between the object and the pushing corridor edge for the current object position at least. For this purpose, we define a *desired robot pushing position* \mathbf{p} on the push-line, defined as the segment $\overline{\mathbf{ot}}$, as shown in Fig. 4d, with an assumption that pushes are achievable if the robot is in the neighbourhood of the push-line.

In narrow corridors, it may happen that condition (3) is never fulfilled. Therefore, the push-line has to be far enough from the edges such that the larger body (i.e. the robot or the object) can fit beside the push-line (Fig. 4d).

The optimisation problem to define the intermediate pushing target with weaker conditions is:

$$\begin{aligned}
 \mathbf{t} &= \arg \min_{\mathbf{q}_i \in Q} L_Q(\mathbf{q}_i, \mathbf{g}) & (4) \\
 \text{s.t.} \quad d_{\min}(\mathbf{p}, E) &> 0 & (5) \\
 d_{\min}(\overline{\mathbf{ot}}, E) &> \max(d_o/2, d_r/2) & (6)
 \end{aligned}$$

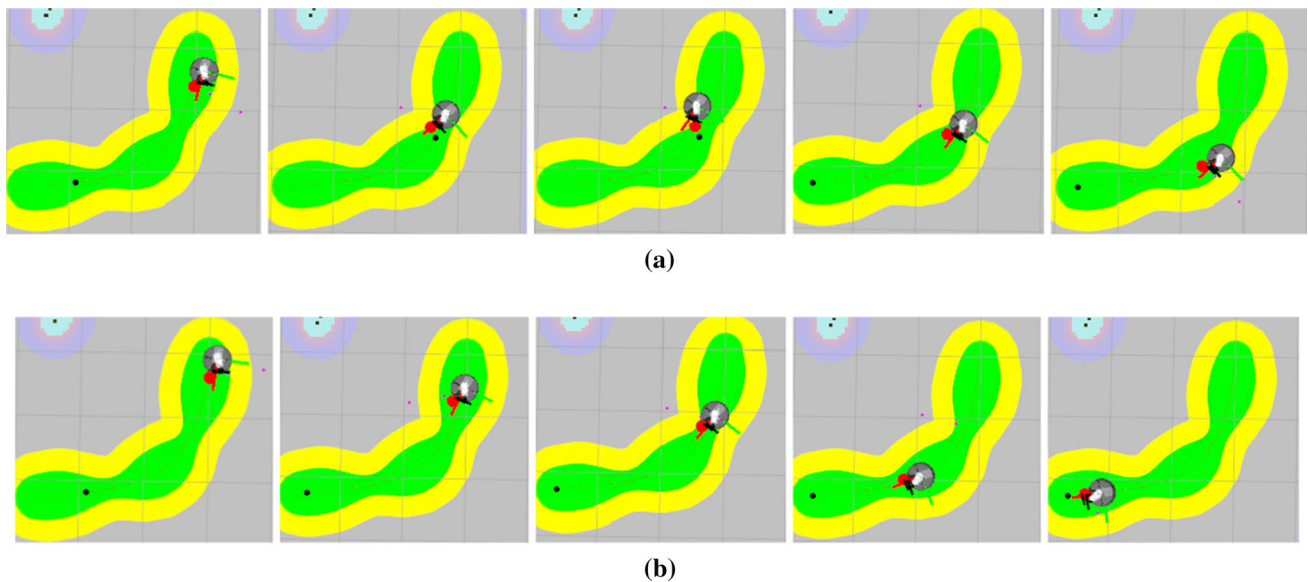


Fig. 5 Comparison of the presented pushing strategies for the same experimental setup. The black coloured point on the path represents the current intermediate target \mathbf{t} . The red circle represents the object

bounding blob. Compare the intermediate targets between **a** the *strict* strategy and **b** the *relaxed* strategy (Color figure online)

where

- \mathbf{p} is the point obtained by extending the push-line segment $\overline{\mathbf{ot}}$ by a distance of $d_o/2 + d_r/2$ from the object centre, as shown in Fig. 4d,
- $d_{\min}(\mathbf{p}, E)$ is the minimum distance between the point \mathbf{p} and the edges of the pushing corridor,
- $d_{\min}(\overline{\mathbf{ot}}, E)$ is the minimum distance between the push-line segment and the edges of the pushing corridor.

These conditions do not guarantee collision avoidance but often result in successful pushes, as demonstrated in experiments presented in Sect. 6. The optimization problem (1)–(3) is referred to as the *strict* strategy, and that defined by (4)–(6) as the *relaxed* strategy. Examples of pushing targets chosen with both strategies are given in Fig. 5. Both strategies will result in a similar pushing target if the object is near the pushing path. If the object is close to the edge of the object corridor, the *strict* strategy will choose a pushing target with a short distance, while the *relaxed* strategy still produces long push-lines.

5 Adaptive pushing

The push-behaviour of the objects depends on different properties, such as mass distribution of the object, frictional forces, the object's shape, contact points and pushing dynamics, making it uncertain. Humans are able to quickly discover strategies for pushing, by reacting to the object's motion,

using senses and intuition. We try to build in similar concepts for robots, proposing a controller which adapts to the object's behaviours under pushing.

The proposed controller is adaptive for different shapes and properties of objects and the robot. At the beginning of a push, parameters of the controller are set such they correspond to the pushing interaction of two circles. While pushing a robot collects statistics and adapts parameters of the controller to the exhibited pushing behaviour. With an assumption that object shape can be approximated with an circle, angular velocities are ignored.

The proposed controller (Fig. 6a) is a cascade of controllers for object motion and robot motion. The controller in the outer loop controls the object motion by determining the desired robot velocity \mathbf{u} . This vector is an input for the inner loop which consists of the robot's velocity controller. We assume that this controller is responsive and fast concerning the command velocity. This controller depends on the dynamics of the robot. In the following Sect. 5.1, we discuss in detail how object motion controller is designed.

The input for it is the error of the object movement direction is given by Fig. 6b:

$$\gamma = \theta_o^d - \theta_o$$

where the desired object movement direction θ_o^d is defined by the push-line $\overline{\mathbf{ot}}$. While pushing, at each time step k , an error reading γ_k is added to the error trace $\Gamma_k = \{\gamma_1, \dots, \gamma_k\}$.

