Sampling-Based Model Predictive Control for Contact-Rich Manipulation

Sharanya Venkatesh*, Bibit Bianchini*, Alp Aydinoglu, Michael Posa

***Abstract*— For robotic systems to be useful in everyday lives, they need to perform dexterous manipulation tasks in messy human environments. Enumerating all the possible physical interactions a robot can make with its environment and other objects is intractable, necessitating ways for robots to resolve how to perform tasks online, and in real time. However, on-the-fly globally optimal control is infeasible due to the high required computational load. Workarounds that preempt the computational cost by training or computing in advance lack the necessary ability to recover when real events are not foreseen by previous data or offline plans. Recent advances in model predictive control (MPC) leverage significant model simplification but demonstrate contact-implicit controllers capable of real time rates. These local model-based controllers require extra assistance for even simple manipulation tasks. We demonstrate using parallel local model-based control methods, ultimately making hybrid decisions at two levels: a contact-implicit locally optimal controller, wrapped by a layer which considers the controller's performance from a sampling of end effector configurations. Our hierarchical controller is shown to perform simulated rolling tasks with a sphere and a jack at rates with expected real-time capability.**

I. Introduction

High-quality, dexterous manipulation in the real world is challenging because it can require significant computation, rendering on-the-fly globally optimal control infeasible. One workaround is to train a learned policy that can have fast inference during deployment [1], albeit requiring training time and a large dataset. Planning [2], [3], [4] can be used as a dataset-free alternative, spending time offline to compute high quality trajectories that reach global optimality. Some recent online model predictive control (MPC) contributions work effectively because they rely on pre-computed reference trajectories [5]. Planning times for dynamics model-based methods may be orders of magnitude faster than training times for learning-based methods, though still prohibitively long to use in real time. Further, many recent works in this area restrict planning to 2D [6], [7] due to the complexity of the 3D problem. Additionally, executing pre-planned trajectories is non-trivial. Manipulation in the wild often quickly deviates from optimal plans, necessitating online control strategies that may never recover the original plan.

Alternative to leveraging offline training or planning, recent works [8], [9] have made progress towards the global optimization problem by instead using local linear complementarity system (LCS) approximations to the global system to formulate a more tractable MPC problem. These methods have zero reliance on pre-computed reference trajectories and thus are veritable real-time, on-the-fly controllers. However, LCS models linearize geometry, preventing pushing in opposite directions. In practice, these LCS-based controllers bypass this limitation via the use of heuristics to guide the end effector to the right location, but this is restrictive and limited in generalization.

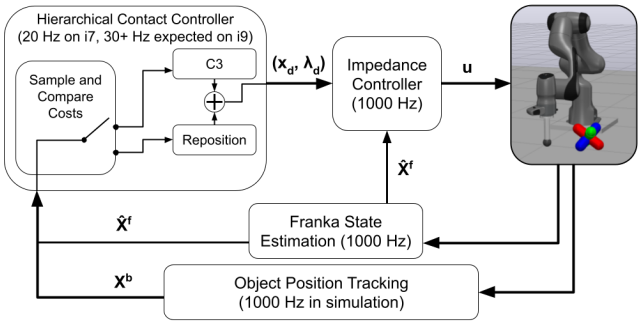

Fig. 1: A block diagram depicting our hierarchical controller (top left block) that performs contact implicit, real-time control for manipulation. See Section III-D for more information on run rates.

In this work, we add a sampling layer on top of the local LCS-based control (Figure 1). The sampling broadens the controller's local view into a global sampling of local linearizations from different end effector positions. At each sample, we compute the cost of the sample's local LCS-based optimization problem and choose to relocate the robot's end effector when there is ample cost benefit. Our hybrid solution aims to perform real-time manipulation of arbitrary geometry by combining global sampling with real-time control on local linearizations, an effective approximation of the global multi-contact MPC problem.

A. Contributions

In this paper, we contribute:

- A novel combination of global sampling with local control for multi-contact manipulation.
- Measurements that demonstrate our approach is capable of real-time rates.
- Preliminary presentation of two simulation experiments to validate the efficacy of our approach.

Given this is work in progress, we emphasize the following as future work in this line of research:

- Exploration of different sampling strategies.

*The first two authors contributed equally to this work.

