Sensorless Single-agent Point-contact Push Planning

Jialiang Zhao, Aohan Mei

***Abstract*— This paper addresses the problem of pushing an arbitrary shaped object and make it move along a desired path using a single agent. During the pushing process, only point-contact between the agent and the object exists and there is no perception of the location of the object to serve as feedback. The proposed approach is based on the object kinematics that could predict the pose of the objects center of mass after each push. Push planning, path planning and pushing execution are discussed in this paper. Experiments show that we could move arbitrary shaped object along an arbitrary path using this system.**

I. Introduction

Pushing, as an important way of moving object, has a lot of characters worth researching. The topic about how to manipulate a sensorless push planning with a single robot is discussed in this paper. The difficulty of this problem is that we only know the position and configuration of the robot and the object at the initial point, after which the object is pushed along its ideal path with pure mathematical calculation.

Considering all the difficulties above, an algorithm was derived to predict the objects position and configuration after every single pushing action of robot. The algorithm determines the dynamics of the object by importing the concept of Friction Center, Friction Cone, Candidate Control Input, to name a few, which will be discussed with detail in the following contents. By sampling contact points and pushing direction, the path planning process was successfully achieved.

In push planning, it is shown that, given the position of contact point, the objects center of mass, the friction coefficient of supporting surface, by sampling numerous candidate contact points and select the best one, we could move arbitrary shaped object to move along an arbitrary path.

In path planning, this paper derives a Safety Margin which could avoid the collision between the agent and the object. This approach is generalized to the case where any 2-D object whose shape could be described with mathematical words are able to be applied with the algorithm and generate a valid plan.

Experiments were carried out with Turtlebot2 in real setup and simulator.

A. Prior Works

By deriving the conception of maximum independent capture discs (MICaDs), [1] Attawith, Fred and Jean (2002) developed a method using three disc-shaped robots to manipulate a polygonal object in the plane in the presence of obstacles, while several assumptions like no jamming occurs during execution process.

Mason (1986), demonstrating pushing is an essential component of many manipulator operations, presented a theoretical exploration of the mechanics of pushing and illustrated the feasibility of the theory in analysis of robotic manipulator operations [2]. In his article, he introduced a Voting Theorem which assumed quasi-static pushing and uniform contact coefficient of friction and could predict the rotation direction of a pushed object given the friction cone edges and line of pushing.

By studying the mechanics, controllability and planning of pushing process, Kevin and Mason (1996) developed algorithms to automatically find pushing plans to position and orient parts in the plane [3].

Goldberg (1991) demonstrated that for any n-vertices polygonal, a shortest sequence of sensorless mechanical gripper actions which could orient the object up to symmetry in its convex hull always exists, which suggests that a sequence of single-point pushing along the ideal path of the object also exists [4].

Huan Liu (2011) derived a physics-based box pushing model involving three steps: perceive, plan and act. He also optimized Masons Voting theorem and provided an algorithm to accurately judge and predict the motion of the object, which provided a lot of solid theoretical support for this paper [5].

B. Our Contribution

An improved physical model of the kinematics of planar push operations was introduced. With this model, this paper further presented a method to plan a path of an agent to push an arbitrary shaped object to move along an arbitrary path without unexpected collision.

With numerous experiments in both physics simulator and real setup, we could achieved a $0.5m$ accuracy of moving a $0.7m$ cubic paper box along a $4m$ S-shaped path within 100 pushes.

II. System Dynamics

This section discusses the dynamics in this system. Friction and object kinematics are analyzed.

A. Friction

We mainly explore two kinds of friction: friction of the contact and friction between the object and the supporting surface.

- Contact Friction. A contact friction exists between the agent and the object. The contact friction always opposes to the direction of the relative motion of the agent

This research was conducted in the EECS department, UC Berkeley

The authors are with EECS Department, University of California, Berkeley, Berkeley, CA, 94704.

to the object. This friction is modeled as a friction cone, which will be discussed later.

- Sliding Friction. A friction between the object and the supporting surface exists. If the object translates, this friction effects as a constant force opposes to the relative translation direction. If the object rotates, this friction effects as a torque opposes to the rotation. Static and kinetic friction coefficients are assumed to be the same.