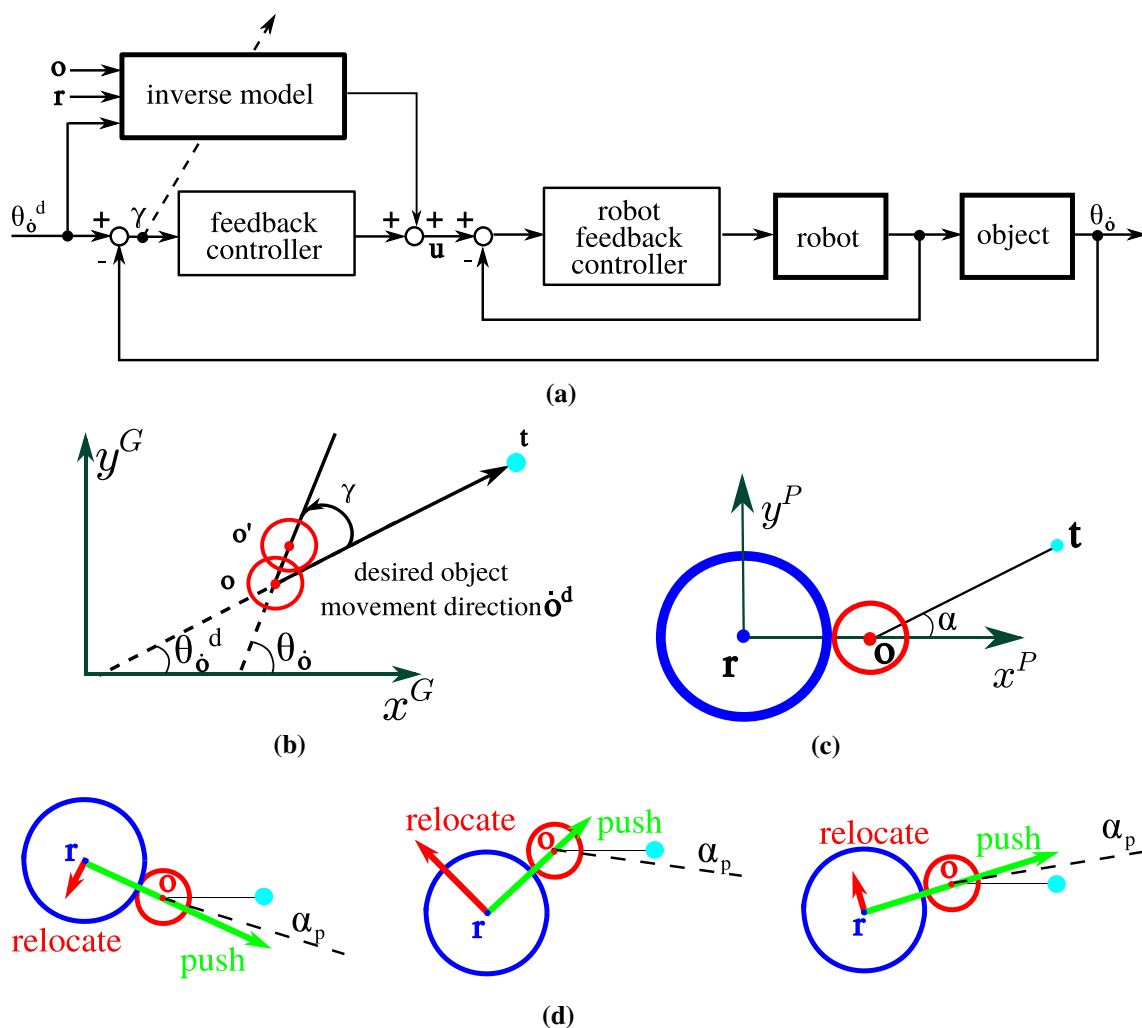


Fig. 6 **a** The proposed adaptive motion control loop consists of an inverse model representing the adaptive feedforward component, and a feedback controller as described in Sect. 5.1. The inner loop consist of the robot’s velocity controller. **b** The error of the object movement γ .

c The angle α , defined in the push frame \mathcal{P} , is used to determine the robot’s movement direction. **d** Examples of push and relocate vectors for different locations of the robot, the object and the target point and different values of the angle α_p

5.1 Object motion controller

The intermediate target moves as the object moves. While pushing, a robot needs to adjust its position with respect to the new desired pushing direction. If a robot is in a position from which it cannot directly push the object, it should relocate to the position from which the push can be achieved. Thus, two primitive robot movements are necessary: *pushing* and *relocating*.

The primitive movements pushing and relocating, were utilized in previous work. Hermans et al. (2013c) introduced a proportional controller with components for pushing and for moving a pusher behind the object. Saito et al. (2014) also introduced a recovery action to reposition the robot behind the object while pushing. Igarashi et al. (2010) developed

a strategy combining motions for orbiting around the object and pushing it. The final motion model resembles dipole field lines. The model is derived for a pusher and a slider with circular shapes, ignoring physical properties of real-world objects.

Guided by their results, we define the inverse model of the robot-object pushing interaction in a similar way, by blending these two movements in the feedforward controller component. The output of the inverse model is given by

$$v_{\text{reference}} = \psi_{\text{push}}(\alpha)\hat{d}_{\text{push}} + \psi_{\text{relocate}}(\alpha)\hat{d}_{\text{relocate}} \tag{7}$$

where \hat{d}_{push} and $\hat{d}_{\text{relocate}}$ are unit vectors determining the movement directions for pushing and relocating, as discussed in Sect. 5.2, and ψ_{push} and ψ_{relocate} are the corresponding activation functions. The system adapts the activation func-

tions ψ_{push} and ψ_{relocate} taking the observed object movement behaviour into account. This results in smooth transitions between pushing and relocating. The adaptation of functions is done through learning as described in Sect. 5.3.

The feedback term is defined by a proportional controller and an adaptive steady component. The control velocity command for the robot \mathbf{u} is given in polar coordinates as

$$\begin{aligned} \theta_{\mathbf{u}} &= \underbrace{\theta_{\mathbf{v}}^{\text{reference}}}_{\text{feedforward term}} - \underbrace{K_{\mu_{\gamma}}\mu_{\gamma} - K_{\gamma}\gamma}_{\text{feedback term}} \\ r_{\mathbf{u}} &= V \end{aligned} \tag{8}$$

where μ_{γ} is the mean of the observed $\gamma \in \Gamma_k$, and $K_{\mu_{\gamma}}$ and K_{γ} are corresponding gains. V is the desired constant magnitude of the robot velocity. The adaptive part of the feedback controller μ_{γ} compensates for constant deviations in the object movement directions which can be caused by non-uniform distributions of mass and friction.

The behaviour of the system gradually changes from the slow feedback control to the rapid feedforward control as the adaptive component learns the inverse model of the current robot-object interaction.

5.2 Primitive movement actions for the pushing skill

To simplify the description, let us assign a local pushing frame \mathcal{P} to the object \mathbf{o} where the $x^{\mathcal{P}}$ axis coincides with the direction $\overrightarrow{\mathbf{r}\mathbf{o}}$.

The primitive push direction is defined by the vector of the robot’s direct movement towards the object $\overrightarrow{\mathbf{r}\mathbf{o}}$. The relocation movement has to drive the robot to a position from where pushes are achievable. The relocation direction is orthogonal to the primitive push direction, i.e., $\hat{\mathbf{d}}_{\text{push}} \perp \hat{\mathbf{d}}_{\text{relocate}}$. Their directions are defined by the angle $\alpha \in (-\pi, \pi]$ which is the current orientation of the push-line $\overrightarrow{\mathbf{o}\mathbf{t}}$ in the pushing frame as in Fig. 6c.

Push and relocate directions are defined as:

$$\hat{\mathbf{d}}_{\text{push}}^{\mathcal{P}} = \text{sgn}(\cos \alpha) \hat{\mathbf{i}}^{\mathcal{P}} \tag{9}$$

$$\hat{\mathbf{d}}_{\text{relocate}}^{\mathcal{P}} = \text{sgn}(\sin(\alpha_p - \alpha)) \hat{\mathbf{j}}^{\mathcal{P}} \tag{10}$$

where $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ are unit vector in directions of the $x^{\mathcal{P}}$ axis and of the $y^{\mathcal{P}}$ axis respectively, α_p is the value of α for which the robot should move only in the push direction $\hat{\mathbf{d}}_{\text{push}}$ such that the object advances on the push-line. For an object with ideal properties this value is $\alpha_p = 0$. The systems estimates the value of α_p for a particular object on the fly. The sign function in Eq. (9) enables the escapes from the front side of the object in case that $|\alpha| > \pi/2$. This direction blended with the relocation direction drives the robot around the object such that $\alpha \rightarrow \alpha_p$. Examples for the push and relocate vectors are shown in Fig. 6d.

5.3 Learning activation functions

A robot needs to push various sets of objects which manifest different properties during pushing. Thus, equal parameters of the push activation function in Eq. (7) for all of them will not result in successful pushing. To adapt the activation functions to specific objects, we let the robot learn the push activation function $\psi_{\text{push}}(\alpha)$ on the fly.

