Sampling-based Planning for Underactuated or Deformable Systems Utilizing Contact

Shang Ma, Ellis Ratner, Donggun Lee, and Claire J. Tomlin

***Abstract*— Manipulating underactuated or deformable systems must use contact with the environment. Most of the previous sampling-based methods consider the environment an obstacle to avoid but not objects we can utilize for better manipulation. This paper presents a sampling-based motion planning approach where we utilize contact with the environment for sufficient system controllability. Our key idea is to divide the state space into the non-contact space and the other and to apply two different schemes in each space. Near the boundary of the two spaces, our method samples a *pseudo goal*. Rapidly-exploring Random Tree generates a path between the initial state and the *pseudo goal*, and numerical simulation, such as PyBullet, validates a path from the *pseudo goal* to the goal. Our simulation demonstrates a high success rate and numerical efficiency compared to other methods. Finally, we present a demonstration of our planner on a real-robot manipulation problem.**

Fig. 1. (Left) A vacuum gripper holds an object, causing it to deform due to its weight. (Right) A parallel gripper holds a flexible piece of paper. The robot can change the configuration of the paper by making contact with the environment.

I. Introduction

Modern robotic systems, whether operating in an industrial or a home setting, must often manipulate non-rigid objects to complete their tasks. For example, an industrial robot working in a warehouse that grasps a heavy box with a vacuum gripper must account for the suction cups' flexibility in precisely manipulating it. Due to the flexibility, the robot can not fully control the box's motion. For example, in Fig. 1 (Left), we see how the suction cup deforms due to the object's weight. However, it is possible to leverage *contact* with the environment to complete the task despite not having full control over all degrees of freedom. For example, the robot can use the shelf's surface to correct the box's orientation and slide it into place. Therefore, enabling the robot to develop motion plans that utilize the contact to compensate for the flexible, underactuated dynamics would significantly increase the robot's capabilities. Unfortunately, however, planning for an underactuated system utilizing contact is a challenging problem.

Sampling-based motion planning approaches, such as the Rapidly-exploring Random Tree (RRT) algorithm [1], have been used widely in robotics due to their simplicity to implement and ability to solve complex problems. However, for contact-rich manipulation problems with underactuated dynamics, there has been relatively less success in applying RRT-based approaches. Furthermore, widely-available simulators such as Gazebo [2], Mujoco [3], Bullet [4], and so on, have enabled more reliable, higher-fidelity simulation of contact dynamics. Motivated by this, in this paper, we propose to bring together the advantages of sampling-based planning and simulation-based models of contact dynamics to plan efficiently for underactuated systems that utilize contact.

Specifically, we propose a two-step approach based on the RRT algorithm. Our key idea is to introduce a *pseudo-goal* state as an intermediate step to guide the planner towards contact surfaces in the environment. We then divide planning into first planning a path from the start to the pseudo-goal and then from the pseudo-goal to the goal. Since the robot is guaranteed not to be in contact before reaching the pseudo-goal, we can save on computation by not simulating the contact dynamics during this first stage of planning. Only when connecting the pseudo-goal to the goal do we need to consider the contact dynamics, which we do by running a dynamics simulator within the planner. As a comparison, to solve the same problem using a conventional RRT algorithm, a naive approach is to plan only over the actuated states and ignore the underactuated states.

Our contribution has three-fold:

1) our sampling-based algorithm leverages a dynamics simulator to plan for an underactuated system that utilizes contact;
2) our simulation result demonstrates the advantages of our method compared to baseline methods;
3) we demonstrate our method for deformable objects via an experiment.

II. Related Work

Sampling-based approaches, such as the RRT [1] and RRT-Connect [5] algorithms, are widely used in robotics due to

S. Ma, E. Ratner and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA. {shangma, eratner, tomlin}@berkeley.edu

D. Lee is with the Aerospace Controls Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA donggun@mit.edu

their simplicity in implementation and success at solving complex motion planning problems. Our work focuses on developing an extension to the RRT algorithm specific to problems involving *underactuated dynamics* that utilize *contact*.

