Simultaneous Trajectory Optimization and Contact Selection for Contact-rich Manipulation with High-Fidelity Geometry

Mengchao Zhang[1], Devesh K. Jha[2], Arvind U. Raghunathan[2], Kris Hauser[1]
[1] University of Illinois Urbana-Champaign [2] Mitsubishi Electric Research Laboratories (MERL)

Abstract—**Contact-implicit trajectory optimization (CITO) is an effective method to plan complex trajectories for various contact-rich systems including manipulation and locomotion. CITO formulates a mathematical program with complementarity constraints (MPCC) that enforces that contact forces must be zero when points are not in contact. However, MPCC solve times increase steeply with the number of allowable points of contact, which limits CITO's applicability to problems in which only a few, simple geometries are allowed to make contact. This paper introduces simultaneous trajectory optimization and contact selection (STOCS), as an extension of CITO that overcomes this limitation. The innovation of STOCS is to identify salient contact points and times inside the iterative trajectory optimization process. This effectively reduces the number of variables and constraints in each MPCC invocation. The STOCS framework, instantiated with key contact identification subroutines, renders the optimization of manipulation trajectories computationally tractable even for high-fidelity geometries consisting of tens of thousands of vertices.**

I. Introduction

Humans and other organisms treat contact as a fact of life and utilize contact to perform dexterous manipulation of objects and agile locomotion. In contrast, the majority of current robots avoid making contact with objects as much as possible, and tend to avoid contact-rich manipulations like pushing, sliding, and rolling [5, 7]. Trajectory optimization [10] has been investigated as a tool for generating high quality manipulations, but choosing an effective mathematical representation of making and breaking contact remains a major research challenge. Two general classes of methods are available: hybrid trajectory optimization and contact-implicit trajectory optimization (CITO). Hybrid trajectory optimization divides a trajectory into segments in which the set of contacts remains constant, but it requires the contact mode sequence to be known in advance [17] or explored by an auxiliary discrete search. CITO [16, 12, 13, 15, 14] allows the optimizer to choose the sequence of contact within the optimization loop. CITO formulates contact as a complementarity constraint to ensure that the contact forces can be non-zero if and only if a point is in contact [16]. Although the resulting mathematical programming with complementary constraint (MPCC) [11] formulation is less restrictive than hybrid trajectory optimization, it still requires a set of predefined allowable contact points on the object. Moreover, MPCC rapidly becomes more challenging to solve as the number of complementarity constraints increases, so past CITO applications were limited to a small handful of potential contact points.

This paper introduces the simultaneous trajectory optimization and contact selection (STOCS) algorithm to address the scaling problem in contact-implicit trajectory optimization. To the best of our knowledge, this is the first method capable of optimizing contact-rich manipulation trajectories with high-fidelity geometric representations in 3D. This method paves the way for manipulation planning with raw sensor input, such as point clouds derived from RGBD images, and eliminates the need for geometry simplification.

STOCS applies an infinite programming (IP) approach to dynamically instantiate possible contact points and contact times between the object and environment inside the optimization loop, and hence the resulting MPCCs become far more tractable to solve. This paper presents a novel method, Time-active Maximum Violation Oracle (TAMVO) with spatial disturbance and temporal smoothing, for selecting salient contact points and contact times, which encourages the IP framework to converge quickly toward a feasible solution. We demonstrate the effectiveness and efficiency of STOCS in solving 3D pushing, sliding, pivoting, and rolling tasks with irregular objects and environments, whose models can include up to tens of thousands of vertices.

II. Approach

A. Problem Description

Our method requires the following information as inputs: 1) Object initial pose region: $Q_{init} \subset SE(3)$. 2) Object goal pose region: $Q_{goal} \subset SE(3)$. 3) Object properties: a rigid body $\mathcal{O}$ whose geometry, mass distribution, and friction coefficients with both the environment μ_{env} and the manipulator μ_{mnp} are known. 4) Environment properties: rigid environment $\mathcal{E}$ whose geometry is known. 5) Robot's contact point(s) with the object: c^{mnp}. 6) A time step Δt and number of time steps T in the trajectory.