If the robot is behind the object ($\alpha \approx 0$), it should be able to push it towards the target. However, these values may vary based on the object and environment properties. Thus, a robot needs to learn which values of α result in the object advancing towards the target.

We characterize deviations α that are good for pushing as deviations for which a robot is able to decrease the error of the object movement direction ($|\dot{\gamma}| < 0$). Assuming these can be modelled as random variables, we describe them with a *von Mises* distribution:

$$p(\alpha|\mu, \kappa) = \frac{e^{\kappa \cos(\alpha-\mu)}}{2\pi I_0(\kappa)} \tag{11}$$

where $0 \leq \mu \leq 2\pi$, $0 \leq \kappa \leq \infty$ 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 $\kappa \rightarrow \infty$ to a point distribution. The distribution of angles α which resulted in successful pushes defines the deviation $\alpha_p = E[\alpha|\mu, \kappa]$ in (10).

We estimate the model parameters μ and κ using Maximum a-posteriori (MAP) estimation. Given evidence of angles $A_k = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$ that resulted in successful pushes, we can make an estimate of the model by Bayes’ rule:

$$p(\mu, \kappa|A_k) = \frac{p(A_k|\mu, \kappa)p(\mu, \kappa)}{p(A_k)} \tag{12}$$

where $p(A_k|\mu, \kappa)$ is the likelihood of the observed data A_k , $p(\mu, \kappa|A_k)$ is the posterior of distribution of μ, κ , and $p(\mu, \kappa)$ is the prior in this setting.

As soon as a new observation α_{k+1} is obtained, we update the likelihood $p(A_{k+1}|\mu, \kappa)$ and calculate the new posterior $p(\mu, \kappa|A_{k+1})$. Determining the posterior is done using a bivariate conjugate prior for the von Mises distribution with unknown concentration and direction, originally developed by Guttorp and Lockhart (1988) and revised by Fink (1997). The conjugate prior for the von Mises distribution is defined with hyperparameters R_0, c and ϕ as

$$p(\mu, \kappa|R_0, c, \phi) = \frac{1}{K} \frac{e^{R_0\kappa \cos(\phi-\mu)}}{2\pi I_0(\kappa)^c} \tag{13}$$

where K is a normalizing constant given by

$$K = \int_0^\infty \frac{I_0(R_0\kappa)}{I_0(\kappa)^c} d\kappa.$$

The prior hyperparameters values are set to $R_0 = 5.84$, $c = 7.0$ and $\phi = 0.0$. These values correspond to an object with a circular shape and uniform mass and friction distributions.

Using the model (11) that describes angles α that resulted in successful pushes, we define the push activation function as

$$\psi_{\text{push}}(\alpha) \propto p(\alpha|\hat{\mu}, \hat{\kappa}),$$

where $\hat{\mu}$, $\hat{\kappa}$ are maximum-a-posteriori estimates of the distribution of angles α that resulted in successful pushes. Since the activation function $\psi_{\text{push}}(\alpha)$ should produce values between 0 and 1, the output of the model is divided by the value of the von Mises distribution at $\alpha = \hat{\mu}$.

The final vector of a robot movement (7) is determined as the sum of push and relocation vectors. Thus, the relocation activation function is defined as

$$\psi_{\text{relocate}}(\alpha) = \sqrt{1 - \psi_{\text{push}}^2(\alpha)}.$$

6 Experimental results

We conducted several experiments, both in simulation and in real environments, to examine the effectiveness of object delivery, the pushing effort and the impact of varying objects of the proposed method.

6.1 Implementation details

We used a holonomic robot with a circular base, a Robotino¹ by Festo, in both simulated and real environments. The diameter of our Robotino is $d_r = 46$ cm. In real-world scenarios, we also used a SQUIRREL² robot, shown in Fig. 1a, which is also based on a Robotino platform. The diameter of the bounding circle of this robot is 54 cm. The simulator used is Gazebo.³ The robot is equipped with a camera, which is used for object tracking. The rotational degree of freedom of the robot was used to control the robot's orientation, such that it keeps the object in the camera's view. For this purpose, we used a proportional-derivative controller to control the angular velocity $\dot{\theta}_r$ of the robot, i.e. $\dot{\theta}_r = PD(e_r^\theta)$ where

¹ <http://www.festo-didactic.com/int-en/services/robotino>.

² <http://www.squirrel-project.eu/>.

³ <http://gazebo.org/>.

$e_{\theta_r} = \theta_{\text{over}} - \theta_r$. The velocity controller in the inner loop is the Robotino's built-in velocity controller.

For object tracking we used two different methods, AR-tag tracking using the ARToolKitPlus⁴ software library for the Robot Operating System (ROS), and the object pose tracker by Prankl et al. (2015). The algorithm used for mapping is GMapping (Grisetti et al. 2007). A spatial resolution of 0.05 m was used to build the map. Localization is done using the Rao-Blackwellized PF SLAM method (Thrun et al. 2005). The algorithm used for path planning is Dijkstra provided by ROS. Paths are obtained for the object shape of a polygon corresponding to the robot and the object size. The path resolution was set to 0.05 m. To detect movable obstacles and changes of corridors we used IR distance sensors on the Robotino's base. Due to the sensor and odometry inaccuracies, in Algorithm 1 a threshold of 0.05 m for expected readings was added.

$K_\gamma = 0.1$ and $K_{\mu_\gamma} = 0.05$ were set for (8) in all experiments. $V = 0.3$ m/s was set in the simulation tests and $V = 0.1$ m/s was set in all real-world experiments.

6.2 Performance measures

To examine the effectiveness of the presented approach, we quantify the success of object deliveries. A push was considered successful if the distance of the object to the desired goal location was below the precision limit, the corridor was not violated by the robot nor by the object, and delivery was done within the time limit of 400 s.

As a measure of the pushing effort for the object delivery in different scenarios, we compare mean arc lengths of the object path L_{Q_o} and the robot path L_{Q_r} and durations of successful object deliveries.