B. Friction Cone

Friction between the object and the agent can be analyzed with a friction cone [6], as in Fig. 1

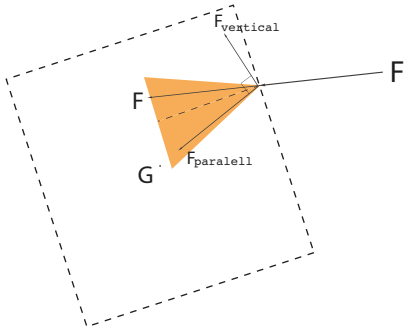


Fig. 1: Friction between the object and the agent.

If a pushing force lies within the friction cone, this force will not slide along the contact surface because of the contact friction. The size of friction cone is determined by the friction coefficient between the object and the agent. The angle of the cone with respect to its normal line can be determined by

$$\alpha = tan^{-1}\mu$$

If the push lies outside of the friction cone, the contact point will slide along the contact point. In this case we can divide the push into two components, one along the sliding direction which should be on the edge of the object, the other one along the normal of friction cone.

In this problem, since we are able to choose pushing direction, and in order to avoid accumulated error and for simplicity, we only plan pushes along the normal of the friction cone.

C. Object Kinematics

Object kinematics resulted from the push is discussed in this section. We assume density of the object and friction coefficient between the object and the supporting surface are uniform, so that the center of mass is the box's geometric center, and lies directly above the friction center.

As a result, as shown in Fig. 2, the resulted motion of the object can be viewed as a combination of a translation along the line of the contact point and the center of mass, CG, and a rotation about the friction center [5], which is also the center of mass, G.

Note that this only holds when the pushing distance CC' is infinitesimal.

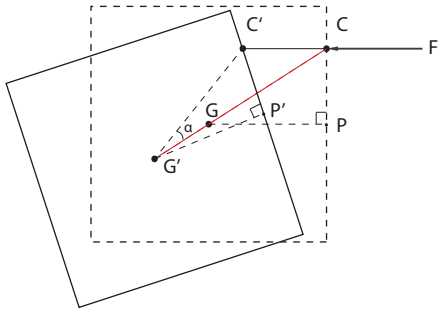


Fig. 2: Resulted motion of a cubic box from a push at point C.

Fig. 3 shows a zoom-in of the triangles. We can solve the displacement of center of mass, $\boldsymbol{GG'}$, and the rotation α by solving the two triangles. That's to say, given $C, \boldsymbol{CC'}, P$, we are going to solve for $\boldsymbol{GG'}$ and α.

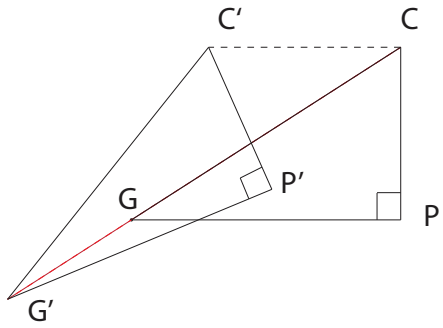


Fig. 3: Resulted motion of a cubic box from a push at point C. Zoom-in.

We can find $\boldsymbol{GG'}$ by

$$\begin{aligned}\boldsymbol{GG'} =& \boldsymbol{CG'} - \boldsymbol{CG} = \frac{\boldsymbol{CG}}{|CG|}|CG'| - \boldsymbol{CG} \\ =& \frac{\boldsymbol{CG}}{|CG|}(|CQ| + |QG'|) - \boldsymbol{CG} \\ =& \frac{\boldsymbol{CG}}{|CG|}(|CC'| \cdot cos(\angle GCC')) + \\ & \sqrt{|CG|^2 - (|CC'|^2 \cdot sin(\angle GCC'))^2} - \boldsymbol{CG}\end{aligned} \tag{1}$$

Rotation can be found with

$$\angle CG'C' = cos^{-1}(\frac{\boldsymbol{G'C} \cdot \boldsymbol{G'C'}}{|G'C| \cdot |G'C'|}) \tag{2}$$

III. Path Planning

As ahown in Fig. 4, path planning for the agent can be achieved in three steps: push planning, which generates the pushes that could achieve the goal without consideration of actual execution; path planning, which takes the current poses of agent and object into consideration and checks collision to generate a collision-free path for the agent; push execution, which controls the hardware to move along the desired agent path.

