Noname manuscript No.
(will be inserted by the editor)

A Case-Based Approach to Mobile Push-Manipulation

Tekin Meriçli · Manuela Veloso · H. Levent Akın

Received: date / Accepted: date

Abstract The complexity of the potential physical interactions between the robot, each of the pushable objects, and the environment is vast in realistic mobile push-manipulation scenarios. This makes it non-trivial to write generic analytical motion and interaction models or tune the parameters of physics engines for every robot, object, and environment combination. We present a case-based approach to push-manipulation that allows the robot to acquire, through interaction and observation, a set of discrete, experimental, probabilistic motion models (i.e. *probabilistic cases*) for pushable passively-mobile real world objects. These probabilistic cases are then used as building blocks for constructing *achievable* push plans to navigate the object of interest to the desired goal pose as well as to potentially push the movable obstacles out of the way in cluttered task environments. Additionally, incremental acquisition and updating of the probabilistic cases allows the robot to adapt to the changes in the environment, such as increased mass due to loading of the object of interest for transportation purposes. The purely interaction and observation driven nature of our method makes it robot, object, and environment (real or simulated) independent, as we demonstrate through validation tests in a real world setup in addition to extensive experimentation in simulation.

T. Meriçli
Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul, Turkey
E-mail: tekin.mericli@boun.edu.tr

M. Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, United States
E-mail: veloso@cmu.edu

H. L. Akın
Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul, Turkey
E-mail: akin@boun.edu.tr

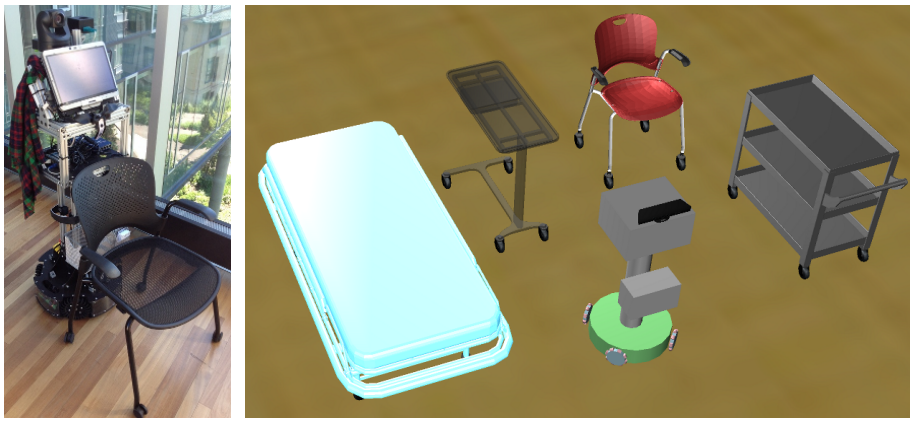
Keywords Mobile push-manipulation · experience-based mobile manipulation · mobile manipulation learning · mobile manipulation planning

1 Introduction

Depending on the requirements of the manipulation task and the constraints imposed by the physical properties of both the robot and the objects, it may not always be possible to utilize prehensile manipulation techniques to solve the problem. In those cases, non-prehensile manipulation approaches [14], such as *push-manipulation*, may be more suitable. This study investigates one such scenario as our omni-directional mobile service robot CoBot [22] (Fig. 1(a)) is not equipped with a manipulator arm. CoBot is expected to push-manipulate a set of passively-mobile objects (Fig. 1(b)) in such a way to transport them to their desired poses while avoiding collisions in task environments cluttered with both stationary and potentially movable obstacles. This is not a trivial problem and it gets even more challenging in our case due to the following facts:

- Our manipulable objects move on passive caster wheels as opposed to sliding on high-friction surfaces and stopping immediately after the pushing is ceased. This introduces additional motion uncertainty, rendering our objects of interest inherently more difficult to push-manipulate.
- Considering the sophisticated contact properties of our objects of interest due to their complex 3D structures, it is neither trivial nor feasible to build analytical interaction and motion models for each and every one of them. Hence, traditional model-based planning approaches will not solve the problem in a flexible and scalable way.

In [16–18], we presented a method that does not require any explicit analytical models for neither the objects nor the robot to be able to construct and



(a) (b)

Fig. 1 (a) CoBot push-manipulating an office chair. (b) Some of the realistically simulated passively-mobile objects and the CoBot used as the pusher.

execute push-manipulation plans. Rather, interactions of the robot with the objects and the resulting observations are used to build object-specific *experimental motion models*, which are stored as *cases* similar to a case-based planning approach [30]. The acquired experimental models are then used as building blocks for constructing safe and *achievable* push-manipulation plans. An experience-based Rapidly-exploring Random Trees (RRT) variant planning algorithm we contributed, namely Exp-RRT, is used to plan collision-free paths for the primary object of interest (OOI), whereas state lattice graphs are utilized to plan how to push the movable obstacles out of the way. The constructed plans are then executed by reiterating the corresponding pushing motions one after another while actively monitoring the execution.

Having acquired experience about how several objects in its task environment move in response to various pushes enables the robot to modify and rearrange the task environment to fit its needs, which can be useful for easing the burden on the future planning tasks. The complexity of the task environments can be reduced considerably by treating every movable object as permeable while constructing push plans for the primary OOI. The results of our experimental evaluation show the advantage of being able to manipulate the obstacles in various scenarios.

We also demonstrate the advantage of having a purely interaction and observation driven case acquisition and tuning approach in adapting the past experience to the novel situations, such as changes in the dynamics of the manipulated objects due to loading/unloading, instead of having to go through the learning process all over for each such novelty. Our experimental results show that, after adaptation, the cases acquired for unloaded OOIs could be successfully used for manipulating loaded ones, which have significantly different dynamics.

The following sections elaborate on the case-based push-manipulation method we introduced in [16,17] and extended in [18] to lay the foundations for the primary contributions of this paper on improving push-manipulation planning and execution performance, mainly through;

- treating the movable objects, which the robot knows how to push-manipulate, as "permeable" in order to be able to construct more relaxed plans for the primary OOI, knowing that those objects could be pushed out of the way,
- continuously updating, in a weighted manner, the stored cases based on the most recent observations in order to be able to adapt to novel situations as well as to achieve incremental improvements in the robustness of plan execution.

2 Related Work

Complex manipulation tasks can be achieved by utilizing the simple mechanics of pushing in cases where the object is too bulky or heavy to lift, or the robot simply lacks a manipulator arm. As a natural result of being one of the most interesting and elegant methods used within the non-prehensile manipulation domain [14,8], the problem of push-manipulation planning and execution has attracted several robotics researchers. Here we review the most recent related studies.