This work was supported by the National Defense Science and Engineering Graduate Fellowship. This material is based upon work supported by the National Science Foundation under Grant No. FRR-2238480

All authors are with the General Robotics, Automation, Sensing, and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, PA 19104, {sshastry, bibit, alpayd, posa}@seas.upenn.edu

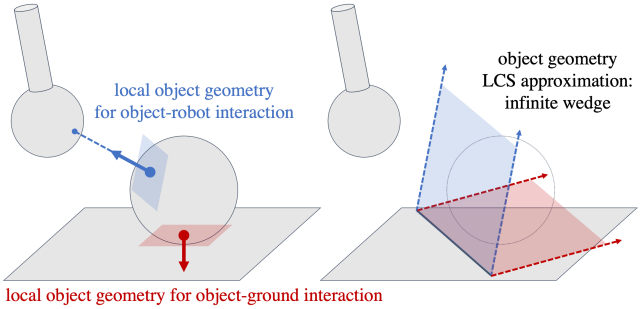

Fig. 2: Left: A spherical end effector approaching a spherical object on a flat table. Right: The linear complementarity system (LCS) approximation of a geometry converts each contact into a hyperplane tangent to the contact surface.

- Further tuning of our simulation experiments to improve performance.
- Demonstration on hardware.

II. Background

A. Linear Complementarity System

We use linear complementarity systems (LCS) as local representations of multi-contact systems. An LCS is a differential/difference equation coupled with a variable that is the solution of a linear complementarity problem (LCP). Generically, an LCP is defined as:

Definition 1: Given a vector $q \in \mathbb{R}^m$, and a matrix $F \in \mathbb{R}^{m\times m}$, the LCP$(q, F)$ describes the following mathematical program:

$$\begin{aligned} \text{find} \quad & \lambda \in \mathbb{R}^m \\ \text{subject to} \quad & y = F\lambda + q, \\ & 0 \le \lambda \perp y \ge 0. \end{aligned}$$

In our context, the vector λ typically represents the contact forces and slack variables, y represents the gap function, and the orthogonality constraint embeds the hybrid structure.

Definition 2: A linear complementarity system describes the trajectories $(x_k)_{k\in\mathbb{N}_0}$ and $(\lambda_k)_{k\in\mathbb{N}_0}$ for an input sequence $(u_k)_{k\in\mathbb{N}_0}$ starting from x_0 such that

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + D\lambda_k + d, \\ 0 \le \lambda_k &\perp Ex_k + F\lambda_k + Hu_k + c \ge 0, \end{aligned} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $\lambda_k \in \mathbb{R}^{n_\lambda}$, $u_k \in \mathbb{R}^{n_u}$, $A \in \mathbb{R}^{n_x\times n_x}$, $B \in \mathbb{R}^{n_x\times n_u}$, $D \in \mathbb{R}^{n_x\times n_\lambda}$, $d \in \mathbb{R}^{n_x}$, $E \in \mathbb{R}^{n_\lambda\times n_x}$, $F \in \mathbb{R}^{n_\lambda\times n_\lambda}$, $H \in \mathbb{R}^{n_\lambda\times n_u}$ and $c \in \mathbb{R}^{n_\lambda}$.

Vector x_k represents the state and often consists of the generalized positions q_k and velocities v_k. In computing the LCS representation of a given system, all of the quantities A, B, D, d, E, F, H, c are functions of state x_{current}. For a given k, x_k, and u_k, the corresponding complementarity variable λ_k can be found by solving LCP(Ex_k+Hu_k+c, F) (see Definition 1). Similarly, x_{k+1} can be computed using the first equation in (1) when x_k, u_k, and λ_k are known. Given an input u_k and state x_k, we represent the next state with $x_{k+1} = \text{LCS}(x_k, u_k)$.

While certain LCPs may lack or have multiple solutions, for the sake of simplicity, here we assume that the mapping exists and is unique.