Our method will output a trajectory τ that includes the following information at time t: 1) Object's configuration: q_t. 2) Object's velocity (angular and translational): v_t. 3) Robot's contact point(s): c_t^{mnp}. 4) Manipulation force: u_t. 5) Object's contact points with the environment: $\tilde{Y}_t$. 6) Contact force at each object-environment contact point: $z(y_t)\ \forall y_t \in \tilde{Y}_t$.

In this paper, we treat all objects and environments as rigid bodies and we assume the contact between the manipulator

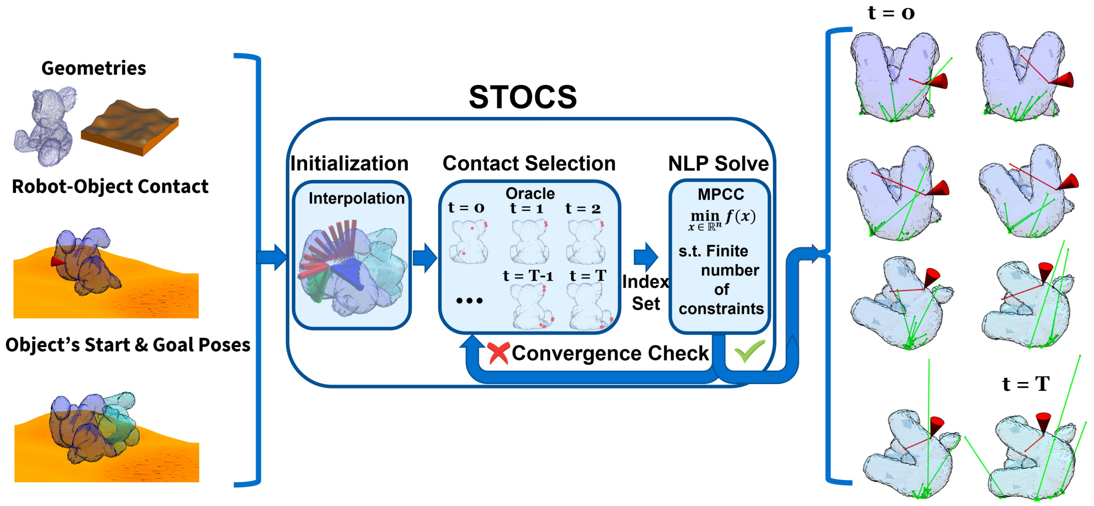


Fig. 1: STOCS accepts as input the high-fidelity geometry of the object (represented by a dense point cloud) and the environment (represented by a signed distance field), the robot's contact point, and start and goal poses of the object (left). The STOCS algorithm first generates an initial trajectory by linearly interpolating between the start and goal poses, and then it iterates between selecting contact points and solving a finite-dimensional MPCC to decide a step direction until the convergence criteria are met (center). As output (right), STOCS produces the pose of the object, active object-environment contact points (green dots) and forces (green lines), and manipulation force (red lines). Nonpenetration, Coulomb friction, complementarity, and quasi-dynamic stability are enforced throughout the trajectory.

and the object is sticking contact.

B. STOCS Trajectory Optimizer

Overall, as shown in Alg.1, STOCS formulates contact-rich trajectory optimization as an *infinite program* (IP), which is a constrained optimization with a potentially infinite set of variables and constraints. It uses an *exchange method* to solve the IP, by wrapping a contact selection outer loop around a finite optimization problem. The inner problem formulates CITO with complementarity constraints and solves an MPCC. We refer the reader to [19, 16] for the detailed format of the MPCC problems solved inside STOCS.