Similar to the potential field based motion planners [9], the algorithm presented in [7] computes dipole-like vector fields around the object to guide the motion of the robot to get behind the object and push it towards the target. Robots with

circular bumpers are used to push circular and rectangular objects of various sizes in single and multi-robot scenarios. Relatively slow robot motions and high friction for the objects are assumed to simplify the model. A promising method for handling objects with more complex shapes is proposed in [11]. A robot with a simple circular shape interacts with irregular-shaped flat objects with quasi-static properties to collect hundreds of samples on how the object moves when pushed from different points in different directions. Using a non-parametric regression method on this data, a mapping between the actions and the effects is built, similar to the approach described in [31]. This method resembles ours as the observations on the object's motion in response to various pushing actions are utilized to model its behavior instead of writing explicit analytical models for that purpose. A global sampling-based planner is combined with a local randomized push planner in [33] to explore various configurations of a polyflap and come up with a series of manipulator actions that will move the object to the intermediate global plan states. The experiments are run in an obstacle-free tabletop setup and the state space is limited to the reach of the robot arm, which is relatively small, as a stationary manipulator with a rigid fingertip is used as the pusher. The randomized local planner utilizes a realistic physics engine to predict the object's pose after a certain pushing action, which requires explicit object modeling. Using the same problem setup, an algorithm for learning through interaction the behavior of the manipulated object that moves quasi-statically in response to various pushes is presented in [10]; however, the learned object behavior is not used for push planning. The observed outcomes of a limited set of four linear and two rotational pushes are used in [24] for planning in a quasi-static tabletop setup where a manipulator arm with a spherical rigid finger push-manipulates objects with simple geometric shapes, ensuring single point of contact.

In scenarios where some of the obstacles in the task environment are movable, it is usually advantageous to take their passive-mobilities into account when planning for the primary object of interest. In handling movable obstacles, the Navigation Among Movable Obstacles (NAMO) framework [26–28] contributes practical algorithms, which globally reason about free space connectivity and identify which objects to move as well as where to move them starting from the very beginning of the planning process. Rigid grasps are used for pushing or pulling the movable obstacles out of the way. The framework presented in [5] leverages push-manipulation for rearranging the clutter in tabletop scenarios via single-step linear pushes to be able to reach and grasp the target object as well as reducing the uncertainty prior to grasping by utilizing the funneling effect of pushing.

From the perspective of learning through interaction and self exploration, our work also resembles various work done in the domain of learning object affordances [29,4]. It also aligns with the MOSAIC model presented in [32,6], which is a general framework that integrates forward models into motor control with the intention to learn inverse models for tasks, and how to select the appropriate inverse model given a certain task.

According to our survey of the literature, the most common push-manipulation scenarios seem to involve pushing of objects with primitive geometric shapes using circular or point-sized robots, or rigid fingertips on surfaces with relatively high friction that makes the object stop immediately when the pushing motion is ceased. Even then relatively complex mathematical models are used for contact modeling and motion estimation, or physics engines of simulators are utilized for

these purposes. Regarding planning among movable obstacles, the most relevant studies either perform push-manipulation in quasi-static tabletop scenarios or utilize grasping for moving the obstacles out of the way. Our approach differs from many of these proposed ones as follows:

- The 3D real world objects we handle present complex contact properties as they may contact the robot on various points (Fig. 2).
- We cannot assume that our objects would come to a stop immediately after a push since they move on passive caster wheels, which contribute to their motion uncertainty.
- We perform *mobile* push-manipulation in a large-scale environment cluttered with static and movable obstacles, requiring construction and execution of safe and *achievable* push plans.
- We do not build any explicit analytical models or learning based mappings; we only utilize experimental models acquired by observing the effects of various pushes for planning and execution.
- Our approach allows push-manipulation among movable obstacles as our robot can push them out of the way in case a potential collision is anticipated during the manipulation of the primary OOI.
- Our robot can adapt the acquired experimental models to the observed changes in the environment, such as changes in the dynamics of the OOI due to loading/unloading, hence apply past experience to solve novel problems quickly.

3 Experience-based Push-Manipulation

Robots should ideally learn and further sharpen their manipulation and corresponding prediction-based planning skills by interacting with their environment and observing the outcomes as it is not trivial, efficient, and scalable to define analytical interaction and motion models for each and every object that we expect the robot to manipulate. For these reasons, we let our robot experiment with the pushable objects either through self-exploration or demonstration via joysticking, and observe how they move in response to various pushes. These observations are

(a) (b) (c)

Fig. 2 The object may contact (indicated by a red arrow) the robot on its (a) body, (b) basket, or (c) base, or a combination of these depending on its 3D structure and the pushing direction, making it non-trivial to model explicitly.

then turned into *experimental models*, or *cases* analogous to a Case-Based Reasoning/Planning (CBR/P) system [1,25], to be used for planning and execution.

Our algorithm consists of the following components:

- A set of object-specific *probabilistic cases* represented as *sequences* composed of the robot's motion commands, its resulting active trajectory, and the object's corresponding passive trajectory, together with a distribution representing the uncertainty in the object's motion,
- A *generative planner* that makes use of these probabilistic cases as building blocks to construct achievable and collision-free push plans,
- An *execution monitoring module* to cease execution and trigger re-planning whenever there is a significant discrepancy between the expected and the actual motion of the object during plan execution.

3.1 Object-Specific Probabilistic Cases

Each individual interaction of the robot with the OOI is stored as a *sequence* of pose-action pairs for the robot and the corresponding poses for the OOI, representing their active and passive trajectories, respectively. A static global frame of reference, φ_G, is attached to the environment. The poses of the robot and the OOI within φ_G are denoted as φ_R and φ_O, respectively. We also define an auxiliary frame of reference, φ_S, to indicate the last stationary pose of the OOI before it starts being pushed (Fig. 3(a)).

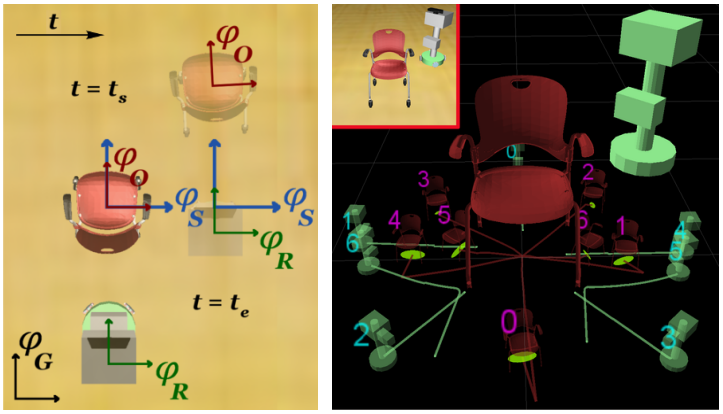
Let $\wp_R$ be φ_R w.r.t. φ_O, and $\wp_O$ be φ_O w.r.t. φ_S, both of which are denoted as $\langle x, y, \theta \rangle$. Invariance to φ_O is achieved by recording $\wp_R$ together with the motion command at that moment and the corresponding $\wp_O$. Therefore, a sequence S_i of length l in a set of sequences $\mathbf{S_o}$ associated with an OOI takes the form,