6.3 Experiments

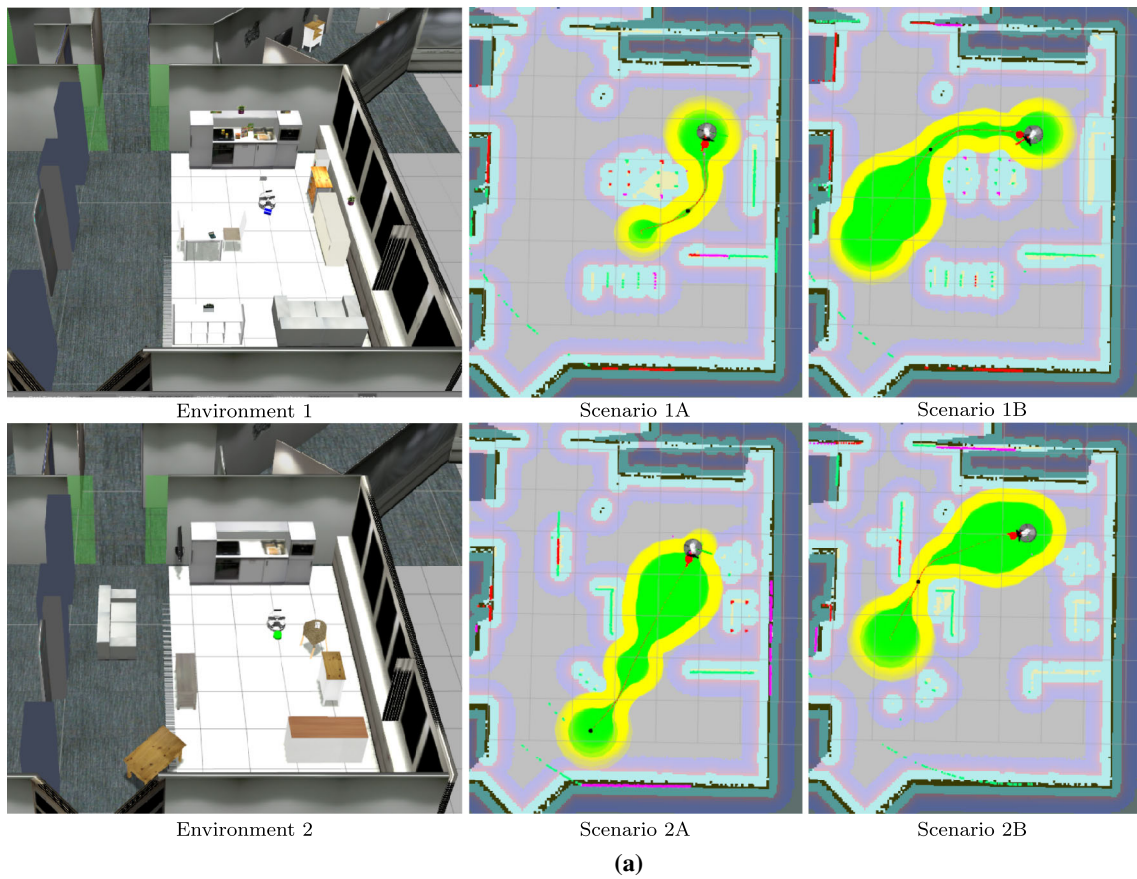
6.3.1 Experiment 1: pushing strategies

The first set of experiments addressed the problem of utilization of pushing corridors and comparison of proposed pushing strategies.

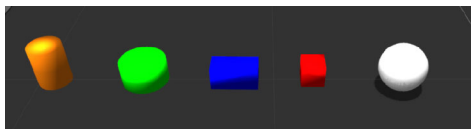
It examined the simplification of the pushing path with the strategies *strict* and *relaxed* presented in Sect. 4.

Experimental setup Experiments were performed in simulation in four different scenarios as shown in Fig. 7a. For each test, a new pushing path and its clearance were obtained using the navigation system which resulted in slight variations of corridors for same scenarios. We let the robot push five different objects of basic shapes (Fig. 7b) in each scenario. The precision limit of the object delivery was set to 0.05 m. 20

⁴ http://wiki.ros.org/v4r_artoolkitplus.

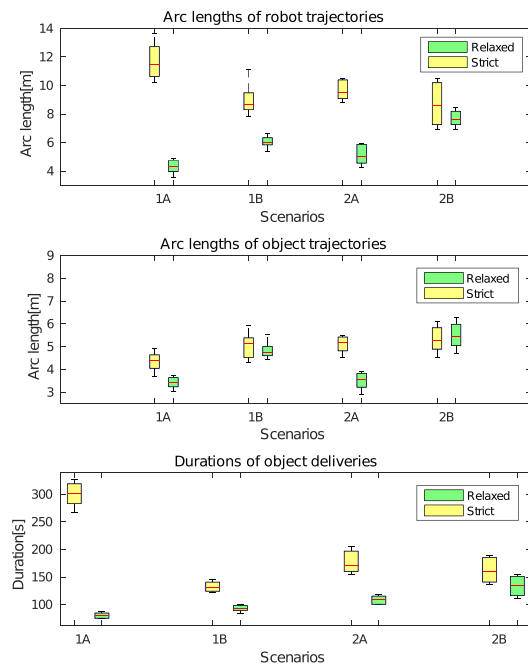


(a)



The object set
 'orange cylinder' $d_o = 0.14\text{m}$
 'green cylinder' $d_o = 0.18\text{m}$
 'blue box' $d_o = 0.224\text{m}$
 'red box' $d_o = 0.142\text{m}$
 'white ball' $d_o = 0.2\text{m}$

(b)



(c)

Fig. 7 a Pushing environments used to evaluate *strict* and *relaxed* strategies. b The object set used in simulated experiments. c Results of object deliveries in simulation environments. Five object deliveries were tested for each scenario and each strategy (Color figure online)

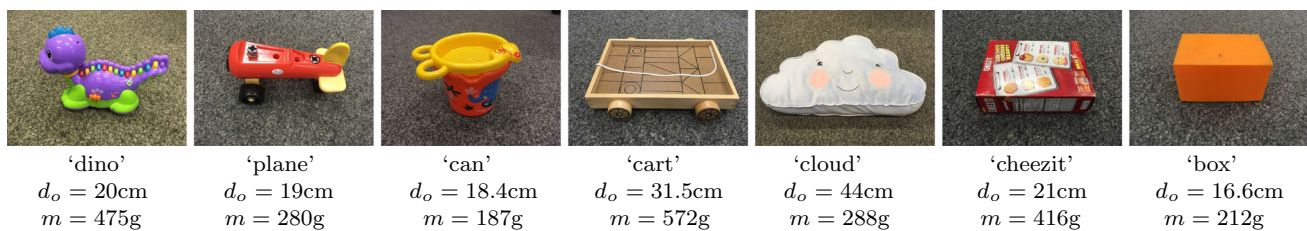


Fig. 8 The set of objects used in the experiments, with their sizes and masses

object deliveries were performed with each proposed pushing strategy. In Scenarios 1A, 2A, and 2B, the corridor contains such narrow passages that there the object corridor disappears entirely. Therefore, no guarantees can be made, and there is no solution which can be found by the *strict* strategy. Whenever there was no solution for the optimisation procedure Eqs. (1)–(3) of the *strict* strategy, the point on the path closest to the object was taken as the current target for this scenario.

Results The *strict* strategy resulted in 19/20 successful object deliveries, while the *relaxed* strategy resulted in 17/20 successful object deliveries. A comparison of the pushing effort for both strategies is given in Fig. 7c. The single failure of the *strict* strategy occurred in Scenario 2B, where the object ended up in a position unreachable by the robot while moving within the narrow part of the corridor between the grey sofa and the grey drawer. The fallback strategy was triggered only in the narrow part of the corridors for the *strict* strategy, mostly in Scenario 2B.

6.3.2 Experiment 2: exploitation of free area

This set of experiments addressed the problem of exploitation of the available free area and adaptation while pushing various objects with a real robot. We compared the proposed approach of pushing within the corridor with the *strict* and *relaxed* strategies against pushing on the collision-free path as a standard approach. In the path following strategy, the intermediate pushing target is defined by a fixed a look-ahead distance. At each time step, the intermediate target is chosen as a point on the path at a distance of 0.1 m from the orthogonal projection of the object position on the path. We refer to this strategy as the *fixed-lookahead* strategy.

Experimental setup Our benchmark problem is a cluttered children's room. The set of objects used for pushing experiments contained objects of various shapes, sizes, frictional properties, mass distributions and quasi-static properties which appear in this and similar household scenarios, as shown in Fig. 8. We designed synthetic corridors to compare strategies. Corridor 1 and Corridor 2 are shown in Fig. 9a and b respectively. The precision limit of the object delivery

was set to 0.05 m. We performed three trials of delivery with the real robot for each particular corridor with each object. AR-tag tracking was used in this set of experiments.

Results Pushing efforts and success rates of Experiment 2 are given in Table 2. Failures were caused by the robot pushing the object out of the corridor or to an area where it was no longer able to push it back to the inner part. Examples of pushes from this set of experiments are given in Online Resource 1 and 2.

6.3.3 Experiment 3: controllers

The third set of experiments was designed to test the effectiveness of the proposed pushing motion controller for pushing unknown objects. We compared the proposed adaptive controller with three non-learning controllers for pushing unknown objects.