Many existing approaches to planning through underactuated systems with contact use optimization-based approaches. Posa et al. plan through contact using nonlinear optimization with complementarity constraints and demonstrate applications to grasping and manipulation [6]. These approaches, however, are often complex to implement and can suffer from numerical stability issues. [7]–[9] attempt to mitigate these issues by optimizing over a relaxation of the dynamic constraints. In contrast to these approaches, we build on sampling-based planning methods with the aim of avoiding drawbacks such as implementation complexity and numerical stability.

Since first introduced, RRTs have been applied to *kinodynamic* planning problems, i.e., planning problems involving both kinematic and dynamic constraints, e.g. [10], [11]. More recent approaches such as Kinondynamic RRT* [12] and LQR-RRT* [13] use information about the underlying dynamics to guide the sampling; however, these approaches do not readily extend to problems involving complex constraints like contact.

[14] proposes to hierarchically combine sampling-based and optimization-based planners to solve manipulation problems involving contact. Similarly, the recently-developed TrajectoTree approach combines a high-level tree search over contact mode sequences with a low-level optimization-based planner to plan through contact [15]. Similarly, CMGMP uses an optimization-based method to generate contact-based motion primitives, which it then sequences into a full plan using an RRT [16]. R3T is a recently-developed approach for planning through contact, which combines an RRT with the concept of reachable sets [17]. These approaches use an analytic dynamics model rather than a simulation-based model, as we do in this work.

III. Preliminaries and Notations

In this paper, we consider a system whose state consists of actuated and under-actuated joints.

$$X(x) = (X_a(x), X_{ua}(x)), \tag{1}$$

where $X_a(x) \in \mathbb{R}^{n_a}$, $X_{ua}(x) \in \mathbb{R}^{n_b}$. For instance, in a classic cart-inverted-pendulum system, as shown in Fig. 2, X_a is the position of cart on the horizontal direction, and X_{ua} is the angle of pendulum. In addition, $X(g) = (X_a(g), X_{ua}(g))$, $X(\text{init}) = (X_a(\text{init}), X_{ua}(\text{init}))$, and $X(p) = (X_a(p), X_{ua}(p))$ represent the goal state, initial state, and pseudo goal state. For convenience, this paper deals with a particular state setting: $X_a = (x, z) \in \mathbb{R}^2$ and $X_{ua} = \theta \in \mathbb{R}$.

The obstacle in configuration space is defined as $O \subset \mathbb{R}^{n_a}$, and $C \subset \mathbb{R}^{n_a}$ is the space where the underactuated system could have contact with the obstacle.

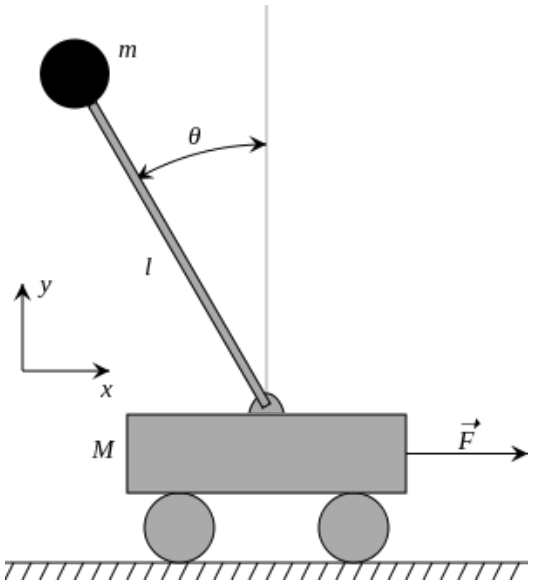


Fig. 2. A classic underactuated cart-pendulum system. The motion on the horizontal direction x is an actuated state X_a, the angle theta is an underactuated state X_{ua}.