$$S_i : ((\wp_{R_0}, a_0, \wp_{O_0}), \ldots, (\wp_{R_{l-1}}, a_{l-1}, \wp_{O_{l-1}})), i \in [1, |\mathbf{S_o}|]$$

where a_j is the action associated with $\wp_{Rj}$, denoted as $\langle v_x, v_y, v_\theta \rangle$ indicating the omni-directional motion command composed of the translational and rotational velocities of the robot. Fig. 3(b) provides the visualization of the robot and object trajectories within the stored sequences. The transparent, scaled-down robot figures indicate the push initiation poses (i.e. $\wp_{R_0}$ of each sequence) whereas the scaled-down object figures indicate the mean observed poses of the object after the pushes (i.e. $\wp_{O_{l-1}}$ of each sequence). The robot trajectory (indicated by green curves) and the object trajectory (indicated by red curves) that belong to the same sequence are marked with the same ID value. Final object pose uncertainty is depicted with the yellow ellipses drawn around the mean final poses. This is what makes our cases *probabilistic* ones as the output has uncertainty. The process of acquiring these experimental models is elaborated in Section 3.2.

3.2 Experimentally Acquiring Probabilistic Cases

When the robot is to acquire experience about a given pushable object through self-exploration, it determines m random push initiation locations $\wp_{R_0}$ relative to and immediately around the object together with the corresponding random pushing durations τ ranging from 1 to 3 seconds. We name these tuples *push*



(a) (b)

Fig. 3 (a) Various reference frames used during sequence recording and replay depicted before ($t = t_s$) and after ($t = t_e$) a push. (b) Visualization that corresponds to the scene shown in the upper left corner of the image. The robot trajectory and the corresponding object trajectory components of 7 different probabilistic cases are illustrated.

configurations, $\varsigma = \{\varsigma_0, \ldots, \varsigma_{m-1}\}$, $\varsigma_i : (\wp_{R_0}, \tau)$, which are used to carry out the premier pushes on the OOI. On the other hand, we can also demonstrate the robot more sophisticated and informative pushing motions for the given objects via the use of a joystick. In either case, a new sequence is created and saved whenever the robot tries a particular push for the first time. The additional practices are merely reiterating of these newly learned sequences to update the parameters of the distributions associated with each of them to represent the uncertainty in the observed final pose of the relevant object after a push. We approximate those distributions with 3-dimensional Gaussians based on the analysis of the actual relative final pose data of one of the sequences logged during push execution [18].

After each reiteration of S_i^{new} for modeling uncertainty, the corresponding distribution parameters are incrementally updated according to Eq. 1 and Eq. 2, based on our Gaussian approximation.

$$\bar{\wp}_{O_t^f} = \bar{\wp}_{O_{t-1}^f} + \frac{\wp_{O_t^f} - \bar{\wp}_{O_{t-1}^f}}{t} \tag{1}$$

$$\Sigma_{\wp_{O_t^f}} = \frac{(t-1)\Sigma_{\wp_{O_{t-1}^f}} + (\wp_{O_t^f} - \bar{\wp}_{O_t^f})(\wp_{O_t^f} - \bar{\wp}_{O_{t-1}^f})^T}{t} \tag{2}$$

In these equations, $\bar{\wp}_{O_t^f}$ denotes the mean of the observed final object pose after the t^{th} trial for a specific S_i^{new}, where $t \in [1, n]$, and $\Sigma_{\wp_{O_t^f}}$ is the corresponding covariance, which in our case is a 3×3 matrix as we are dealing with 3 DoF poses in the form of $\langle x, y, \theta \rangle$. This compact representation eliminates the need for storing all of the previously observed individual poses.

Similarly, it would be possible to update the object trajectories themselves by aligning them with the recently experienced ones using a Dynamic Time Warping (DTW) [23] approach, and computing the mean object trajectory accordingly.

However, since the object trajectories are relatively short and the obstacles in the environment are large, it is plausible to assume that the trajectories observed at the very beginning of the learning process are decent representatives of the future ones. This assumption simplifies the computational complexity of the proposed method while not degrading the overall performance of the system significantly.

3.3 Achievable Push-Manipulation

The experimentally acquired probabilistic cases present primitives that can be used for building various kinds of graphs to search for the path to the requested goal pose in the task environment. In order to maintain a small computational footprint, we take a sampling-based approach to planning the push-manipulation path for the primary OOI. We modify the original Rapidly-exploring Random Tree (RRT) algorithm [12,13] and use the previously acquired probabilistic cases as building blocks for constructing the tree [17,18]. Since the probabilistic cases encode the motion of both the robot and the OOI together with the associated uncertainty, using them to construct the push plan ensures *achievability* from both the robot's and the OOI's perspective in terms of motion feasibility and collision-safety. Our contributed experience-based RRT (Exp-RRT) algorithm is given in Algorithm 1.

Algorithm 1 The Exp-RRT algorithm.

function BUILDEXPRRT(ooi, q_{init}, q_{goal})
  $Tree \leftarrow q_{init}$; ▷ set the root of the tree as the initial configuration
  $q_{new} \leftarrow q_{init}$;
  while sim(q_{new}, q_{goal}) < THRESHOLD **do** ▷ while the goal is not reached
    $q_{rand} \leftarrow$ Sample(); ▷ sample a random configuration
    $q_{most\_similar} \leftarrow$ MostSimilar($Tree, q_{rand}$);
    $q_{new} \leftarrow$ Extend($ooi, q_{most\_similar}, q_{rand}$);
    $Tree$.add(q_{new}); ▷ add the new configuration to the tree
  end while
  return Trajectory($Tree, q_{new}$); ▷ return solution
end function
function MOSTSIMILAR($Tree, q_{target}$) ▷ find "most similar" configuration
  return $\arg\max_{q \in Tree}$ sim(q, q_{target});
end function
function EXTEND($ooi, q_{source}, q_{target}$) ▷ extend using the best sequence
  $S_{trans} \leftarrow \{\text{Transform}(S_i, q_{source})\} \forall S_i \in ooi.S$;
  $S_{safe} \leftarrow \{S_{trans} \setminus \{\text{Colliding}(S_t)\}\} \forall S_t \in S_{trans}$;
  return $\arg\max_{S_i.q^f \forall S_i \in S_{safe}}$ sim($S_i.q^f, q_{target}$);
end function

At each iteration, we sample a random pose, $\langle x, y, \theta \rangle$, with probability p or use the goal as the sample with probability $1 - p$. The *closest* node of the tree to the new sample is the one that gives the maximum similarity value according to the similarity function defined in Eq. 3,

$$sim(p_1, p_2) = \frac{d_{max}}{dist(p_1, p_2)} cos(p_1.\theta - p_2.\theta) \tag{3}$$

where d_{max} is the maximum possible distance that can be obtained in the task environment and $dist(p_1, p_2)$ is the Euclidean distance between the locations of the poses. Therefore, the closer the locations of the two poses and the smaller the angular difference between their orientations, the more similar they are. After the closest node to the sample is determined, imagining the OOI to be on the pose of the closest node, this time the final expected poses of the sequences originating from that imaginary pose are checked against the sample according to the similarity function defined in Eq. 3. The tree is extended towards the sample by using the final projected object pose of the sequence that gives the highest similarity value and is collision-free for both the object and the robot. This process is repeated until the pose of the newly added node falls within predefined distance and orientation difference limits to the goal pose.