To examine the effect of adaptiveness of components in the proposed controller, we compared it to a *non-adaptive* version of our proposed controller where all adaptive values are held fixed during a push. We set the values of ψ_{push} to correspond to prior hyperparameter values $R_0 = 5.84$, $c = 7.0$ and $\phi = 0.0$, and $\alpha_p = 0$.

The second controller for comparison was the *centroid alignment controller* for pushing towards the centroid of an object from a given point (Hermans et al. 2013c). In this controller, the robot control command is defined as $\mathbf{u}^G = K_g e_g + K_c e_c$, where e_g is the vector defined by the distance of the object position to the target not longer than 0.2m, and e_c is the vector defining the robot displacement from the push line. $K_g = 0.2$ and $K_c = 0.4$ were set in all experiments with the velocity magnitude saturation of 0.2m/s . In the original work these coefficients were set to $K_g = 0.1$ and $K_c = 0.2$ which produced very slow motions for the robot base when object is near the target.

We also compared our controller with the *dipole-field* pushing method proposed by Igarashi et al. (2010). The robot command for this controller is defined by $\mathbf{u}^P = V \begin{bmatrix} \cos \alpha \\ -\sin \alpha \end{bmatrix}$ where α is defined the same as for our controller, as in Fig. 6b, and V is the velocity magnitude. Vectors of the robot move-

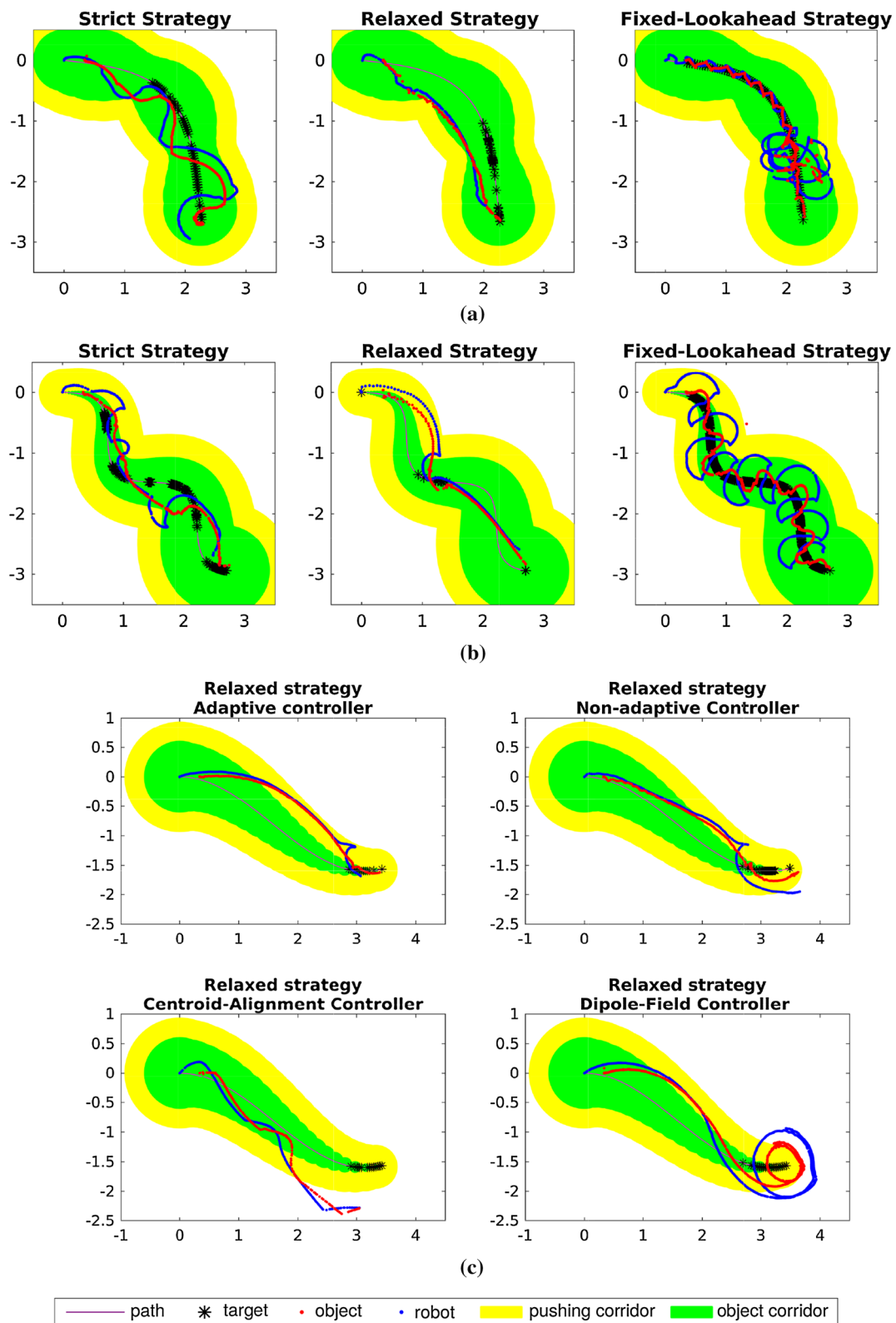


Fig. 9 **a** Examples of pushing trajectories for different pushing strategies from the 2nd set of experiments for Corridor 1 and the object ‘dino’. **b** Examples of pushing trajectories for different pushing strategies for

Corridor 2 and the object ‘cloud’. **c** Examples of pushing trajectories for different pushing controllers from the 3rd set of experiments for Corridor 3, the object ‘plane’ and the relaxed strategy

Table 2 Comparison of pushing effort for different pushing strategies

Strategy	L_{Q_r} (m)	L_{Q_o} (m)	Time (s)	Success
Corridor 1				
<i>Fixed-look</i>	9.5867	8.7938	120.9857	18/21
<i>Strict</i>	4.7355	4.6426	59.0214	20/21
<i>Relaxed</i>	6.1826	6.1801	69.2667	19/21
Corridor 2				
<i>Fixed-look</i>	8.0976	7.5975	101.2600	18/21
<i>Strict</i>	7.2940	6.2419	88.3857	18/21
<i>Relaxed</i>	6.1147	5.8851	46.0042	17/21

ment resemble dipole-like fields that guide the robot behind the object and push it towards the target.

Experimental setup We let the real robot push objects, the same as in Experiment 2, using the AR-tag tracking method. We used only the *relaxed* strategy to determine the pushing target in this set of experiments. We also created a synthetic scenario defined by Corridor 3 in Fig. 9c. The precision limit of the object delivery was set to 0.1 m, since the smaller limit of 0.05 m was too low for the comparison of controllers due to high failure rates.

Results Success rates and pushing efforts of successful deliveries from this set of experiments are given in Table 3. In the case of the *dipole-field* controller, two failures appeared due to the robot orbiting around the object and reaching the time limit. In the case of the *centroid* controller, four failures appeared due to the steep change of the pushing direction, where the controller was not able to steer the robot behind the object without pushing it out of the corridor. Our *non-adaptive* controller was successful at pushing objects of uniform mass and friction distribution such as ‘can’ or ‘box’, but other objects were pushed out of the corridor by this controller. Successful object deliveries with *non-adaptive* controller resulted in smooth trajectories, and the lowest pushing effort.

The pushing success differs for each object. Thus we report the success of deliveries for all individual objects in Table 4. Examples of pushes from this set of experiments are shown in Online Resource 3.

6.3.4 Experiment 4: real-world scenario

The fourth set of experiments was performed with the real robot in cluttered dynamic environments. The task was to push objects among scattered toys and tangled heaps on the floor to designated locations. Examples of pushes are presented in Online Resource 4.