The oracle design is a key component of STOCS. We compare the Maximum Violation Oracle (MVO) used in [19], which adds the closest / deepest penetrating points between the object and the environment at each time step along the trajectory, along with a new method, Time Active Maximum Violation Oracle (TAMVO) with smoothing. TAMVO selects index points more judiciously and only adds the closest / deepest penetrating points at a specific time step.

We denote the index set $\tilde{Y}$ instantiated at the k^{th} outer iteration as $\tilde{Y}^k$. MVO may include index points that do not generate active contact forces during the iteration. For instance, as illustrated in Fig. 2(a), the closest or deepest penetrating points at time step t are typically active only around that specific period.

To address this issue, we introduce TAMVO (Alg. 2). In this refined approach, the index set is no longer the same across time steps. Lines 8–12 identify closest points at each time step. Duplicate points (within threshold ϵ) are excluded in lines 13–18. Given default parameters $n_t = 0$ and $N_s = [0]$, adds only the closest or most deeply penetrating points at the current time step t to $\tilde{Y}_t^k$.

Algorithm 1 STOCS

```
Require: $q_{start}$, $q_{goal}$, $c^{mnp}$
 1: $\tilde{Y}^0 = [\,]$                                   ▷ Initialize empty constraint set
 2: $z_0 \leftarrow \emptyset$                              ▷ Initialize empty force vector
 3: $x_0 \leftarrow$ initialize trajectory($q_{start}$, $q_{goal}$, $c^{mnp}$)
 4: for $k = 1, \ldots, N^{max}$ do
 5:     ▷ Update constraint set and guessed forces $z_k$
 6:     Add all points in $\tilde{Y}^{k-1}$ to $\tilde{Y}^k$, and initialize their forces in
        $z_k$ with the corresponding values in $z_{k-1}$
 7:     Call Oracle to add new points to $\tilde{Y}^k$, and initialize their
        corresponding forces in $z_k$
 8:     $x_k \leftarrow x_{k-1}$
 9:     ▷ Solve for step direction
10:     Set up inner optimization $P^k = P(\tilde{Y}^k)$
11:     Run $S$ steps of an NLP solver on $P^k$, starting from $x_k, z_k$
12:     Set $x^*, z^*$ to its solution, and $\Delta x \leftarrow x^* - x_k$, $\Delta z \leftarrow z^* - z_k$
13:     Do backtracking line search with at most $N_{LS}^{max}$ steps to find
        optimal step size $\alpha$ such that $\phi(x_k + \alpha\Delta x, z_k + \alpha\Delta z; \mu) \leq$
        $\phi(x_k, z_k; \mu)$
14:     ▷ Update state and test for convergence
15:     $x_k \leftarrow x_k + \alpha\Delta x$, $z_k \leftarrow z_k + \alpha\Delta z$
16:     if Convergence condition is met then
            return $x_k$,$z_k$
17: return NOT CONVERGED
```

Choosing only the closest points at the current iterate is potentially not the most ideal choice unless the current iterate is near-optimal. A better choice would anticipate which points are active at the optimum. To address this, we introduce the following two strategies designed to mitigate this issue.

Spatial Disturbance (SD). Recognizing that the current iterate is likely to be in the neighborhood of the optimal solution, the SD approach introduces perturbations to the current solution to add new candidate contact points. Consequently, in lines 9–12 of Alg. 2, q_t is perturbed with disturbance n_s to choose closest points. We choose to perturb along each dimension of

