PokeRRT: A Kinodynamic Planning Approach for Poking Manipulation

Anuj Pasricha*, Yi-Shiuan Tung, Bradley Hayes, and Alessandro Roncone

***Abstract*— This work introduces *PokeRRT*, a novel motion planning algorithm that demonstrates poking as an effective non-prehensile manipulation skill to enable fast manipulation of objects and increase the size of a robot's reachable workspace. Our qualitative and quantitative results demonstrate the advantages of poking over pushing and grasping in planning object trajectories through uncluttered and cluttered environments.**

I. Introduction

Non-prehensile manipulation (NPM) offers a complementary solution to prehensile (*grasping*) manipulation by significantly expanding the size and dimensionality of the operational space of a robotic manipulator [1], [2], [3]. Realistic robot applications such as those that expect the robot to operate in the presence of occlusions, in ungraspable configurations, or in dense clutter, may result in failure modes for robot operation through traditional grasping. Therefore, it is advantageous in such cases to introduce NPM primitives into the robot's skillset.

In this work, we focus on poking as a core NPM primitive and demonstrate how it allows rapid object manipulation and expands the reachable workspace of a manipulator arm. Recent work in NPM has focused on pushing manipulation due to the availability of large-scale datasets [4] and the inherent controllability of the skill. Contrary to pushing which operates under the quasistatic assumption to reduce modeling complexity [5], [6], [7], poking must consider the non-negligible effects of inertial forces since the object continues sliding over its support surface after robot–object contact is broken.

II. Current Work

Poking is an NPM primitive comprised of two phases: i) *impact*, where the robot end-effector strikes an object at rest and sets it into translational and rotational motion, and ii) *free-sliding*, where the object slides across a planar support surface and comes to rest due to Coulomb friction. Poking has a number of desirable characteristics that makes it complementary to grasping [3]. Additionally, poking serves as a generalized form of pushing in cases where where applied impulse forces are low. In this paper, we present a sampling-based kinodynamic planner called *PokeRRT* which decouples skill modeling and path planning and specifically leverages the following advantages of poking over pushing and grasping: i) it uses instantaneous contact and operates outside the quasistatic regime to expand the size of the manipulator's reachable workspace, ii) it is inherently faster and therefore capable of covering large distances in short periods of time, and iii) it does not impose restrictions on the shape or size of objects being manipulated.

[1]The authors are with the Department of Computer Science, University of Colorado Boulder, 1111 Engineering Drive, Boulder, CO USA `firstname.lastname@colorado.edu`

* Corresponding author.

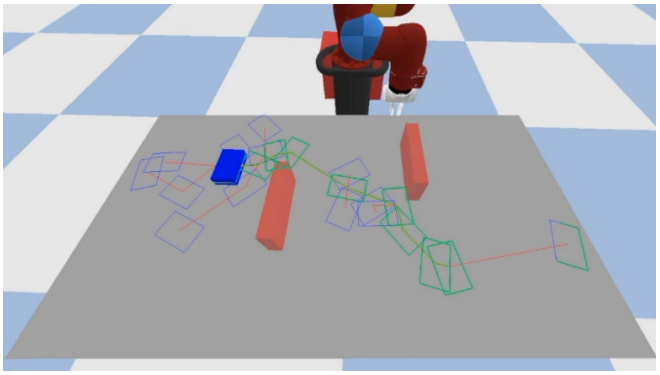

Fig. 1: *PokeRRT* plans an object (blue) path using poke actions through an obstacle-rich (red) environment with a kinodynamic approach to sampling-based motion planning. The planning graph is shown in blue and the execution path is highlighted in green.

PokeRRT leverages PyBullet as the forward model to validate actions that respect robot and object dynamics [8]. Path planning for poking operates in the (x, y, θ) object configuration space in a closed-loop manner to compensate for inaccuracies in the simulation poking model (i.e. it replans if the resultant pose from a poke action is outside a predefined threshold) and consists of six steps:

1) Points are sampled uniformly on the object contour and filtered through a conical region originating from a randomly sampled object configuration.
2) Striking points are generated by extending the sampled contour points away from the object in the normal direction.
3) Valid end-effector velocity magnitudes that can be applied by the robot are sampled for each striking point.
4) Feasible actions are applied in simulation to get resultant poses.
5) The resultant pose closest to the randomly sampled configuration is added to the planning graph.
6) This procedure is repeated until a resultant pose falls in the task goal region, at which point the shortest path between start and goal object configurations is calculated. The shortest path is defined as one with the lowest overall number of pokes to leverage poking's capacity to cover large distances quickly.