Taking uncertainty into account during RRT planning has been studied in the literature [15,2]; however, we do it for the manipulated objects instead of the robot itself in addition to performing it in a novel way. For this purpose, we derive $2L+1$ sigma points representing the extremes of the final pose distributions from the mean and the covariance using Eq. 4-6, where L is the dimensionality of the state space. In our case $L = 3$ as we are dealing with 3 DoF poses.

$$\chi_0 = \bar{\wp}_{O^f} \tag{4}$$

$$\chi_i = \bar{\wp}_{O^f} + \zeta(\sqrt{\Sigma_{\wp_{O^f_n}}})_i, i = 1, \ldots, L \tag{5}$$

$$\chi_i = \bar{\wp}_{O^f} - \zeta(\sqrt{\Sigma_{\wp_{O^f_n}}})_i, i = L+1, \ldots, 2L \tag{6}$$

In these equations, $\bar{\wp}_{O^f}$ is the mean of the final OOI poses observed so far for a particular sequence, $(\sqrt{\Sigma_{\wp_{O^f_n}}})_i$ is the i^{th} column of the matrix-square-root of the covariance matrix $\Sigma_{\wp_{O^f_n}}$, and ζ is the scalar scaling factor that determines the spread of the sigma points around $\bar{\wp}_{O^f}$. Increasing ζ increases the conservativeness of the planner. That is, since the push plans are constructed out of the sequences, and the corresponding distributions as well as the object and robot trajectories are used for collision checking, the value of ζ has an indirect effect on adjusting the minimum allowed distance of the planned paths from the obstacles. In our experiments, we used $\zeta = 3$. Each of the extreme poses represented by the sigma points are checked for collision and the case is marked as *unachievable* and not used for extending the tree if any of these poses are in collision with the objects in the environment (Fig. 4). After the push plan is constructed for the OOI, during plan execution, a standard RRT planner is used for moving the robot to $\wp_{R_0}$ of the corresponding robot trajectories of the sequences along the push plan.

The constructed push plan is executed by replaying one after another the robot trajectories of the chain of sequences that transports the OOI to the desired pose. Even though the plan is constructed by taking into account the uncertainties in the expected final OOI poses, the object inevitably digresses from its foreseen path, especially when it needs to be transported for a long distance. In such cases, re-planning is triggered if the result of the hypothesis test indicates statistically significant difference between the expected object pose and the observed one. [1]

[1] A video showing the simulated robot push-manipulating an overbed table can be seen here: `http://youtu.be/rN22PSjsniY`

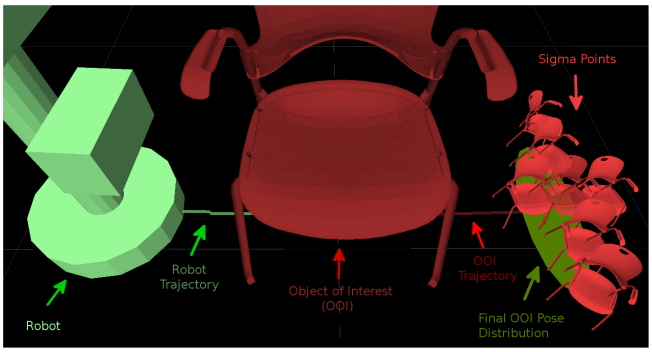


Fig. 4 The *sigma points*, illustrated as scaled-down OOI figures, indicate the extreme poses (i.e. the extreme points on the corresponding distribution) that could be observed after a push.

4 Push-Manipulation Among Movable Obstacles

The robot may take advantage of its experience about the other pushable objects in the environment to ease the burden of manipulation planning and execution for the primary OOI. As the robot would know how to push-manipulate them, all the known movable objects in the task environment can essentially be treated as *permeable* during push plan construction for the primary OOI, allowing the solution paths to pass through them. Before executing each push along the constructed push plan, the robot checks whether the action would potentially result in a collision with any of the movable obstacles. In case a collision is anticipated, a state lattice graph [20,21] is expanded for each of the involved movable obstacles. Since we do not know where to push the obstacle ahead of time, full graph construction until candidate collision-free configurations are found is a more suitable approach than using Exp-RRT, which would require a particular goal to be specified. Full planning graphs are expanded by using the relevant probabilistic cases as building blocks until collision-free poses for the movable obstacles are reached.

Fig. 5 illustrates a single branch expansion in the process of state lattice construction for the overbed table object. As opposed to various applications of the state lattices in the literature, we have no concerns regarding the continuity of the successive motion primitives as our primitives represent discrete achievable motions of the OOI, not the robot. Therefore, none of the cases are eliminated during graph construction, except for the ones that are in collision with the environment. Among the set of candidate paths that first reach collision-free configurations, the one that pushes the obstacle farthest from the OOI's path as well as all the other obstacles is selected for execution. The push plan for the OOI is ceased until all the immediately blocking movable obstacles are pushed out of the way according to the generated plan, and it is resumed when the primary path is clear. Fig. 6 shows the stages of movable obstacle clearance during push-manipulation of a stretcher. [2] Even though we cannot claim that our approach to handling push-manipulation among movable obstacles produces optimal plans and results, our experiments demonstrate its practicality as we never observed a failure in our test scenarios.

[2] A video showing experience-based push-manipulation among movable obstacles can be seen here: `http://youtu.be/8YSjWfMuJZo`