B. C3

A common cost metric for evaluating a trajectory $x_{0:N}$ coupled with its contact forces $\lambda_{0:N-1}$ and control inputs $u_{0:N-1}$ is given by

$$\begin{aligned} f_{\text{cost}}(x_{0:N}, \lambda_{0:N-1}, u_{0:N-1}) &= \dots \\ &\sum_{k=0}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k) + x_N^T Q_N x_N \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k + D\lambda_k + d, \\ & Ex_k + F\lambda_k + Hu_k + c \ge 0, \\ & \lambda_k \ge 0, \\ & \lambda_k^T(Ex_k + F\lambda_k + Hu_k + c) = 0, \\ & (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) \in \mathcal{C}, \\ & \text{for } k = 0, \dots, N-1, \text{given } x_0, \end{aligned} \tag{2}$$

where N is the planning horizon, Q_k, Q_N are positive semidefinite matrices, R_k are positive definite matrices, $\mathcal{C}$ is a convex set (e.g. input bounds, safety constraints, or goal conditions), and $\boldsymbol{x}^T = [x_1^T, \dots, x_N^T]$, $\boldsymbol{\lambda}^T = [\lambda_0^T, \lambda_1^T, \dots, \lambda_{N-1}^T]$, $\boldsymbol{u}^T = [u_0^T, u_1^T, \dots, u_{N-1}^T]$. Prior work [8] developed a real-time, multi-contact controller called Consensus Complementarity Control (C3) that efficiently solves the mathematical optimization problem to minimize this cost:

$$\gamma(x_{\text{current}}, x_{\text{goal}}) = \min_{x_k, \lambda_k, u_k} f_{\text{cost}}(x_k, \lambda_k, u_k), \tag{3}$$

$$x^*, \lambda^*, u^*(x_{\text{current}}, x_{\text{goal}}) = \arg\min_{x_k, \lambda_k, u_k} f_{\text{cost}}(x_k, \lambda_k, u_k). \tag{4}$$

One control loop of C3 is executed by solving the optimization, applying $u^*[0]$ from (4) to solve the LCS in (1) and obtain x_{k+1}, and sending this desired state to a high-rate impedance controller. This process is repeated in every time step in a receding horizon manner.

III. Methods

C3 has been demonstrated to work well in real scenarios including rolling a ball through a trajectory with a Franka arm and in a variety of simulation examples. However, these demonstrations require additional heuristics to force the end effector out of regions where the local LCS view is antagonistically simplified. These heuristics depend on the system and the goal, and bake in *a priori* knowledge for what regions are more amiable towards allowing C3 to make progress towards the goal.

It is the aim of this work to replace the heuristic-based, scenario-specific repositioning rules with a global sampling-based, geometry-agnostic scheme. A high-level controller trades off current C3 efficacy with future investment: we directly compare the costs from (3) with the end effector in its current location to the end effector at other sampled locations, plus costs on travel and switching states. The high-level controller switches between C3 mode and repositioning mode based on these cost comparisons.

Given the current state x_{current}, goal state x_{goal}, a target number of additional samples to consider n_s, a strategy for

generating the next sample $x_{\text{sample}} = f_{\text{sample}}(x_{\text{current}})$, and a few cost hyperparameters, the algorithm for running our hybrid controller is provided in Algorithm 1. Its output is the next state x_{next} to send to the impedance controller.

A. Generating Samples

The LCS approximation of the object-end effector pair approximates the object's geometry as a hyperplane intersecting and tangent to the object's closest point to the robot's end effector (Figure 2). When the end effector is between the object and the goal, C3 cannot make progress towards the goal by contacting the hypothetical hyperplane (the normal force would have to be negative), so it chooses no contact.

Thus the goal of sampling other end effector locations is to generate other LCS approximations with substantially different hyperplane simplifications. In practice, sampling around the object produces a diverse set of LCS approximations, and quickly one amenable towards making progress is found.

In this work, we generate samples as a **radially symmetric fixed number of samples about the (x, y) position of the object** (in the sphere example) and a **random sampling on a 2D circle or on the surface of a 3D sphere**. For random sampling, any sample to which our algorithm chooses to reposition is stored and pursued while other samples are randomly generated. If lower cost, other samples can supersede the actively pursued sample. Any sampling strategy is defined as $x_{\text{sample}} = f_{\text{sample}}(x_{\text{current}})$.

It remains future work to explore other possible sampling strategies, e.g. using cross-entropy methods. More discussion in Section V.