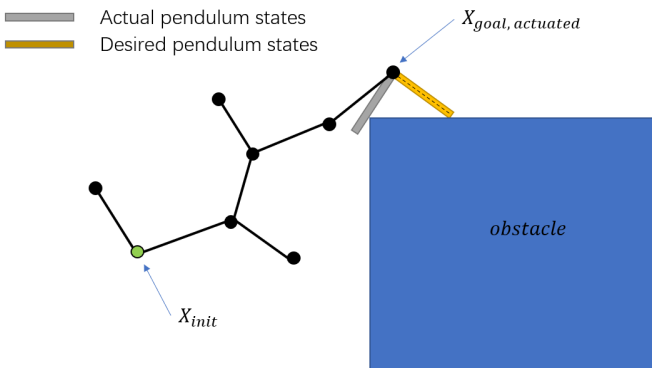


Fig. 3. An illustration of conventional RRT tree diagram in underactuated setting while in collision with the obstacle

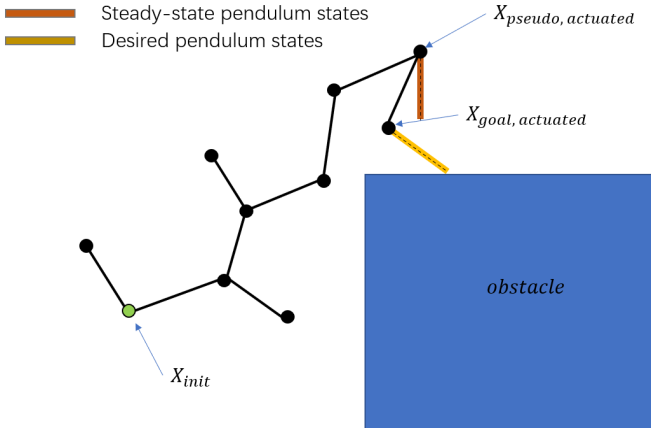


Fig. 4. An illustration of underactuated RRT tree diagram in same setting as Fig. 3

IV. ALGORITHM

We first consider some challenges in solving this planning problem with a sampling-based algorithm. A naive approach would be to plan only over the actuated state X_a, ignoring the underactuated state X_{ua} altogether. The advantage is that we can directly apply the RRT algorithm.

The underactuated state is determined by simulation. In the task scenario presented by this paper, the changing angle θ of the underactuated joint is a result of accelerations on (x, z). The path was given to the simulation as input to compute θ. Due to the nature of underactuated dynamics, two different paths with same start $X(\text{init})$ and same goal $X_a(g)$, may result different final θ_{final} result as shown in Fig. 3 and Fig. 4. In other words, we may reach the point whose actuated state is $X_a(g)$ but whose under-actuated state is not $X_{ua}(g)$. Additionally, as the value of changing θ during motion also depends on its previous value, $\theta_{k+1} = \theta_k + \Delta\theta$ where $\Delta\theta$ is function of accelerations over (x, z). Therefore the entire motion needs to be simulated to ensure the fidelity of the final computed θ_{final}. This leads to computational challenges, later discussed in Section V.

The variant of the RRT method we present in this paper utilizes the existence of obstacles. The focus was path planning so the introduction of $X(p)$ is to help eliminate possible contact in the process of "normal" RRT iteration so that a "collision-free contact-free" path can be planned from $X(\text{init})$ to $X(p)$ without using simulation or complex dynamics constraint. One important premise is the use of the steady state, as mentioned above. Before running the original RRT algorithm, given the final goal $X(g)$ that is in contact with the obstacle O, a set of corresponding actuated state $X_a(g)$ can be derived. Considering the obstacle O and the model of the underactuated system, a pseudo-goal $X(p)$ is randomly sampled to be replaced as the destination of the growing tree. For the particular task scenario in the paper, $X_{ua}(g)$ represents the position of the end of the pendulum, which is also in contact with obstacle O, and $X_a(g)$ represents a set of states of the baseline of the pendulum so that when the system moves its actuated joints to this state, there exists a possible underactuated state that $X(g)$ is achieved. The C region was defined so that when a $X_a(x)$ is sampled outside this region, the system is free from any contact with the obstacle. Therefore $X(p)$ represents the coordinate of (x,z) while θ is 0.