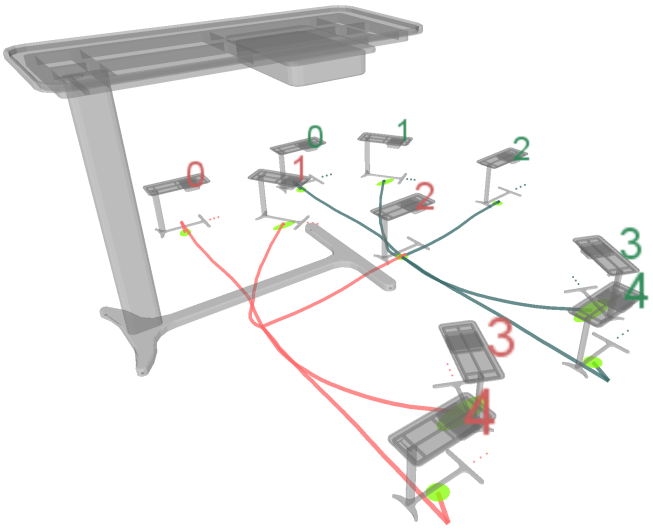
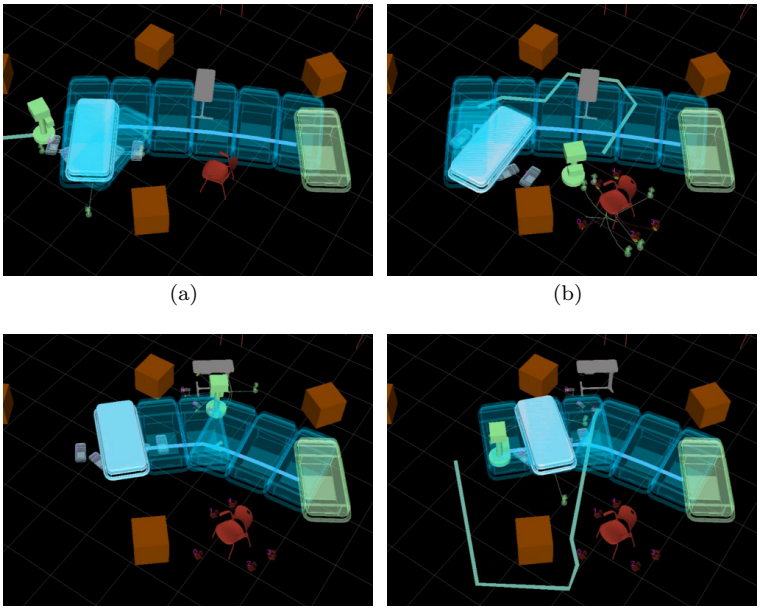


Fig. 5 The construction of the state lattice is achieved by repeating the set of probabilistic cases at the end of every individual case in a breadth-first manner.



(a) (b)

(c) (d)

Fig. 6 Clearing movable obstacles along the path of the stretcher. (a) First the stretcher is push-manipulated until a collision is anticipated. (b) The chair is pushed out of the way. (c) The overbed table is cleared after resuming the original push plan and navigating the stretcher a little further. (d) The original push plan for the stretcher is resumed.

5 Adapting Past Experience to Novel Situations

After the initial learning period, the acquired probabilistic cases are usually reliable enough to push-manipulate the known OOIs within the environment successfully, requiring re-planning every once in a while during task execution. However, especially considering that all of our potential OOIs are instances of office, hospital, and service furniture, their occupation state, and hence dynamics, may change from time to time as they are mainly intended for transporting various loads. Fig. 7 depicts such a scenario, where the overbed table is loaded with several objects of unknown masses. This addition to the object changes its dynamics and it starts behaving differently than expected, as illustrated in the figure. Even in such cases where the OOIs are loaded with other objects, we would still like to be able to make as much use of the past experience as possible to solve the novel problem without having the go through the learning process all over again.

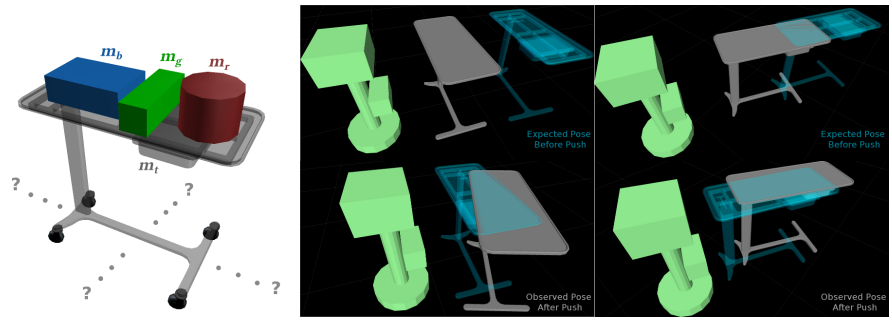


Fig. 7 A large difference is observed between the expected pose and the realized one when the loaded OOI is tried to be push-manipulated using the past experience obtained from the unloaded OOI. However, the robot can incrementally adapt the corresponding cases to these new observations instead of trying to learn a completely new set of cases for the loaded OOI.

To achieve that, we seek a way of modifying the available distributions to quickly accommodate for the difference between the expected and the observed object behavior. However, since the parameters of the distributions associated with individual cases are updated incrementally after each trial during case acquisition, the latest observations start becoming less and less effective in shaping the distributions as the number of practices per case increases. We overcome this problem by introducing a weighting factor to determine how much relative weight the distributions representing the past experience should have over the newly acquired experience. This weight coefficient is computed by evaluating the difference between the expected and the observed pose of the object via the pose similarity function defined in Eq. 3 and multiplying the obtained value with a scaling factor K, as defined in Eq. 7. In our experiments, we set the value of K to be equal to the number of initial practices for each of the sequences, that is $K = 20$. The obtained weight is used for adjusting the influence of the past distribution defined by $\bar{\wp}_{O^f_{t-1}}$ and $\Sigma_{\wp_{O^f_{t-1}}}$, as shown in Eq. 8 and Eq. 9. That is, if the difference between the expected pose and the observed one is significant, then their similarity value will be small, resulting in a decrease in the weight of the past distribution parame-

ters, allowing them to quickly shift towards the new observation. Otherwise, these updates will have a similar effect as the ones performed by the original equations given in Eq. 1 and Eq. 2.

$$W = sim(p_{expected}, p_{observed})K \tag{7}$$

$$\bar{\wp}^{W}_{O^f_{t-1}} = W\bar{\wp}_{O^f_{t-1}} \tag{8}$$

$$\Sigma^{W}_{\wp_{O^f_{t-1}}} = W\Sigma_{\wp_{O^f_{t-1}}} \tag{9}$$

In addition to handling drastic changes in the OOI's dynamics, it is also possible to utilize the flexibility of our incremental learning approach to enable the robot to continuously learn and adapt by updating the parameters of the corresponding distributions after each push during regular task execution.

6 Experimental Evaluation

We performed the majority of our experiments in Webots simulation environment [19] while validating the feasibility of the underlying method in a real world scenario [18] (Section 6.1.1). The final placement of an OOI was considered successful if the distance of the object to the desired goal was below $0.2m$ and the orientation difference was below $\pi/9$ radians. These constraints are quite tight considering the dimensions of the objects that our robot is manipulating, such as a $0.8m \times 0.45m$ utility cart and a $1.9m \times 0.9m$ stretcher manipulated in a $15m \times 15m$ environment. Given the acceptable final location and orientation difference thresholds, if we were to construct an $\langle x, y, \theta \rangle$ grid over the entire task environment, we would have a total of $(15.0/0.2)^2$ $\langle x, y \rangle$ locations with 9 different orientation values at each location. Therefore, the maximum number of Exp-RRT nodes was set to be $9 * (15.0/0.2)^2 = 50625$ based on the task constraints. We sampled a random pose with probability $p = 0.05$ and the goal pose with probability $1 - p = 0.95$.

