Planning to Poke: Sampling-based Planning with Self-Explored Neural Forward Models

Lars Henning Kayser*, Michael Görner*, Matthias Kerzel, Stefan Wermter, Jianwei Zhang†

***Abstract*— This work presents a self-supervised robotic pusher setup that can acquire forward models for pushing tabletop objects by training neural networks. The resulting models are used for multi-step path planning. An explicit world representation allows generating collision-free and feasible object trajectories that avoid local minima. Through closed-loop replanning, the system can generate sequences of pokes that move objects to specified goal poses.**

I. INTRODUCTION & RELATED WORK

Goal-directed pushing has attracted much interest in robotics research over the years. Mason et al. [1] described theoretical foundations of two-dimensional push mechanics and provided the means for analytical approaches to predict push effects (e.g. [2]). To apply such approaches in practice though, the models require known friction distributions and coefficients of pushed objects which are hard to estimate. Even when key values of the model are approximated from data, the resulting predictions rely on a number of assumptions that are invalid in most situations, e.g., a uniform weight distribution within the object [3].

Thus, most current research aims to learn to predict push effects directly [4, 5, 6, 7]. Because it is inherently unstable to push objects in straight lines with single-contact pushes, many authors restrict their setups to stable pushes with multiple contacts or contacting surfaces. In end-to-end learning frameworks, a robotic pusher can learn policies to push objects towards goals from raw camera images [8]. Such approaches demonstrate impressive behavior when enough training data can be collected. However, the behavior of the overall system becomes hard to control and multi-step prediction is infeasible. In contrast, we learn object-centric push models for a single-contact pusher that can be used together with traditional planning-based systems to generate collision-free tentative trajectories.

II. REPRESENTATION

In order to reason about pushes applied to an object, in contrast to pushes applied to the robot's environment, we generate object-centric models that rely on object pose estimation. To get precise estimations, we utilize fiducial markers [9] attached to the object in our current experiments (see Fig. 1). We further assume the object will never tip

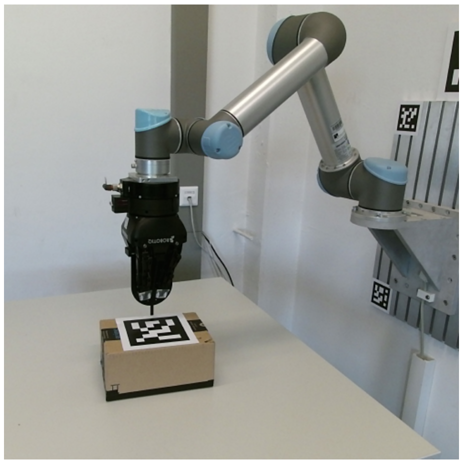

Fig. 1. The robotic setup pushes a target object around on the table. A rounded cylinder is mounted as push tool to ensure a single tool-object contact.

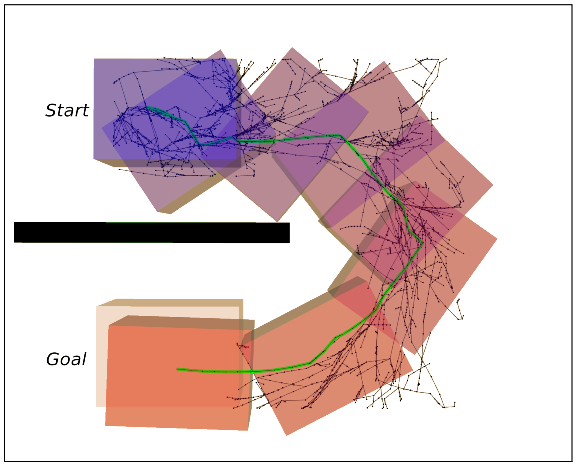

Fig. 2. Collision-free sequence of pushes to reach goal (in green) based on learnt push dynamics. The black graph illustrates the explored state space during sampling-based planning.

over when pushed, and thus its state can be represented sufficiently in $SE(2)$. While this, of course, does not hold in general, it is often justified in contexts with low-friction surfaces when pushes are applied parallel to the support surface.

To help select reasonable push candidates, we additionally require a bounding contour of the object. For a simple box, the contour can be identical with the shape of the object, but any reasonably accurate convex contour suffices for our purposes.

*These authors contributed equally

†All authors are with the University of Hamburg, Germany.
Email `lkayser,goerner,kerzel,wermter, zhang@informatik.uni-hamburg.de`

This research is partially funded by the Horizon2020 RISE project STEP2DYNA and the German Research Foundation (DFG) and the National Science Foundation of China in project Crossmodal Learning, TRR-169.