7 Discussion

7.1 Pushing strategies

Both strategies presented (*strict* and *relaxed* conditions) showed a high success rate in the object delivery task for pushing unknown objects in Experiments 1 and 2. The proposed strategy with *strict* conditions showed to be the best in the case of wide pushing corridors. However, pushing efforts and delivery times are much higher than for the strategy with *relaxed* conditions, as shown in Fig. 7c. For narrow corridors, such as in Scenario 1A in Fig. 7a, the delivery time is the longest for the *strict* strategy, since the robot tries to keep the object close to the pushing path. The *relaxed* strategy can cause the object to get close to an obstacle, to the point that it is no longer possible for the robot to manipulate it. Thus, the success rate for this strategy is lower.

In real-world scenarios, a part of corridors might be very narrow. In practice, it would be possible to switch between *strict* and *relaxed* strategies based on some criteria, e.g. observing the ratio of the minimum corridor width in a corridor segment and diameters of the robot and the object.

7.2 Exploitation of free area

The examination of robot trajectories reveals that the robot simplifies the pushing task, as allowed by the corridor. Such sample executions are provided in Fig. 9a, b in Experiment 2. Our algorithm results in long, smooth pushes as long as they are possible. The robot tries to push an object towards the final goal if possible, but becomes more conservative as soon as deviations from a desired direction are observed. Contrary to that, the strategy of pushing an object along the path brings additional constraints that result in longer execution times and higher pushing efforts (Table 2). On the other hand, keeping the object on the path, if possible, guarantees obstacle avoidance.

7.3 Adaptation

The results given in Table 4 show that the proposed method is robust with regard to the change of the corridor shape and the shapes and sizes of objects. While pushing, the robot can have single or multiple contact points with the object. The pushing success rate depends on observations of the object behaviour, especially observations made at the beginning of a push. The motion is generated for a circular robot and an object of circular shape with uniform friction and mass distribution based on the prior values at the start of a push. If the object behaviour is very different than that and the corridor is narrow at the start, then it becomes difficult for the robot to keep the object within the corridor. An example of such a scenario is

Table 3 Comparison of pushing effort for different push controllers in Corridor 3 with *relaxed* strategy

Strategy	L_{Q_r} (m)	L_{Q_o} (m)	Time (s)	Success
<i>Adaptive</i>	4.3529	4.4050	53.6308	19/21
<i>Non-adaptive</i>	3.7564	3.8940	37.2708	12/21
<i>Dipole-field</i>	4.5969	4.7584	47.5775	10/21
<i>Centroid</i>	4.1633	4.548	110.0542	8/21

Table 4 Pushing success counts for each object

Strategy	<i>Relaxed</i>				
	<i>Adaptive</i>	<i>Adaptive</i>	<i>Non-adaptive</i>	<i>Centroid</i>	<i>Dipole-field</i>
Controller	<i>Adaptive</i>	<i>Adaptive</i>	<i>Non-adaptive</i>	<i>Centroid</i>	<i>Dipole-field</i>
‘dino’	5/6	8/9	1/3	1/3	1/3
‘plane’	5/6	8/9	1/3	1/3	1/3
‘can’	6/6	9/9	3/3	3/3	3/3
‘cart’	5/6	6/9	0/3	0/3	2/3
‘cloud’	6/6	8/9	1/3	1/3	1/3
‘cheezit’	5/6	8/9	2/3	0/3	1/3
‘box’	6/6	9/9	3/3	2/3	1/3

pushing the ‘cart’ in Corridor 2 in Experiment 2. Thus, the success rate for this object is lower compared to others.

7.4 Robustness

The learning process with regard to object behaviour varies, based on the complexity of an object’s behaviour, as can be seen in the examples given in Fig. 10. Objects with wheels, such as ‘cart’ and ‘plane’, bring additional challenges for the proposed strategy, since they do not fulfil the conditions for object delivery guarantees. This affects the success rate of the object delivery of these objects, as shown in Table 4. An object with a uniform mass distribution and a simple shape, such as the object ‘can’, appears to be the easiest to push. However, other objects from the set with quasi-static properties appear to be easy to push in different scenarios. The robot adapts its movement incrementally, based on observations. Most objects have an irregular shape, and the activation functions also adapt to changes in the contact locations. Such a change can be observed for the object ‘dino’ in Fig. 10, which the robot pushed from different sides, due to changes of the intermediate target.

7.5 Difficulties

There occurred some difficulties while performing real-world tests. Controlling the object movement is challenging, due to uncertainties about the location and movement of objects. The success of the proposed approach largely depends on vision-based measurements of the motion of

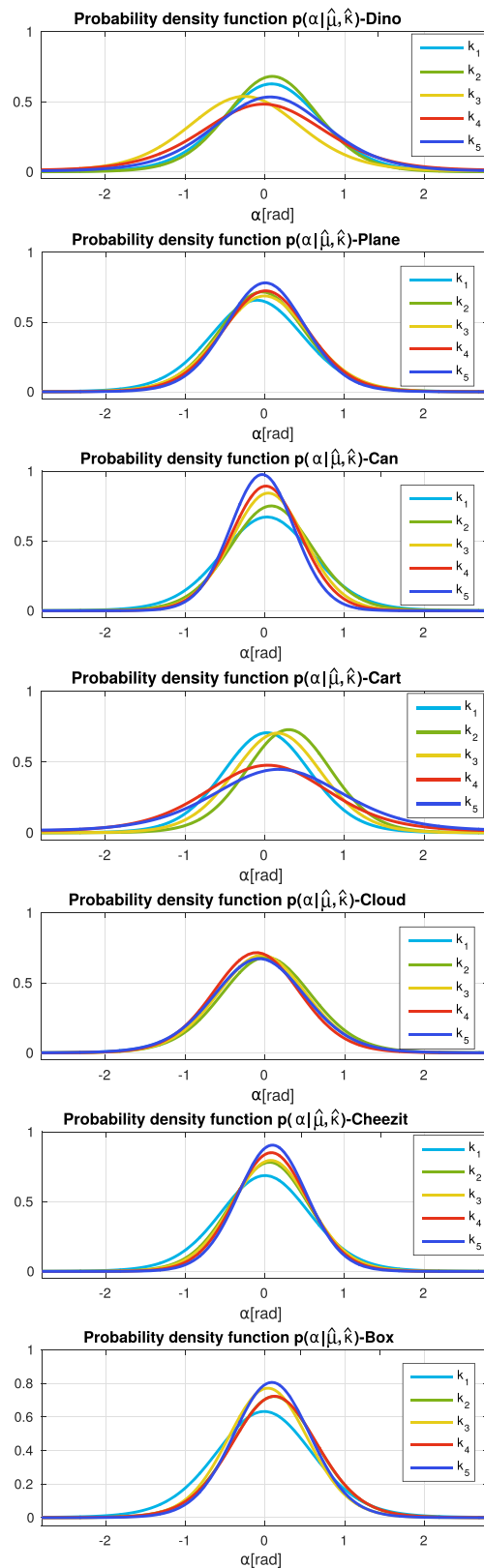


Fig. 10 Changes of probability density functions which characterize activation functions ψ_{push} in five different time steps ($k_1 < k_2 < k_3 < k_4 < k_5$) during a single push in Corridor 1 using the *relaxed* strategy (Color figure online)

objects, and the robot's localization. However, two tested tracking methods demonstrated a high enough accuracy for the execution of real-time deliveries. The proposed approach is robust enough to manage the constant errors of an object tracking method, such as the wrongly estimated center of the object. The main issue appears to be if the camera loses the object from view. Another limitation is that the camera is mounted on the robot, and the object has to fit within the camera's view, placing an upper bound on their allowed size.