It must be noted that the simulation environment is essentially a black box for the robot, as the real world would be, and the only information that the simulator provides to the robot is the poses of the objects. It is a black box for us as well. In addition to the object meshes for visualization, we only provide empirically determined mass values and wheel friction coefficients for the ODE physics engine of the simulator to take care of the inter-object interactions to make them behave reasonably realistically. The only motivation behind using a realistic 3D simulator is to obtain a setup that "looks" an "behaves" reasonably realistically as we are not concerned with transferring any knowledge from the simulated environment to the real world or vice versa. In other words, both the internal and external parameters of the simulator are totally irrelevant to the operation of our method. Therefore, even if we had not set the physics parameters realistically, the method would still work and learn how to push-manipulate objects under those circumstances, which makes it totally independent of the robot, the object, and the environment (simulated or real).

6.1 Evaluation of the Underlying Method

The aim of our first experiment was to understand how the number of practices per sequence (corresponding to the n in Eq. 1 and Eq. 2) affects the robustness of the constructed push plan, measured in the number of re-plans performed during execution. We used the chair as the primary OOI and utilized the set of 7 sequences illustrated in Fig. 3(b). The desired goal pose was approximately $4m$ away diagonally, which corresponds to an average of 12 pushes with the available set of cases. Starting from a minimum of 5 practices per sequence, with an increment of 5, we tried up to 20 practices per sequence, and performed 10 planning and execution experiments with each of these values. Fig. 8 shows the results, where a general tendency (indicated with the green curve) of decreasing number of re-plans can be observed. Given the average path length of 12, almost 3 consecutive pushes can be achieved before re-planning in case of 20 practices per sequence, whereas that number is below 2 for the case of 5 practices per sequence.

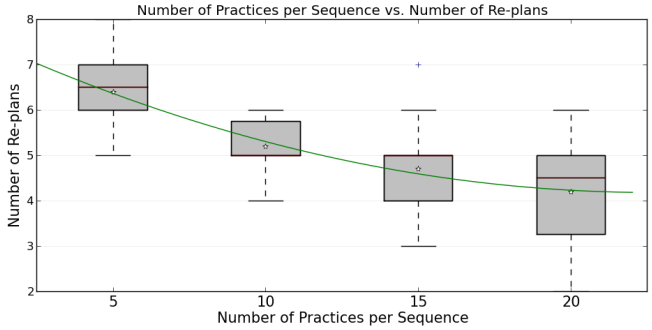


Fig. 8 The number of re-plans during task execution decreases with the increasing number of practices per individual case. This tendency illustrated by the green curve is an indicator of the corresponding distributions becoming more stable and reliable in their predictive abilities.

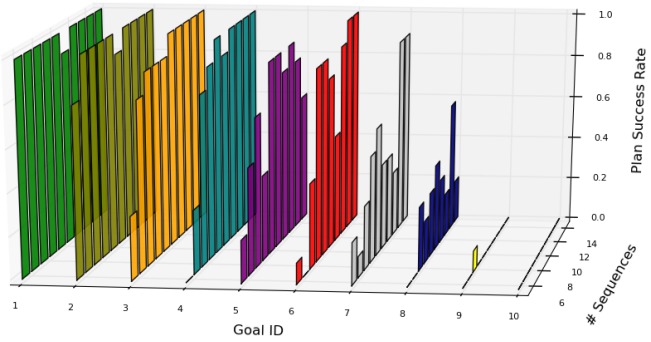


Fig. 9 The number of cases versus plan success rate for 10 random goals. The general tendency is towards an increased plan success rate with the increasing number of cases.

The second experiment we performed was to see the effect of the number, hence the variety, of the cases in solving various planning problems. For that purpose, we created 10 random goal configurations scattered around the cluttered task environment. Again using the chair as the primary OOI, we started from 5 cases (i.e. sequences) and went all the way up to 14 cases with increments of 1. Artificially limiting the maximum number of allowed Exp-RRT nodes to 2000, for each number of cases, we performed 10 planning experiments for each of the 10 random goals and measured whether the planner succeeded (1 for success and 0 for failure) and if so how many nodes it had to expand before reaching the goal. We obtained a success rate value by counting the number of successes after each set of 10 experiments per goal and dividing the sum to the number of experiments (i.e. 10). Fig. 9 illustrates how the plan success rate changes for each of the 10 goals with the increasing number of cases. The figure shows a general tendency towards an increasing success (i.e. reachability) rate for each goal as the number of cases increases. Out of the last two goals where the robot performed the poorest in terms of planning, one of them was at the opposite corner of the environment with many obstacles in between, and the other one was within close proximity of another obstacle, which made it difficult for the robot to generate a collision-free placement plan for these goal configurations.

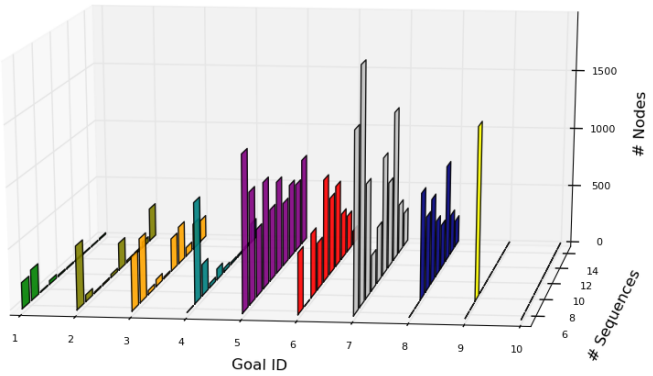


Fig. 10 The number of cases versus the number of expanded Exp-RRT nodes. The general tendency is towards a reduced number of Exp-RRT nodes with the increasing number of cases.

Fig. 10 visualizes the second measurement in the same experiment scenario, which is the average number of expanded Exp-RRT nodes during the 10 experiments performed for each of the goals. It can be observed that, for each goal, the number of nodes tends to decrease with the increasing number of cases, which means that a solution can be found quicker with a richer set of cases.

6.1.1 Real World Validation

In order to validate our underlying method in terms of its robot, object, and environment (i.e. real or simulated) independence, we also ran a preliminary real world test in addition to the extensive experimentation we performed in simulation [17, 18]. In this test, our CoBot robot [22] was asked to arrange a set of chairs in a

desired seating formation around a round table. Due to the abundance of uncertainty in the real world, it becomes difficult to construct a stable world model. In the physical setup, the robot's global pose information comes from the localization module [3] and the OOI is perceived through the Augmented Reality (AR) tags placed on it, both of which result in noisier information compared to the perfect ones received in simulation. It also becomes important to employ good trackers so that the robot can still have an idea of where the object is even if it is not visible within the robot's field of view.