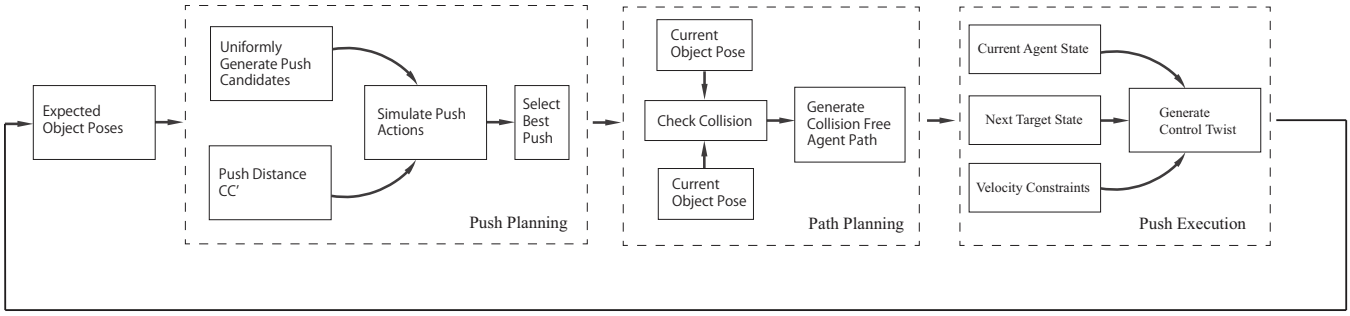


Fig. 4: Diagram for path planning.

A. Choosing Contact Point

Before the actual pushing begins, the paper pre-defined an ideal path in our computer. From the kinematic calculation above, we are able to calculate out the next position G of the objects center of mass. Based on the shape and dimension of the object, we evenly sample 1000 candidate points from the edges of the object, using the kinematic algorithm above to calculate 1000 different G. From the 1000 candidate results we choose the one closest to our ideal path. The contact point corresponding to the optimized G is selected to be the execution contact points.

Algorithm 1 Choosing Contact Point

Input: desired object path ξ and uniformly sampled contact points $\{p_k\}$ on the edges of the object

Output: The best contact point p_d that makes the object move along the desired path.

$p_d = None$
$d = \infty$
for $k = 1$ to n **do**
    GG' according to Eq. 1
    $G' = G + GG'$
    $d_t = distance(\xi, G')$
    if d_t is smaller than d **then**
        $p_d = p_k$
        $d = d_t$
return p_d

B. Step Size

From the kinematic algorithm, we are acknowledged that only when the step size is infinitesimal could the kinematic equation hold true. This condition is infeasible in practical condition because if the step size is too small, it will result into too many pushes. However, if the step size is too large, it will enlarge the error accumulation. After trial-and-error process, this paper, in order to find the suitable step size that could roughly follow the kinematic equation and minimize the pushing times, defines the step size to be a constant number 0.18m.

C. Collision Avoidance

Imagine if we do not consider the avoidance of collision, there is a great chance that during the transition of the robot from last contact point to next contact point an unexpected and undetected collision between the agent and the object will exist. This will cause an undesired translation or rotation of the object and consequently result into an exaggeration of error accumulation. Consider the elimination of unexpected collision and the possible execution error of the robot and object, this paper derives a method to avoid collision.

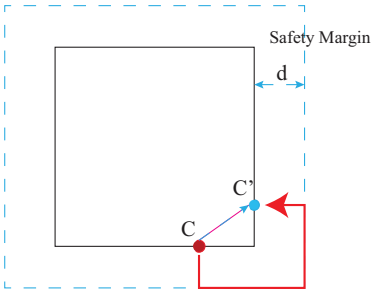


Fig. 5: Safety margin of a cubic object.

According to the dimension and shape of the object, we adopt a safety margin, as shown in Fig. 5 around the edges of the object. The effect of the safety margin is like assembling a fence around the object to protect it. During contact point switching process, there are two conditions of the robot:

- The shape of safety margin is convex and the connection line between the adjacent contact points lies within the convex area. If that is the case, we will use the algorithm to drive the robot to find the shortest path to circle around the safety margin to reach the next contact point.
- The safety margin is concave and the connection line of the adjacent contact points lies out of it. In this case, the robot is just demanded to drive a straight line towards the next contact point.