8 Related work

We split prior work into *i.* effort related to the problem of learning object dynamics, *ii.* work about learning contact points for successful pushes, *iii.* work related to the task of the object delivery and *iv.* work on the design of pushing controllers for tasks such as avoidance learning and object singulation.

8.1 Learning dynamics of object motion

Modelling of the dynamics of object motion from observations of push actions was the topic of several robotic studies such as those by Mason (1985), Katz and Brock (2008), Ruiz-Ugalde et al. (2011). Lynch and Mason (1996) presented an analytical model of the dynamics of pushing with a robotic manipulator. They also studied the controllability of pushing with a point contact or a stable line contact. Ruiz-Ugalde et al. (2010) presented a mathematical model for the planar sliding motion. Robot movement is determined based on friction coefficients between the robot hand, the table and a rectangular object.

In contrast, Salganicoff et al. (1993) created one of the first systems that took into account vision feedback for learning a forward model of a planar object for pushing in an obstacle-free environment by controlling only one degree of freedom. Scholz and Stilman (2010) learned object-specific dynamic models for a set of objects which can be pushed from many predefined points on the perimeter. In a similar tabletop scenario, Omrcen et al. (2009) used a neural network for the robot to learn the pushing dynamics for flat objects. They proposed a learning process to describe the relationship between the direction of a poke and the observed object motion for a class of objects. The learned models are specific to individual objects. In recent work, Agrawal et al. (2016) proposed learning physics models with deep neural networks from experience and simultaneous learning of forward and inverse models for vision-based control. This approach shows a high level of performance in predicting the outcomes of pokes. However, it requires large amounts of data and does not generalise to new objects. Kopicki et al. (2017) learned many simple, object-specific and context-specific predictors for rigid

object motions. Their models match the predictions of a rigid body dynamics engine for novel objects as well as actions.

In our setup, we also take a machine learning based approach. However, we learn the inverse model of the robot-object pushing behaviour on the fly. This model can be used to gain experience for data-driven methods. Also it could be combined with forward models, learned from experience, in future work.

8.2 Learning contact points for successful pushes

Walker and Salisbury (2008) modelled support friction maps within a two-dimensional environment using local regression. Similarly, Lau et al. (2011) built mappings between actions and effects using a non-parametric regression method. They collected an hundred samples from interactions of the robot with irregularly-shaped flat objects. Kopicki et al. (2011) established an algorithm for learning object behaviour in the case of various push actions for a tabletop scenario. Hermans et al. (2013a), Hermans et al. (2013b) developed an offline, data-driven method for predicting contact locations that are good for pushing objects, based on visual features.

In contrast, we do not enforce pushing at a specific contact location, but instead change the movement direction and the robot-object displacement, based on observations. The centre of mass is not easily observable, and visual features can be misleading. Some examples are the 'box' and 'cheezit' objects from the object set used. 'Cheezit' appears to be not as easy to push as 'box', due to the content inside, which changes the weight distribution while pushing.

8.3 Algorithms for planning pushing actions

Li and Payandeh (2007) examined the problem of object motion from one position to another with a focus on determining adequate pushing actions and developing a push planner. Recent studies from Elliott et al. (2016) showed that it is possible to plan a sequence of simple pushing and pulling actions to bring an object into a graspable pose. Dogar and Srinivasa (2012) introduced a planning framework for clutter rearrangement in a tabletop scenario. Using linear pushes, a robot can reach desired objects. The planner utilises pushing actions to funnel uncertainty by forming push-grasps. Zito et al. (2012) combined a global RRT planner with a local push planner, making use of predictive models of pushing interactions, to plan sequences of pushes to move an object in an obstacle-free tabletop scenario. The local planner utilized a physics engine that requires explicit object modelling. The state-of-the-art method presented by Mericli et al. (2013), Merili et al. (2015), Meriçli et al. (2015) addressed the problem of pushing complex, massive objects on caster wheels with a mobile base in cluttered environments. Based

on experience from self-exploration or demonstration via joy-sticking, they created experimental models for planning the pushing action using an RRT-based method which does not require any explicit analytical model.

In contrast to the proposed approaches in the literature, we define a reconfigurable pushing corridor and design an algorithm for object delivery in active dynamic environments. Corridors enable usage of the free area for pushing instead of clinging to the path.

8.4 Pushing controllers

To compare the effectiveness of the proposed adaptive motion controller, we also tested controllers proposed by Igarashi et al. (2010) and Hermans et al. (2013c) for pushing unknown objects. Unlike these two pushing controllers, our proposed method adapts to the observed object behaviour and thus achieves higher success rates. Hermans et al. (2013c) assumed that the robot should push in a way that is aligned to the center of the object. Thus, their *centroid-alignment* controller always tries to bring the robot behind the object with respect to the pushing target. However pushing through the center of an object is not always best for pushing a particular object, as can be seen from the change of push activation functions in Fig. 10. Also, this controller is sensitive to changes in the pushing target. If a change in the angle of the pushing direction is high, a robot is not able to move behind the object without pushing it away, as can be seen in Fig. 9e.

A high level of friction and slow movements are assumed by Igarashi et al. (2010) to simplify the interaction model in their work. In practice, this controller causes excessive orbiting of the robot around the object when it is near the goal, which can be seen in Fig. 9e. In terms of blending pushing and relocating movements, our approach is similar. However, we let the robot learn activation functions for pushing and relocation around the object.

In our previous work (Krvic et al. 2016) we introduced an adaptive feedback controller also based on the displacement angle of the robot from the desired object movement direction. Building up on it, in this paper we introduce an adaptive feedforward-feedback pushing controller for pushing unknown objects.

9 Conclusions and future work

This paper offers an investigation of pushing unknown objects in a robotic scenario, without the usage of previous experience. We introduced a multilayer pushing skill, where different subtasks of the object delivery are treated. A pushing corridor was introduced to simplify the pushing task while providing collision-avoidance guarantees under mild assumptions, using the available free space efficiently.

The robot's behaviour adapts, according to feedback from interaction with an object.

Instead of tuning control parameters with respect to an object's class or shape, we proposed identifying the object behaviour by learning. We demonstrate that a high success rate of object delivery can be achieved by online learning for objects that manifest quasi-static properties under pushing. Also, pushing objects that do not fall into this group is possible. The proposed solution adapts the inverse dynamic model of the object-robot interaction. This is achieved by the introduced adaptive feedforward-feedback controller. The controller is adaptive in the sense that it updates the amount of activation as regards pushing or relocating, based on the observed errors. This results in smooth robot movements that adaptively blend pushing and relocation.

The skill presented can be a part of a more complex system operating in complex and dynamic environments. The goal to be achieved and the object to be manipulated are defined by a high-level planner, and the proposed skill is used for the task execution. The algorithm presented is computationally cheap and enables real-time push manipulation. Our results show that it is robust for our set of objects of various shapes in different environments.

The pushing skill can be further improved by taking the object orientation into account, which would be one of the goals of future work. Also, it would be interesting to combine our learned inverse dynamic model with methods which learn forward movement models of an object as well with visual features of the object.