During our preliminary tests, the robot was, in general, able to construct a relatively stable world model by combining its perception with its localization information to generate and execute push-manipulation plans [3]. Even though we have not performed detailed experiments in this setup, we observed that there was an overall increase in the frequency of re-planning due to the increased uncertainty in both perception and action in real world. These preliminary tests verified the validity of our method and demonstrated its robot, object, and environment independence as the exact same codes are run on the physical robot as on the simulated one. The only pieces of information used were the robot's localization belief and the pose of the object of interest that the robot itself extracted from the perceived AR tags.

6.2 Handling Movable Obstacles

We ran two sets of experiments to evaluate the benefits of the method we propose for handling and utilizing the movable obstacles in the task environment.

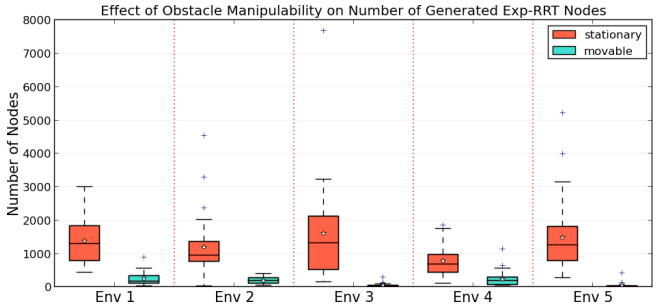
Table 1 Task completion statistics for stationary and movable obstacle configurations.

Obstacle configuration	μ (sec)	σ (sec)	# replans
Stationary (Scenario 1)	187.5	47.33	8
Movable (Scenario 2)	92.05	17.29	5

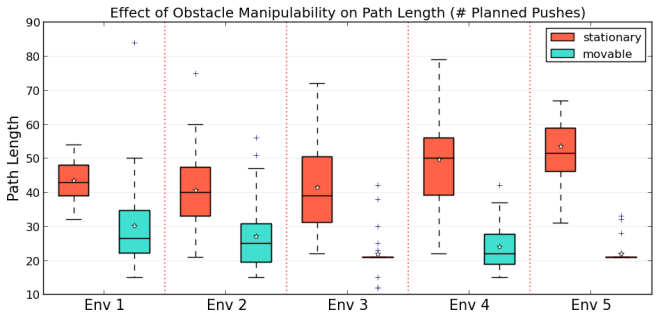
Initially excluding the execution part, we first tried to understand how such capability influences planning performance. We created 5 random test environments in simulation and ran 30 planning trials in two different scenarios. In Scenario 1, all the obstacles were treated as stationary whereas in Scenario 2, some of the obstacles were considered as movable. We observed that planning time is considerably improved in Scenario 2 as a result of many of the planning constraints being eliminated. As Fig. 11 shows, both the means and the variances of the various performance metrics obtained in Scenario 2 (cyan) are considerably lower than those obtained in Scenario 1 (red). The results are intuitive since the planner needs more effort to find potentially more tortuous solution paths that avoid all the obstacles in Scenario 1, which explains longer planning times and path lengths.

In the second set of experiments, we additionally tested the task completion times (i.e. both planning and execution) in a random test environment by running 10 trials for each of the two scenarios. Table 1 shows a significant difference between

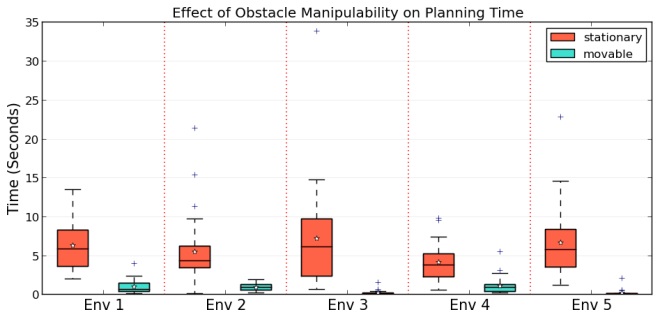
[3] A video of the real robot performing experience-based push manipulation [17,18] can be seen here: `http://youtu.be/TORQdBPHJ3g`



(a) Total number of Exp-RRT nodes.



(b) The length of the solution path.



(c) Planning time.

Fig. 11 Planning performance in the presence of movable obstacles measured in terms of the total number of Exp-RRT nodes, the solution path length, and planning time.

the average task completion times as well as the number of re-plannings during the navigation of the primary OOI, which, in this case, was an overbed table.

6.3 Continuous Adaptation and Experience Transfer

Our last experiment investigates how the ability to continuously adapt to the changes regarding the primary OOI (e.g. loaded versus unloaded) affects robust plan execution measured in the number of consecutive pushes before a re-plan is needed. The simulated overbed table was used as the primary OOI and the desired

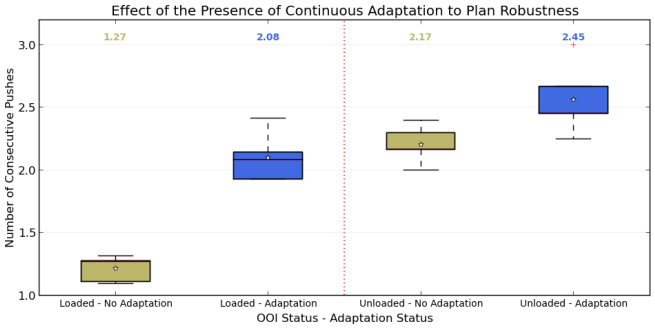


Fig. 12 Continuous adaptation to the observed changes contributes significantly to the robust execution of the generated plans, both under occupation state changes (left) and during regular operation (right). Median number of consecutive pushes are given at the top.

goal pose was placed approximately $10m$ away diagonally, which corresponds to an average of 25 pushes with the available set of cases. The cases were acquired on the unloaded overbed table, and the robot was expected to adapt and utilize them for push-manipulating the loaded one, which has significantly different dynamics as shown in Fig. 7. Adaptation was performed on the fly during plan execution with the hope that the execution would become more robust after each re-planning attempt. Fig. 12 clearly shows that continuous adaptation enables the cases learned on an unloaded OOI to be adapted and used on a loaded one (the left part of the plot) as well as improving the performance during plan execution in the original unloaded OOI manipulation scenario (the right part of the plot). The first box in the plot shows that the robot pretty much had to re-plan after each push when it tried manipulating a loaded overbed table using the cases acquired on the unloaded one. The second box depicts the performance improvement in the case of continuous adaptation, where an average of more than 2 pushes could be achieved by adapting the cases to the novel situation. This is noteworthy considering that the robot had to adapt to the new dynamics on the fly in the course of an average of just 25 pushes in total. It was observed that the robot was able to adapt a particular probabilistic case to a novel situation after 2 iterations on average.