Building on this representation, we can define push candidates. These consist of a contact/start point $\langle x_p, y_p \rangle$ on the contour, a push direction $\vec{v_p}$ and a push distance d_p. Notice that $\vec{v_p}$ and the normal vector α_p of the contour at the contact point form an angle $\beta_p < \frac{\pi}{2}$rad, so that the pusher moves towards the object. In contrast to previous research which parameterized pushes in the center point of the push [8], this representation allows to easily sample push candidates which do not start in an object collision also for small push distances.

III. Methods

A. Autonomous Exploration

To approximate a model of how objects behave when pushed, their behavior has to be observed sufficiently often. To do so, we generate and perform sample pushes which randomly move the object around on the table. For every trial, we uniformly sample a contact point on the object contour and a push vector $d_p \cdot \vec{v_p}$, such that the corresponding $|\beta_p|$ is below some reasonable threshold, 0.5rad in most of our experiments.

The actual push then comprises a series of robotic motions towards a pre-push point close to the contact location, a push-approach and push motion, and finally a retreat away from the object to ensure the object is visible in the camera image. The setup is modeled in the *MoveIt!* planning framework [10] which generates and performs these trajectories autonomously without unintended collisions.

Longer exploration might eventually move the object off the table. A common way to prevent this in self-supervised setups is to enclose the workspace in low walls and use this same workspace for evaluation. However, in our case this approach would lead to multiple contacts with the object for some pushes and the resulting push effects would depend on the current object location. To learn a push model that is intrinsic to the object and can be used for planning collision-free trajectories, we need to keep the object out of contact.

So instead, we define a safe center point in the middle of the support surface and restrict free exploration within a radius around it. Whenever the box moves outside after a push, we restrict the next pushes such that $\vec{v_p}$ always points towards the center point. Assuming a push does not move the object in the exact opposite direction in our exploration domain, this effectively moves the object back inside the safe zone. Additionally, we define a slightly larger emergency-stop radius and stop exploration if the object ends up outside.

The described setup takes approximately 8 seconds to perform a push and collect a full sample. A total of 3000 push samples were collected for two different objects each. While the first object had its weight equally distributed, the second one was deliberately biased by placing a weight in one corner.

As we consider only the object, regardless of its pose in the environment, the amount of samples required to capture the explored domain is reduced tremendously and the collected datasets should suffice. Some characteristic samples for each object are illustrated in Figure 3.

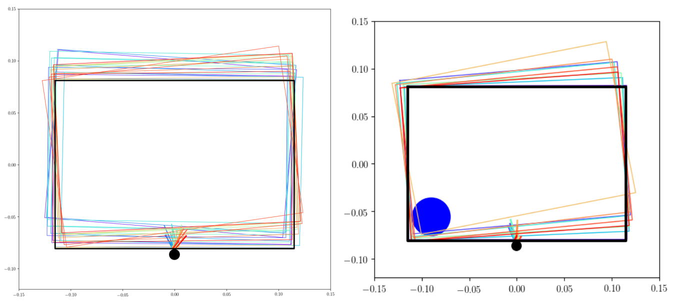

Fig. 3. Illustration of explored push effects on box with uniform weight distribution (left) and biased weight distribution (right). All depicted pushes share the same contact point, but vary in push angle and length as indicated by lines.

B. Forward Model

To feed the samples to a neural network in a sufficiently simple way, we represent them as $\langle x_p, y_p, \alpha_p, \beta_p, d_p \rangle$. All features are min-max normalized. The predictor is trained to output an $SE(2)$ transform $\langle x, y, \gamma \rangle$ for the object. We train a small MLP with a single hidden layer of 100 units, using ReLU activation. In order to account for the type of the output, we utilize a loss that balances the orientation and the translation error:

$$L_{SE(2)} = \sqrt{L_x^2 + L_y^2} + \eta \cdot L_\gamma{}^1$$

Eventually, training the network with *Adam* optimization yields a test-set error of 0.006m for the distance component(x, y only) and 0.022 in $SE(2)$ distance. The network architecture and hyperparameters were empirically determined through manual grid search and *hyperopt*-based optimization [11]. The resulting test-set error is close to the accuracy of our perception setup.