For the sampled pseudo goal $X(p)$, we use Pybullet simulation [4] to validate the path from $X(p)$ to $X(g)$. Tolerance was defined to compare the difference between the simulation result and the desired states. At each iteration step for the simulation, a pair of randomly sampled pseudo-goal and final goals are given as input for the simulation module. The simulation will continue to iterate until a pair yields acceptable accuracy or it reaches the given maximum iteration number.

Followed by simulation, a conventional RRT algorithm is applied to find a path from the starting state $X(\text{init})$ to pseudo-goal $X(p)$. We use the typical RRT to connect the initial state and the sampled pseudo goal while the system's actuated state avoids the C region.

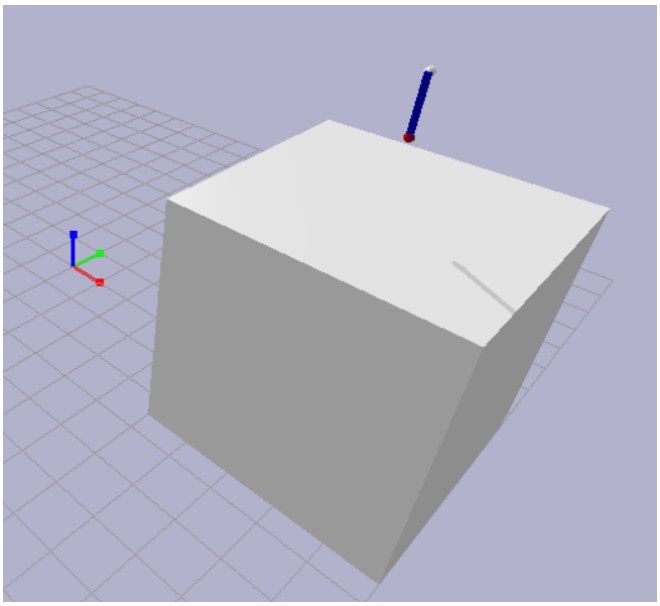


Fig. 5. An example of pseudo goal spawn position with steady state

To summarize, the algorithm:

1) iterates a pseudo-goal $X(p)$, calls simulation to validate its path to $X(g)$;
2) iterates new $X(p)$ if the validation fails;
3) samples actuated states $X_a(x)$, plans a contact-free path towards $X(p)$
4) combines the path towards $X(p)$ and the path from $X(p)$ to $X(g)$ as the final planned path

Algorithm 1 Underactuated RRT method

Input: initial state $X(\text{init})$, goal state $x_{g,a,ua}$, obstacle C, step
Output: T, consisted of states X
for $i \in \{1, ..., N\}$ **do**
    $X(p) \leftarrow$ RandomSampleGoal($X(g)$, C)
    if Pybullet($X(p)$, $X(g)$) **then**
        Break
    end if
end for
Return $X(p)$
T.init($X(\text{init})$)
for $i \in \{1, ..., N\}$ **do**
    $X_{rand} \leftarrow$ RandomSample()
    $X_{near} \leftarrow$ FindNear(X_{rand}, T)
    $X_{new} \leftarrow$ StepLength(X_{new}, step)
    T.add.node(X_{new})
    T.add.edge(X_{near}, X_{new})
end for
Return T

V. EXPERIMENTS AND RESULTS

Experiments on the algorithm were conducted using python and Pybullet simulation packages. Firstly, the configuration space was initialed in the PyBullet environment as shown in Fig. 6. A model of the pendulum was built to represent the underactuated system. Only two dimensions of 3D space were considered, which are x and z. As shown in Fig. 6, the pendulum is free to move in Cartesian space along the directions of x and z. The underactuated system is θ represented by the angle between the vertical axis and the orientation of the blue pendulum. The obstacle is shown as the cube. In this task scenario, the position of the end tip of the pendulum (shown as a red disk) is computed by the full states X_{ua} and X_a. The value output of simulation for x and z is used to evaluate if the path is valid. x_g and z_g can be computed by $X_{ua}(g)$ and $X_a(g)$. The tolerance between (x,z) and (x_g,z_g) indicates whether the system each to the goal. If not, a new pseudo goal $X(p)$ will be sampled until the tolerance is within the range or the maximum number of iterations is reached.