7 Conclusion

The vast complexity of the potential physical interactions between the robot, each of the pushable objects, and the environment in realistic push-manipulation scenarios makes it non-trivial to write generic analytical motion and interaction models or tune the parameters of physics engines for every possible robot, object, and environment combination. We address this problem by introducing a case-based push-manipulation approach that enables the robot to acquire, through interaction and observation, a set of discrete, experimental, probabilistic motion models (i.e. probabilistic cases) for pushable passively-mobile real world objects. The robot then constructs achievable push plans out of these acquired probabilistic cases to navigate the object of interest to the desired goal pose as well as to potentially push the movable obstacles out of the way in cluttered task environments. Additionally,

incremental acquisition and updating of the probabilistic cases allows the robot to adapt to the changes in the dynamics of the objects when continually used after the initial case acquisition phase. As demonstrated through some preliminary tests in a real world setup in addition to extensive experimentation in simulation, our method proves to be robot, object, and environment (real or simulated) independent due to its purely interaction and observation driven nature.

Even though we present a relatively rich set of experimental results in various task setups and scenarios, there still remains a lot of room for further research as well as improvements to the present method. Potential future work includes;

- incorporating Dynamic Time Warping based alignment and adjustment of the successively observed object trajectories to track variance not only over the final object pose but also along the trajectories,
- investigating the potential benefits of propagating uncertainty through the plan graph over our current optimistic approach of keeping the distributions of consecutive push results independent from each other,
- exploring the construction of deterministically optimal paths for the primary object of interest through the utilization of a state lattice planner and comparing the performance against the Exp-RRT,
- performing skill transfer among different objects with similar physical properties in addition to adapting to the changes occurring on the same object,
- evaluating the presented methods through extensive testing and detailed experimentation in a physical setup in the presence of movable obstacles and varying object dynamics.

References

1. Bartsch-Spörl, B., Lenz, M., Hübner, A.: Case-Based Reasoning - Survey and Future Directions. In: Proceedings of the 5th German Biennial Conference on Knowledge-Based Systems, pp. 67–89. Springer Verlag (1999)
2. Berg, J.V.D., Abbeel, P., Goldberg, K.: LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. In: Proceedings of Robotics: Science and Systems (RSS). Zaragoza, Spain (2010)
3. Biswas, J., Coltin, B., Veloso, M.: Corrective Gradient Refinement for Mobile Robot Localization. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2011)
4. Şahin, E., Çakmak, M., Doğar, M.R., Uğur, E., Üçoluk, G.: To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems **15**(4), 447–472 (2007). DOI 10.1177/1059712307084689. URL `http://dx.doi.org/10.1177/1059712307084689`
5. Dogar, M., Srinivasa, S.: A Planning Framework for Non-Prehensile Manipulation under Clutter and Uncertainty. Autonomous Robots **33**(3), 217–236 (2012)
6. Haruno, M., Wolpert, D.M., Kawato, M.M.: MOSAIC Model for Sensorimotor Learning and Control. Neural Computation **13**(10), 2201–2220 (2001). DOI 10.1162/089976601750541778. URL `http://dx.doi.org/10.1162/089976601750541778`
7. Igarashi, T., Kamiyama, Y., Inami, M.: A Dipole Field for Object Delivery by Pushing on a Flat Surface. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2010)
8. K. M. Lynch and M. T. Mason: Dynamic nonprehensile manipulation: Controllability, planning, and experiments. International Journal of Robotics Research **18**, 64–92 (1997)
9. Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research **5**(1), 90–98 (1986)

10. Kopicki, M., Zurek, S., Stolkin, R., Mörwald, T., Wyatt, J.: Learning to predict how rigid objects behave under simple manipulation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2011)
11. Lau, M., Mitani, J., Igarashi, T.: Automatic Learning of Pushing Strategy for Delivery of Irregular-Shaped Objects. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2011)
12. LaValle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. Tech. rep., Computer Science Department, Iowa State University (1998)
13. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
14. Lynch, K.M.: Nonprehensile Robotic Manipulation: Controlability and Planning. Ph.D. thesis, Robotics Institute, Carnegie Mellon University (1996)
15. Melchior, N., Simmons, R.: Particle RRT for Path Planning with Uncertainty. In: 2007 IEEE International Conference on Robotics and Automation, pp. 1617–1624 (2007)
16. Meriçli, T., Veloso, M., Akın, H.L.: Experience Guided Achievable Push Plan Generation for Passive Mobile Objects. In: Beyond Robot Grasping - Modern Approaches for Dynamic Manipulation, IROS'12 (2012)
17. Meriçli, T., Veloso, M., Akın, H.L.: Achievable Push-Manipulation for Complex Passive Mobile Objects using Past Experience. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2013)
18. Meriçli, T., Veloso, M., Akın, H.L.: Push-Manipulation of Complex Passive Mobile Objects using Experimentally Acquired Motion Models. Autonomous Robots pp. 1–13 (2014). DOI 10.1007/s10514-014-9414-z. URL http://dx.doi.org/10.1007/s10514-014-9414-z
19. Michel, O.: Webots: Professional Mobile Robot Simulation. Journal of Advanced Robotics Systems **1**(1), 39–42 (2004)
20. Pivtoraiko, M., Kelly, A.: Constrained Motion Planning in Discrete State Spaces. In: Field and Service Robotics, pp. 269–280 (2005)
21. Pivtoraiko, M., Kelly, A.: Efficient Constrained Path Planning via Search in State Lattices. In: Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (2005)
22. Rosenthal, S., Biswas, J., Veloso, M.: An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2010)
23. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. Acoustics, Speech and Signal Processing, IEEE Transactions on **26**(1), 43–49 (1978)
24. Scholz, J., Stilman, M.: Combining motion planning and optimization for flexible robot manipulation. In: Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, pp. 80–85 (2010)
25. Spalazzi, L.: A Survey on Case-Based Planning. Artificial Intelligence Review **16**, 3–36 (2001)
26. Stilman, M.: Navigation Among Movable Obstacles. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2007)
27. Stilman, M., Kuffner, J.J.: Navigation Among Movable Obstacles: Real-time Reasoning in Complex Environments. International Journal of Humanoid Robotics **2**(04), 479–503 (2005)
28. Stilman, M., Nishiwaki, K., Kagami, S., Kuffner, J.: Planning and Executing Navigation Among Movable Obstacles. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 820–826 (2006)
29. Uğur, E., Öztop, E., Şahin, E.: Goal emulation and planning in perceptual space using learned affordances. Robotics and Autonomous Systems **59**(7–8), 580 – 595 (2011). DOI http://dx.doi.org/10.1016/j.robot.2011.04.005. URL http://www.sciencedirect.com/science/article/pii/S0921889011000741
30. Veloso, M.M.: Planning and Learning by Analogical Reasoning. Springer Verlag (1994)
31. Walker, S., Salisbury, J.K.: Pushing Using Learned Manipulation Maps. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
32. Wolpert, D.M., Kawato, M.: Multiple Paired Forward and Inverse Models for Motor Control. Neural Networks **11**(7-8), 1317–1329 (1998). DOI 10.1016/S0893-6080(98)00066-5. URL http://dx.doi.org/10.1016/S0893-6080(98)00066-5
33. Zito, C., Stolkin, R., Kopicki, M., Wyatt, J.: Two-level RRT Planning for Robotic Push Manipulation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2012)