B. Evaluating Samples

Each of these sample locations is scored based on its C3 cost from (3) plus costs on travel distance and switching to repositioning mode,

$$c_{\text{sample}}(x_{\text{sample}}, x_{\text{goal}}) = \gamma(x_{\text{sample}}, x_{\text{goal}}) + w_{\text{travel}} \cdot \|q_{\text{sample}} - q_{\text{current}}\| + w_{\text{repos}}, \quad (5)$$

with w_{travel} as a travel distance cost weight, w_{repos} as a fixed cost to repositioning, and where the only difference between q_{sample} and q_{current} is the end effector location. w_{repos} exists to ensure switching back to C3 mode once the end effector reaches the end of a repositioning move, if it hasn't already switched modes.

C. Hierarchy

After generating lists of candidate samples and their associated costs, the high-level controller determines whether to run C3 or to reposition. This switch is implemented with hysteresis (of size h) such that if the controller is currently running C3, it switches only if repositioning is better by h, and vice versa. The algorithm for this decision process and actions is provided in Algorithm 1 and depicted schematically in the upper left block in Figure 1.

Algorithm 1 Hybrid C3 with global sampling

Require: $x_{\text{current}}, x_{\text{goal}}, n_s, f_{\text{sample}}(.), w_{\text{travel}}, w_{\text{repos}}, h$, `doing_c3`

$c_{\text{opt}} \leftarrow \gamma(x_{\text{current}}, x_{\text{goal}})$ from (3)
$u_{\text{current}} \leftarrow u^*(x_{\text{current}}, x_{\text{goal}})$ from (4)
$x_{\text{next}} \leftarrow \text{LCS}(x_{\text{current}}, u_{\text{current}})$ from (1)
if `doing_c3` **then**
  $c_{\text{opt}} \leftarrow c_{\text{opt}} - h$
else
  $c_{\text{opt}} \leftarrow c_{\text{opt}} + h$
end if
for $i \in \{1, \ldots, n_s\}$ **do**
  $x_{\text{sample},i} \leftarrow f_{\text{sample}}(x_{\text{current}})$
  $c_i \leftarrow c_{\text{sample}}(x_{\text{sample},i}, x_{\text{goal}})$ from (5)
  if $c_i < c_{\text{opt}}$ **then**
    $c_{\text{opt}} \leftarrow c_i$
    $x_{\text{next}} \leftarrow$ point along curve from x_{current} to $x_{\text{sample},i}$
  end if
end for
return x_{next}

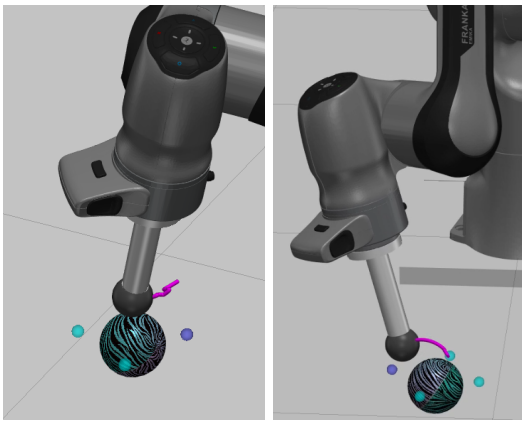

Fig. 3: Manipulating a sphere in C3 (left) and repositioning modes (right). The cyan spheres indicate sampled end effector locations, the dark blue sphere indicates the best sample location, and the pink lines visualize the corresponding end effector motions.

D. Real-Time Rates

On a 12th generation Intel Core i7-12700H with 20 threads, our algorithm for the jack example (Section IV-B) with $n_s = 3$ currently runs at up to 23 Hz. We expect the additional speedup gained by using an off-the-shelf i9 processor will increase our speed to over 30 Hz, a rate at which we are confident our controller can be effective in real time. Results presented here were obtained via running our simulation at < 1 real-time rates that achieve an effective 30 Hz control loop for testing purposes.

IV. Experiments

For both experiments presented herein, we simulate and control a Franka Panda robotic arm using Drake. The arm has a spherical end effector with no additional degrees of freedom. The objects are assumed to be rigid.