Fig. 7 shows the result of the python RRT algorithm. The obstacle is represented as the red-line square boundary. The green line represents the path connected by nodes, while the yellow line represents the tree branch that was not selected as the path. The blue triangle indicates the desired (x_g,z_g) of the goal, which is the position of the pendulum end tip. Black dots represent the final simulation result of the positions for pendulum tip. The yellow dot represents actuated states of the pseudo goal $X(p)$. The purple trajectories represent the output path of the base link (white disk) in the simulation. When the simulation starts, the pendulum system will be spawned with given states of $X(p)$. Then the simulation starts by using the path between $X_a(p)$ and $X_a(g)$ as the trajectory input, As shown in Fig. 7 due to the contact with the obstacle, the purple output path deviates from the green path between the yellow dot and green dot. In this particular case, it took Algorithm 1 one iteration to find the acceptable $X(p)$ that satisfied the required tolerance of distance between actual $X_{ua}(g)$ and desired $X_{ua}(g)$, and hence there is one purple path showing in the plot.

table	total test	success	avg time (s)
RRT-underactuated	20	20	17.4
RRT-baseline	20	1	22.1

TABLE I
SUMMARY OF RESULTS

Experiments using both RRT-underactuated and conventional RRT (RRT-baseline) were conducted using this test scenario described above, and the results were recorded as shown in Table I. As stated above, the resulting path from the RRT-baseline planner is given to the Pybullet simulation to verify if the final goal is reached in both actuated and underactuated state space. It is not surprising that RRT-underactuated worked well for this task while the