C. Integration with Sampling-based Planning

Utilizing the trained forward models, we can apply arbitrary forward-planning methods to generate goal-directed trajectories. In this work, we employ a variant of the well-known *Rapidly-exploring Random Trees* (RRT) [12] algorithm, building on the versatile *OMPL* planning infrastructure [13]. The traditional algorithm plans trajectories in the state space of object poses by sampling random (goal-biased) target poses and extending the closest explored state towards the target. An accurate inverse dynamics model could be used to extend this mechanism to include control. On the other hand, due to the sampling-based nature of the algorithm, it is sufficient for the sampled controls to move the object in the approximate direction of the target. To achieve this, we sample a small number of valid controls and weight them according to our forward model.

This approach also allows us to put external restrictions on the controls to apply by restricting the sample generation. As pushes with a very big normal offset β_p cause only minimal displacement of the object, we restrict candidates to $-0.5 < \beta_p < 0.5$ in practice. Additionally, we aim

[1] $\eta = 0.5$ in the experiments

to generate push sequences with short movements for the robotic arm and smooth execution. To this end, we prefer controls with contact points in the vicinity of the contact point of the previous control.

To detect states which would collide with the known environment during planning, we utilize MoveIt!'s collision checking capabilities and its *Planning Scene* world representation.

Figure 2 illustrates a planning result, including the spanned tree, generated with the described system.

Due to the known inaccuracies in the forward model and stochastic variations during execution, the generated plan cannot be performed successfully without feedback. Instead, we follow the idea of model-predictive control (MPC), using the planning system as the model. MPC usually replans after each execution step, thus assigning particular importance to the first step of each plan. However, with sampling-based single-shot planners, the first action of a solution is not necessarily a correct step towards the goal. To reduce this problem, our execution pipeline tracks the current object pose and only replans when the real pose differs from the plan by more than some threshold. This helps to stabilize execution, effectively following a plan as long as it reflects the current world state.

IV. CONCLUSION & OUTLOOK

We presented an autonomous robotic setup to explore the effects of pushes on specific objects. After learning forward models from the collected data with neural networks, we utilize them in traditional sampling-based motion planning algorithms to generate collision-free sequences of pokes and move the object to arbitrary goal poses.

Thus, we contribute an integration of self-supervised forward-model learning and sampling-based planning, demonstrating its feasibility. The complete system can utilize adequate forward models without analytical restrictions and can overcome local minima. It is an example of how learning-based approaches can complement traditional planning systems.

Future research aims to explore more natural objects, removing the need for fiducial markers by neural-network-based state estimation. To generate smoother and coherent sequences of pushes even with sampling-based planning, we will consider optimizing planners and trajectory optimization as a post-processing step. Additionally, it has been demonstrated recently that push dynamics involving the interaction with predictable contacts can be learned successfully [7]. A similar approach might enable us to plan not only collision-free trajectories but could also allow specified contacts with environment surfaces.

REFERENCES

[1] Matthew T. Mason. "Mechanics and Planning of Manipulator Pushing Operations". In: *The International Journal of Robotics Research* 5.3 (Sept. 1986).

[2] Kevin M. Lynch and Matthew T. Mason. "Stable Pushing: Mechanics, Controllability, and Planning". In: *The International Journal of Robotics Research* 15.6 (Dec. 1996), pp. 533–556.

[3] Federico Ruiz-Ugalde, Gordon Cheng, and Michael Beetz. "Fast adaptation for effect-aware pushing". In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, Oct. 2011.

[4] Manfred Lau, Jun Mitani, and Takeo Igarashi. "Automatic learning of pushing strategy for delivery of irregular-shaped objects". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011.

[5] T. Hermans et al. "Learning Stable Pushing Locations". In: *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EPIROB*. 2013.

[6] Haolin Yang, Fuchun Sun, and Di Guo. "Pushing operation of manipulator based on experience learning: Position prediction of an object and pushing analysis". In: *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*. IEEE, Sept. 2014.

[7] Marek Kopicki et al. "Learning modular and transferable forward models of the motions of push manipulated objects". In: *Autonomous Robots* 41.5 (June 2017), pp. 1061–1082. ISSN: 1573-7527.

[8] Pulkit Agrawal et al. "Learning to Poke by Poking: Experiential Learning of Intuitive Physics". In: *CoRR* abs/1606.07419 (2016). arXiv: `1606.07419`.

[9] John Wang and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016.

[10] David Coleman et al. "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study". In: *Journal of Software Engineering for Robotics* 5.1 (May 2014), pp. 3–16. URL: `http://moveit.ros.org`.

[11] J. Bergstra, D. Yamins, and D. D. Cox. "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013, pp. I-115–I-123.

[12] J.J. Kuffner and S.M. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. IEEE.

[13] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012). `http://ompl.kavrakilab.org`, pp. 72–82.