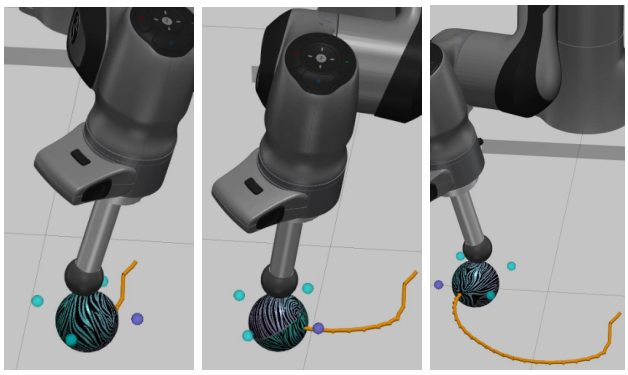

Fig. 4: Manipulating a sphere to track a circular trajectory. The cyan spheres indicate sampled end effector locations, the pink / dark blue sphere indicates the best sample location, and the orange lines visualize the object's traversed trajectory.

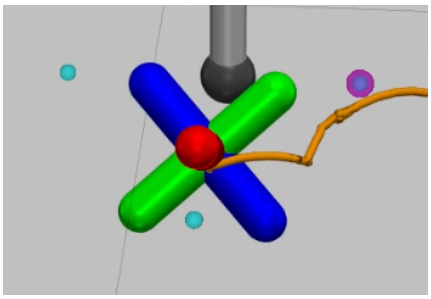

Fig. 5: Manipulating a jack to traverse a trajectory. The object's trajectory is shown by the orange path, sample locations by cyan spheres, and the best sample location demarked by a pink sphere.

A. Trajectory Tracking with a Sphere

This experiment focuses on tracking a circular trajectory using a spherical object. Figure 3 shows the controller running in C3 mode (left) and repositioning mode (right). Our hierarchical controller effectively manipulates the sphere to follow a circular trajectory, as depicted in Figure 4 by the orange path the sphere traversed.

B. Trajectory Tracking with a Jack

In this experiment, we use a more complex non-convex shape. This objective is achieved using a jack, formed by the intersection of three mutually perpendicular capsules. Figure 5 illustrates preliminary results for manipulating the jack to traverse a circular trajectory.

V. Future Work

In addition to refining the performance on our presented examples, we plan on demonstrating our work on hardware. Additionally, in the progress provided in this paper, we used simple sampling strategies which are fairly effective in bringing the end effector to a new location from which it can make further progress to the goal once it is not able to anymore. However, we plan to explore other sampling schema in a future conference paper in this line of work. Here we briefly discuss some considerations for how to generate samples, i.e. how to define $f_{\text{sample}}(x)$.

a) Leveraging Object at Zero Velocity: With every control loop, n_s samples are considered. In practice, we choose n_s to be a small number (typically < 4) to increase the speed of the control loop. Depending on the geometry of the object, there are often repositioning maneuvers during which the object is stationary, and thus previously generated samples and their costs are still valid. We plan on leveraging this and keeping track of valid past samples to help inform the next samples. This could allow n_s to shrink, increasing the speed of the control loop without compromising the breadth of considered samples.

b) Cross-Entropy Sampling: The sample cost calculated in (5) is effectively a scalar field over the 3-dimensional end effector location space. We can use cross-entropy methods to intelligently choose a new sample location x_{sample} based on the costs $\mathbf{c}$ of prior stored samples $\mathcal{S}$.

VI. Conclusion

In this work, we presented a novel combination of a sampling layer with a local LCS-based controller with the ability to perform real-time multi-contact control in simulated examples with no *a priori* specified contact or repositioning scheduling.

Acknowledgment

The authors would like to thank Wei-Cheng Huang for his assistance in and understanding of the C3 code and for facilitating our rate tests.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[2] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason, "Enhancing dexterity in robotic manipulation via hierarchical contact exploration," *arXiv preprint arXiv:2307.00383*, 2023.

[3] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on Robotics*, 2023.

[4] 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.

[5] S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control," *arXiv preprint arXiv:2107.05616*, 2021.

[6] B. Aceituno and A. Rodriguez, "A hierarchical framework for long horizon planning of object-contact trajectories," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 189–196.

[7] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6520–6526.

[8] A. Aydinoglu, A. Wei, and M. Posa, "Consensus complementarity control for multi-contact mpc," *arXiv preprint arXiv:2304.11259*, Apr. 2023.

[9] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3414–3421.