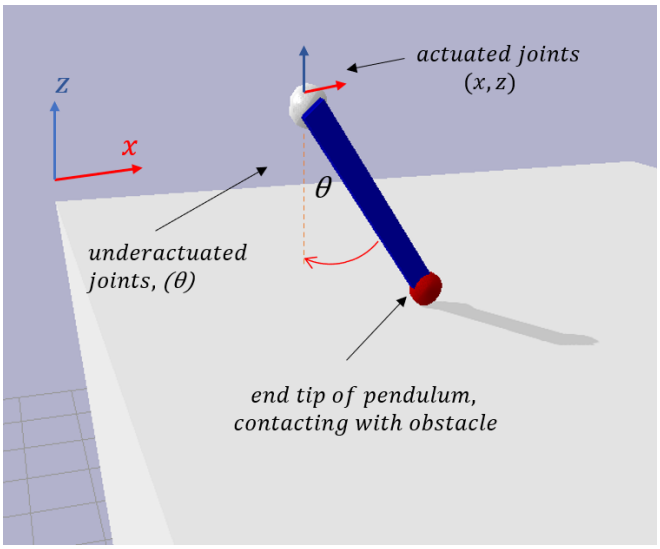


Fig. 6. An illustration of pendulum model

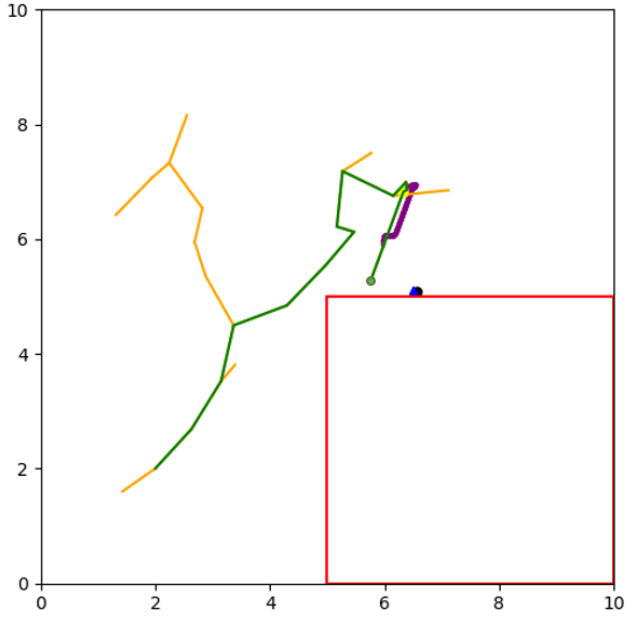


Fig. 7. RRT tree expansion

RRT-baseline struggled to find a path. It is expected that the performance of the planner will be greatly affected by the task scenario, such as the arrangement of obstacles. Furthermore, both RRT algorithms have goal-zoom settings, which usually make the algorithms converge faster. However, it is unclear if it has any impact on their performance over success rate. One of the difficulties of the comparison is that the RRT-baseline and RRT-underactuated have different iteration mechanisms. In a traditional RRT, the iteration will continue until the goal is reached or the maximum iteration number is reached. However, since the simulation result for the RRT baseline is only generated after the planner reaches the goal in actuated state space for the RRT baseline, the

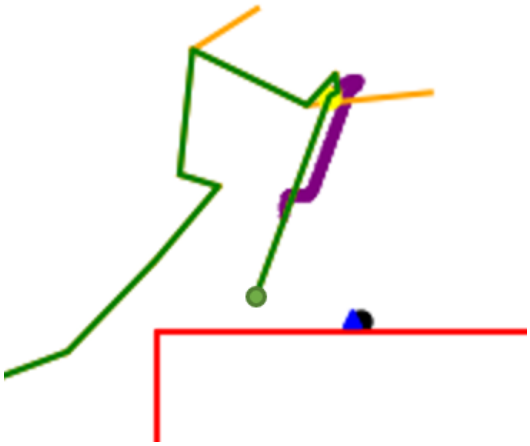


Fig. 8. RRT tree expansion, zoomed in

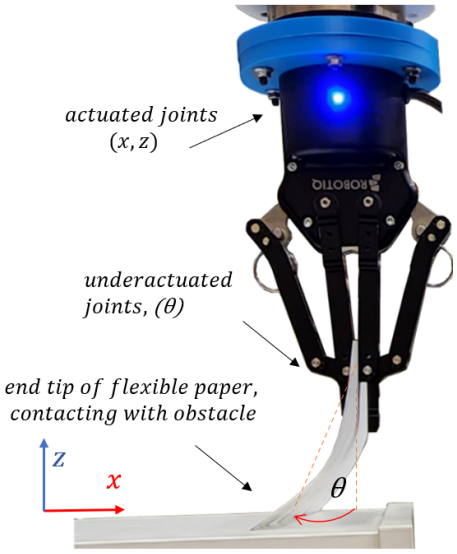


Fig. 9. Oversimplified underactuated model of paper

result of failing to reach the goal in the underactuated state space does not make the algorithm iterate more samples. Hence as shown in the result, the time consumption for the RRT baseline is not un-proportionally large. An alternative RRT baseline can be proposed to tackle this issue by setting the RRT tree one step back. As stated above, the path between $X(p)$ and $X(g)$ was not the focus of this paper. In the future, more work will be done on this aspect, especially when the C region gets large and complex so that a naive straight line path between $X(p)$ and $X(g)$ will no longer work.

VI. Implement on Robot

A simple demonstration using the UR5 robot arm was also performed using the same algorithms. As shown in Fig 10, the UR5 robot is performing a similar task as described in the experiments and results section. In this demo, the end effector holding a paper becomes the underactuated system. Its actuated states x and z were driven by the Cartesian motion of the robot arm. Fig 10 (a) shows the underactuated system achieved desired state by utilizing contact of the shelf surface. Fig 10 (b) shows the pseudo goal at a steady state. And (c) shows that with the same gripper position as (a), the paper's end tip is at an entirely different position. Hence same actuated states $X_a(x)$ lead to different $X_{ua}(x)$. It is worth mentioning that the deformable paper was not specially modeled in the Pybullet simulation for the work of this paper. The focus of the demo was to show the application of utilizing contact and steady-state assumption in the real world. However, future work should address the Sim2Real aspect of this RRT method.