	Task Time [seconds]						Success Rate
Planner	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S1 - S6**
PokeRRT	**49.69 (21.11)**	**196.54 (124.61)**	**167.55 (138.63)**	**64.14 (52.45)**	**116.35 (89.72)**	**171.77 (103.21)**	**0.88 (0.31)**
Two-Level Push Planner	122.68 (47.07)	284.78 (104.40)	249.32 (68.30)	117.58 (67.05)	N/A	N/A	0.44 (0.26)
Pick-and-Place	16.29 (3.76)	17.71 (3.81)	16.14 (3.25)	N/A	19.15 (4.99)	N/A	0.67 (0.00)

TABLE I: Task times (shown as *mean (stddev)*) and success rates (averaged across all six scenarios) are presented for various planning algorithms in simulation across multiple scenarios. Overall, poking is faster than pushing and leads to higher success rates than pushing and grasping.

III. Evaluation

We measure task times (in seconds) and success rates in the final planned path generated by *PokeRRT* across six test scenarios (see Figure 2). Task time is defined as the sum of planning, execution, and replanning times. Simulation results are averaged over 250 trials—a trial fails if the planner does not find a valid plan to the goal region in 240 seconds or if the object falls off the its support surface during execution. To evaluate pushing against *PokeRRT*, we use the *Two-Level Push Planner* presented in [5]. *Pick-and-Place* is performed in an open-loop manner with predefined grasps for known objects.

Table I shows the task times and success rates for *PokeRRT*, *Two-Level Push Planner*, and *Pick-and-Place*. Poking is successful in all scenarios while pushing fails in S5 and S6 and grasping fails in S4 and S6. *Two-Level Push Planner* has a low overall success rate (44%) because pushing fails in S5 due to collision with the workspace divider (since pushing, by definition, must maintain constant contact between the end-effector and the object, thereby satisfying the quasistatic assumption). Constant contact also implies that the push action path is longer than that of instantaneous poking, resulting in collisions between the end-effector and workspace obstacles in narrow spaces in S2 and S3. Pushing also fails in S6 due to limited robot reachability. *Pick-and-Place* always succeeds in simulation for S1, S2, S3, and S5 due to lack of sensing uncertainty. However, it fails in S4 because the object being manipulated is too wide for the gripper and in S6 because the goal pose for the object is out of robot reach.

Task times for *PokeRRT* are lower than those for *Two-Level Push Planner* across all scenarios. *Pick-and-Place* has the lowest task time because it does not involve kinodynamic planning in the object configuration space—the robot moves to object pose, grasps, and moves to goal pose, resulting in a single executed action. Collectively, our qualitative and quantitative results demonstrate that poking expands robot reachability and dexterity by leveraging instantaneous impact and enabling fast object manipulation through uncluttered and cluttered environments. Our next steps involve extensively testing *PokeRRT* in the real-world to see if the empirical results support the presented insights from simulation. We also intend to explore additional applications of poking and characterize its dynamics via a combination of learning and analytical models.

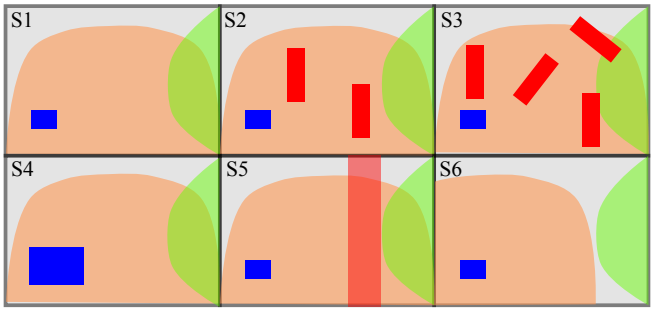

Fig. 2: *PokeRRT* is evaluated in six scenarios—no obstacles (S1), 2 obstacles (S2), 4 obstacles (S3), wide object (S4), tunnel (S5), and non-overlapping shared workspace (S6). Reachable regions of two robots operating in a shared workspace are depicted in orange and green. The first robot successfully pokes the object (blue) from its reachable workspace (orange) to the goal region (green) in all scenarios while avoiding obstacles (red).

References

[1] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 64–92, 1999.

[2] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *arXiv preprint arXiv:1903.11239*, 2019.

[3] W. Huang, "Impulsive manipulation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[4] M. Bauza, F. Alet, Y.-C. Lin, T. Lozano-Pérez, L. P. Kaelbling, P. Isola, and A. Rodriguez, "Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video," *arXiv preprint arXiv:1910.00618*, 2019.

[5] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt, "Two-level RRT planning for robotic push manipulation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 678–685.

[6] A. Kloss, M. Bauza, J. Wu, J. B. Tenenbaum, A. Rodriguez, and J. Bohg, "Accurate vision-based manipulation through contact reasoning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6738–6744.

[7] J. K. Li, W. S. Lee, and D. Hsu, "Push-Net: Deep planar pushing for objects with unknown physical properties." in *Robotics: Science and Systems*, 2018.

[8] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.