Acknowledgements Open access funding provided by University of Innsbruck and Medical University of Innsbruck. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under Grant Agreement No. 610532, SQUIRREL.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Agrawal, P., Nair, A., Abbeel, P., Malik, J., & Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*. [arXiv:1606.07419](https://arxiv.org/abs/1606.07419).
- Bansal, S., Calandra, R., Xiao, T., Levine, S., & Tomlin, C. J. (2017). Goal-driven dynamics learning via bayesian optimization. *CoRR*. [arXiv:1703.09260](https://arxiv.org/abs/1703.09260).
- Bauzá, M. & Rodríguez, A. (2017). A probabilistic data-driven model for planar pushing. *CoRR*. [arXiv:1704.03033](https://arxiv.org/abs/1704.03033).
- Cehajic, D., gen. Dohmann, P., & Hirche, S. (2017). Estimating unknown object dynamics in human-robot manipulation tasks. In

- 2017 IEEE international conference on robotics and automation (ICRA).
- Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical report, DTIC Document.
- Deisenroth, M. P., Fox, D., & Rasmussen, C. E. (2015). Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2), 408–423.
- Dogar, M., & Srinivasa, S. (2012). A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3), 217–236.
- Elliott, S., Valente, M., & Cakmak, M. (2016). Making objects graspable in confined environments through push and pull manipulation with a tool. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 4851–4858).
- Fink, D. (1997). *A compendium of conjugate priors*. Technical Report, Environmental Statistics, Group Department of Biology, Montana State University.
- Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1), 34–46.
- Guttorp, P., & Lockhart, R. A. (1988). Finding the location of a signal: A bayesian analysis. *Journal of the American Statistical Association*, 83(402), 322–330.
- Hangl, S., Ugur, E., Szedmak, S., & Piater, J. (2016). Robotic playing for hierarchical complex skill learning. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2799–2804).
- Hermans, T., Li, F., Rehg, J., & Bobick, A. (2013a). Learning contact locations for pushing and orienting unknown objects. In *2013 13th IEEE-RAS international conference on humanoid robots (humanoids)* (pp. 435–442).
- Hermans, T., Li, F., Rehg, J. M., & Bobick, A. F. (2013b). Learning stable pushing locations. In *IEEE International conference on development and learning and epigenetic robotics (ICDL-EPIROB)*.
- Hermans, T., Rehg, J., & Bobick, A. (2013c). Decoupling behavior, perception, and control for autonomous learning of affordances. In *2013 IEEE international conference on robotics and automation (ICRA)* (pp. 4989–4996).
- Igarashi, T., Kamiyama, Y., & Inami, M. (2010). A dipole field for object delivery by pushing on a flat surface. In *ICRA* (pp. 5114–5119).
- Katz, D., & Brock, O. (2008). Manipulating articulated objects with interactive perception. In *IEEE international conference on robotics and automation, 2008. ICRA 2008* (pp. 272–277).
- Konidaris, G. D., Kaelbling, L., & Lozano-Perez, T. (2014). Constructing symbolic representations for high-level planning. In *Proceedings of the twenty-eighth conference on artificial intelligence*.
- Kopicki, M., Zurek, S., Stolkin, R., Moerwald, T., & Wyatt, J. L. (2017). Learning modular and transferable forward models of the motions of push manipulated objects. *Autonomous Robots*, 41(5), 1061–1082.
- Kopicki, M., Zurek, S., Stolkin, R., Morwald, T., & Wyatt, J. (2011). Learning to predict how rigid objects behave under simple manipulation. In *2011 IEEE international conference on robotics and automation (ICRA)* (pp. 5722–5729).
- Krivic, S., Ugur, E., & Piater, J. (2016). A robust pushing skill for object delivery between obstacles. In *International conference on automation science and engineering*. IEEE, Fort Worth, Texas.
- Lau, M., Mitani, J., & Igarashi, T. (2011). Automatic learning of pushing strategy for delivery of irregular-shaped objects. In *2011 IEEE international conference on robotics and automation* (pp. 3733–3738).
- Li, Q., & Payandeh, S. (2007). Manipulation of convex objects via two-agent point-contact push. *International Journal of Robotics Research*, 26(4), 377–403.
- Li, X., & Zell, A. (2006). Path following control for a mobile robot pushing a ball. *IFAC Proceedings Volumes*, 39(15), 49–54.
- Lynch, K. M., & Mason, M. T. (1996). Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6), 533–556.
- Mason, M. T. (1985). The mechanics of manipulation. In *Proceedings IEEE international conference on robotics & automation* (pp. 544–548).
- Mason, M. T. (1986). Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3), 53–71.
- Meriçli, T., Veloso, M., & Akin, H. L. (2015). A case-based approach to mobile push-manipulation. *Journal of Intelligent & Robotic Systems*, 80(1), 189–203.
- Merikli, T. A., Veloso, M., & Akin, H. L. (2013). Achievable push-manipulation for complex passive mobile objects using past experience. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '13* (pp. 71–78). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
- Merili, T., Veloso, M., & Akin, H. (2015). Push-manipulation of complex passive mobile objects using experimentally acquired motion models. *Autonomous Robots*, 38(3), 317–329.
- Omrčen, D., Boge, C., Asfour, T., Ude, A., & Dillmann, R. (2009). Autonomous acquisition of pushing actions to support object grasping with a humanoid robot. In *9th IEEE-RAS international conference on humanoid robots, 2009. Humanoids 2009* (pp. 277–283).
- Prankl, J., Aldoma, A., Svejda, A., & Vincze, M. (2015). Rgb-d object modelling for object recognition and tracking. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 96–103).
- Ruiz-Ugalde, F., Cheng, G., & Beetz, M. (2010). Prediction of action outcomes using an object model. In *2010 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1708–1713).
- Ruiz-Ugalde, F., Cheng, G., & Beetz, M. (2011). Fast adaptation for effect-aware pushing. In *2011 11th IEEE-RAS international conference on humanoid robots (humanoids)* (pp. 614–621).
- Saito, T., Kobayashi, Y., & Naruse, T. (2014). Planning of pushing manipulation by a mobile robot considering cost of recovery motion. In *NCTA 2014 - proceedings of the international conference on neural computation theory and applications, part of IJCCI 2014, Rome, Italy, 22–24 October, 2014* (pp. 322–327).
- Salganicoff, M., Metta, G., Oddera, A., & Sandini, G. (1993). A vision-based learning method for pushing manipulation. In *In AAAI fall symposium series: Machine learning in vision: What why and*.
- Scholz, J., & Stilman, M. (2010). Combining motion planning and optimization for flexible robot manipulation. In *2010 10th IEEE-RAS international conference on humanoid robots (humanoids)* (pp. 80–85).
- Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics (intelligent robotics and autonomous agents)*. Cambridge: The MIT Press.
- Walker, S., & Salisbury, J. (2008). Pushing using learned manipulation maps. In *IEEE international conference on robotics and automation, 2008. ICRA 2008* (pp. 3808–3813).
- Yu, K. T., Bauza, M., Fazeli, N., & Rodriguez, A. (2016). 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)* (pp. 30–37).
- Zito, C., Stolkin, R., Kopicki, M., & Wyatt, J. L. (2012). Two-level rrt planning for robotic push manipulation. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 678–685).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Senka Krivic received her B.Eng. and M.Eng. degrees in Electrical Engineering with focus on control theory and electronics from the University of Sarajevo, Bosnia and Herzegovina in 2010 and 2012, respectively. She joined the Intelligent and Interactive Systems group at the University of Innsbruck, Austria, in 2014 as a Ph.D. student and a teaching and research assistant. Her current research interests include robot manipulation, robot learning and planning under uncertainty.



Justus Piater is a professor of computer science at the University of Innsbruck, Austria, where he leads the Intelligent and Interactive Systems group. He holds a M.Sc. degree from the University of Magdeburg, Germany, and M.Sc. and Ph.D. degrees from the University of Massachusetts Amherst, USA, all in computer science. Before joining the University of Innsbruck in 2010, he was a visiting researcher at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany, a professor of computer science at the University of Lige, Belgium, and a Marie-Curie research fellow at GRAVIR-IMAG, INRIA Rhne-Alpes, France. His research interests focus on visual perception, learning and inference in sensorimotor systems. He has published more than 150 papers in international journals and conferences, several of which have received best-paper awards, and currently serves as Associate Editor of the IEEE Transactions on Robotics.