D. Motion Execution

In this task we modeled the agent as a unicycle model with non-holonomic constraints [7]. A unicycle-type robot is a robot with one steerable drive wheel [8]. This robot can have acceleration in $\dot{x}$ and $\dot{\theta}$ but cannot move along $\dot{y}$.

For a unicycle-like robot, we give its desired v, w, then we can find its dynamic by

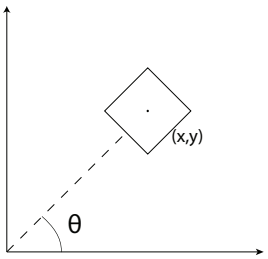


Fig. 6: A unicycle model of mobile robot.

$$\begin{aligned} \dot{x} &= v \cdot cos(\theta) \\ \dot{y} &= v \cdot sin(\theta) \\ \dot{\theta} &= w \end{aligned} \tag{3}$$

In target following problems, a unicycle-like robot can be controlled via a nonlinear inner-outer control structure [9].

The inner-loop, which generates motor control signals from the desired v_d, w_d, is built-in in our agent and is not a focus of this paper. The outer loop can be written as

$$\begin{bmatrix} v_d \\ w_d \end{bmatrix} = \begin{bmatrix} cos(\theta_r - \theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix} - u$$

and

$$u = \begin{bmatrix} -k_1(x_r - x) \\ k_2 v_r sinc(\theta_r - \theta)(y_r - y) - k_3(\theta_r - \theta) \end{bmatrix}$$

where k_1, k_2, k_3 are positive gains [10]. We choose $k_1 = 3.0, k_2 = 2.4, k_3 = 2.4$ in this problem.

IV. Evaluation

A. Step Size Tests

The push distance, or step size, should be infinitesimal, and the kinematic equation only holds in this situation. However, too many pushes will be need if the step size is too small. We test our planner in different step size settings. Fig. 7 shows some results with $0.1, 0.2, 0.4$ meter step sizes.

The number of pushes is roughly negatively linear with step size. In this task we choose 0.18 as the step size, which can finish the S-shaped path cubic box pushing problem in 91 pushes as in Fig. 7c.

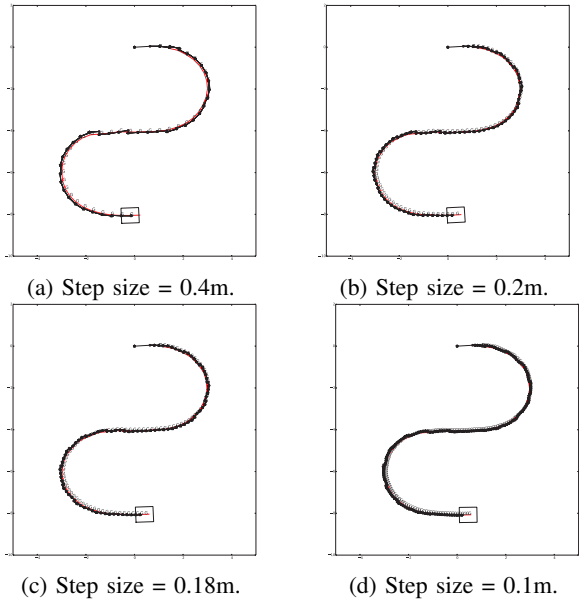
B. Pushing Different Objects

Efficiency and accuracy of the object path following differ a lot with objects of different shape. Here we show some results with 3 different objects.

As shown in Fig. 8, this system gives satisfying solution to different objects.

C. Tests in Simulator and Real Setup

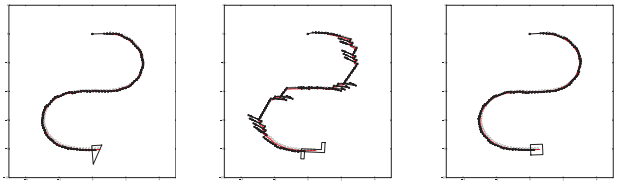
We tested this system both in Gazebo, a physics simulator, and real setup.



(a) Step size = 0.4m. (b) Step size = 0.2m.

(c) Step size = 0.18m. (d) Step size = 0.1m.