VII. Conclusion and Future Work

This paper proposes a two-step RRT algorithm for underactuated systems' path planning. Our method divides the actuated state space into two spaces and their connection point (the pseudo goal). The first space is the contactless space, where the typical RRT generates the first half path to connect the initial state to the pseudo goal, and in the contact-possible space, PyBullet validates the rest of the path. Our simulations and experiment deal with a contact-rich task scenario and show a high success rate with efficient computation compared to the typical RRT.

For future work, more task scenarios need to be tested by introducing more complicated system models and obstacles. A more comprehensive evaluation of performance needs to be developed, and more data should be collected. Additionally, more comparisons need to be done to validate the performance of our method with other methods, such as optimization-based and kinodynamics-based methods.

References

[1] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[3] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[4] E. Coumans, "Bullet physics simulation," in *ACM SIGGRAPH 2015 Courses*, 2015, p. 1.

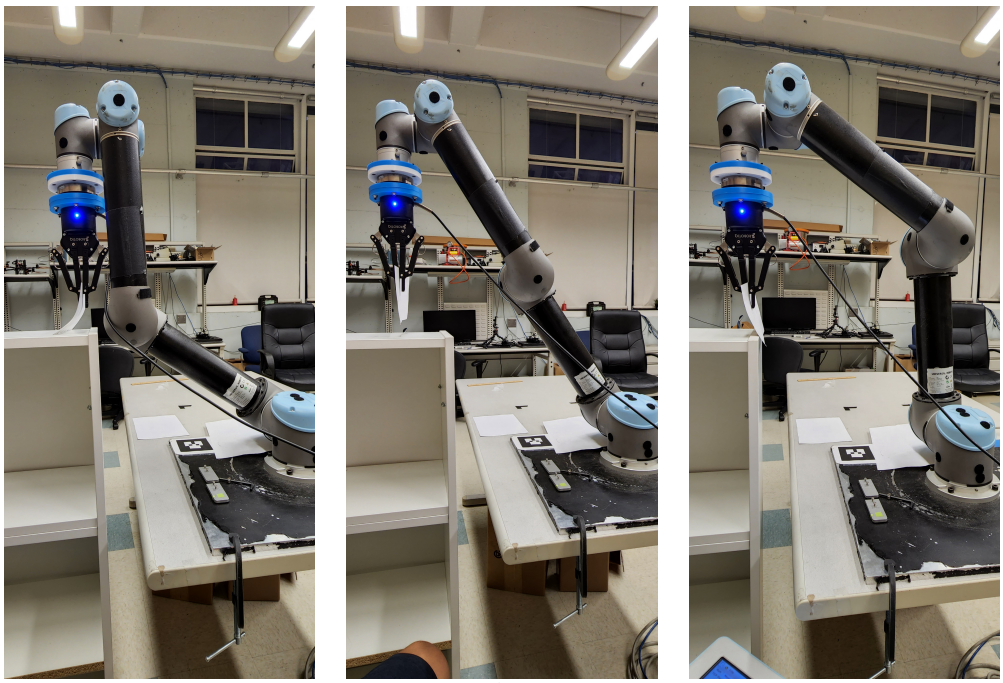
[5] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[6] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[7] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4914–4919.

[8] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, 2012, pp. 137–144.

[9] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.



(a) (b) (c)

Fig. 10. UR5 demonstration using a piece of print paper

[10] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[11] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[12] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[13] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[14] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," in *Robotics Research*. Springer, 2020, pp. 523–539.

[15] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "Trajectotree: trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8262–8268.

[16] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2730–2736.

[17] A. Wu, S. Sadraddini, and R. Tedrake, "R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4245–4251.