Fig. 7: Plots of planning with different step sizes. 7a finishes in 41 pushes. 7b finishes in 82 pushes. 7c finishes in 91 pushes. 7d finishes in 163 pushes.



(a) Triangular object. (b) Z-shaped object (c) Cubic box

Fig. 8: Plots of planed pushes for triangular object, z-shaped object, and cubic object. Step sizes are all set to 0.2m. 8a finishes in 90 pushes. 8b finishes in 97 pushes. 8c finishes in 91 pushes.

1) Tests in Gazebo: Gazebo is a 3D real-time physics simulator which could simulate pushes and friction in different settings [11]. We use Turtlebot Simulator as our agent and objects with different shapes as our pushes object. Friction coefficients are all set to $\mu = 0.35$, which is close to the relative friction between wooden surfaces. Fig. 9 shows the result.

2) Tests in real setup: We tested in a real setup with a Turtlebot 2 [12] robot and a $0.7m \cdot 0.7m \cdot 0.4m$ paper box as the object being pushed as shown in Fig. 10.

Testing results are greatly influenced by the initial poses and the surface condition. With a relatively fixed initial poses, we tested dozens of times of the same pushing procedure. All of the final positions of the object lie within a 0.5m circle centered in the desired final position.

V. Conclusion

This paper introduced an approach to push an object and make it move along a desired path without sensor. Kinematics of the object and prediction of the motion of the objects center of mass are given. We have presented an

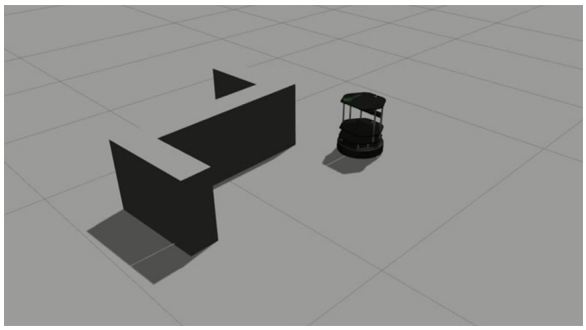


Fig. 9: Test with Gazebo.

Fig. 10: Test with real robot.

implementation of the proposed approach, along with the experimental results.

The step size of the agent, in principle, should be infinitesimal. We have tried to find an ideal number which could balance the number of pushes and the precision of the object trajectory. Indeed, the pre-defined ideal path is a continuous and smooth line, a finite step size could segment the path into several direct lines and exaggerate the error accumulation.

There are also some weak assumptions which may affect the result of the algorithm. In practical condition, we could not guarantee that the friction coefficient of the supporting surface is uniform, nor could we guarantee that the density of the object is evenly distributed. We could also not assert that the actuator is perfect. Hence, the approach proposed by this paper only focuses on ideal condition where every hardware demand is satisfied.

VI. Acknowledgment

This research was supported by the EECS Department, University of California, Berkeley. We thank Professor Ruzena Bajcsy and Teaching Assistants Chris Correa and Valmik Prabhu from the EECS Department, University of California, Berkeley who provided insights and expertise that greatly assisted the research. We would also like to show our gratitude to Vodafone @ Berkeley Robotics Laboratory of EECS Department for providing us with the hardware of our researching and experiment space.

References

[1] A. Sudsang, F. Rothganger, and J. Ponce, "Motion planning for disc-shaped robots pushing a polygonal object in the plane," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 550–562, 2002.

[2] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[3] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996.

[4] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2-4, pp. 201–225, 1993.

[5] H. Liu, "Pushing with a physics-based model," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[6] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[7] C. C. De Wit and O. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1791–1797, 1992.

[8] B. A. Francis and M. Maggiore, "Models of mobile robots in the plane," in *Flocking and Rendezvous in Distributed Robotics*. Springer, 2016, pp. 7–23.

[9] R. Carona, A. P. Aguiar, and J. Gaspar, "Control of unicycle type robots tracking, path following and point stabilization," 2008.

[10] C. C. d. Wit, H. Khennouf, C. Samson, and O. J. Sordalen, "Nonlinear control design for mobile robots," in *Recent trends in mobile robots*. World Scientific, 1993, pp. 121–156.

[11] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[12] W. Garage, "Turtlebot," *Website: http://turtlebot. com/last visited*, pp. 11–25, 2011.