Algorithm 2 Time-Active Maximum-Violation Oracle

```
    Input $q_{0:T}$, $\tilde{Y}^{k-1}$, max object-environment distance $d^*_{max}$,
    contact uniqueness threshold $\epsilon$, time smoothing step $n_t$, spatial
    disturbances $N_s$
    Output $\tilde{Y}^k$
 1: $\tilde{Y}^k \leftarrow \tilde{Y}^{k-1}$
 2: $\tilde{Y}' \leftarrow [[\,]_0, [\,]_1, \dots, [\,]_T]$
 3: for $t = 0, \dots, T$ do
 4:     $y^* = \arg\min_{y \in \tilde{Y}_t} g(q_t, y)$
 5:     $d^* = g(q_t, y^*)$
 6:     if $d^* < d^*_{max}$ then
 7:         add $y^*$ to $\tilde{Y}'[t]$
 8: for $t = 0, \dots, T$ do
 9:     for $n_s \in N_s$ do
10:         $y_s = \arg\min_{y \in Y_t} g(q_t + n_s, y)$
11:         if $g(q_t + n_s, y_s) < d^*_{max}$ then
12:             add $y_s$ to $\tilde{Y}'[t]$
13: for $t = 0, \dots, T$ do
14:     for $t' = t - n_t, \dots, t + n_t$ do
15:         if $0 \le t' \le T$ then
16:             for $y'$ in $\tilde{Y}'[t]$ do
17:                 if $\|y' - y\| > \epsilon \; \forall y \in \tilde{Y}^k_t$ then
18:                     add $y'$ to $\tilde{Y}^k_t$
```

q_t in both positive and negative directions. In 3D, this strategy chooses 12 perturbations accounting for both increases and decreases in x, y, z and $roll$, $pitch$, yaw. An illustration of adding perturbation to rotation in 2D is shown in Fig. 2(e).

Time Smoothing (TS). Considering that the closest points may be active not just at the current time step t, but also during a surrounding interval, in line 14 of Alg. 2, the closest points detected within the adjacent time steps from $t - n_t$ to $t + n_t$, governed by a parameter n_t, are added to the index set of time step t. The effect of using TS is illustrated in Fig. 2(d).

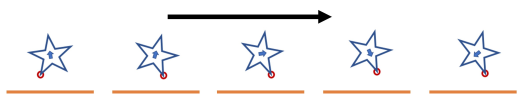
III. Experimental Results

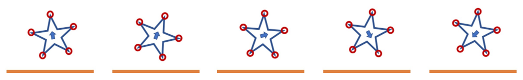
The proposed methods are implemented in Python using the optimization interface and the SNOPT solver [8] provided by Drake [18].

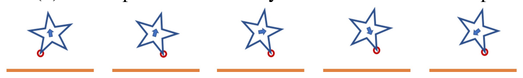
To demonstrate its generalizability, we collect object geometries from the YCB dataset [4], the Google Scanned Objects [6], and 3D models found online [1, 2]. Klampt [3] and the code in [9] are used to find the closest points between two complex shaped geometries.

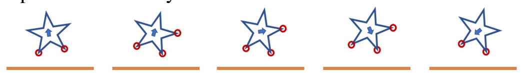
To demonstrate the effectiveness of STOCS in planning with high-fidelity geometric representations, we conduct experiments on ten different objects represented by dense point clouds sampled on the surface of the objects' meshes, and five different environments represented by Signed Distance Field (SDF). The SDFs are calculated offline on a grid that encloses the corresponding environment given a closed polygonal mesh of the environment, and values off of the grid vertices are approximated via trilinear interpolation.

Using STOCS, we plan for pushing, pivoting, rolling and rotating trajectories on these objects. Some planned trajectories are illustrated in Fig. 3, while detailed information regarding the objects' geometries and the solve of the trajectories are


(a) Closest point on the object to the environment at each time step


(b) Index points selected by MVO at each time step


(c) Index points selected by TAMVO without SD and TS at each time step


(d) Index points selected by TAMVO with TS ($n_s = 1$) at each time step

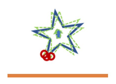

(e) Spatial Disturbance

Fig. 2: Comparing various Oracles. (a) The object's trajectory is depicted as moving from left to right (as indicated by the black arrow) and undergoing clockwise rotation (as indicated by the arrow on the star). (b) In Maximum Violation Oracle (MVO), the closest point on the object is added ot the candidate set at every time step. (c) The Time-Active Maximum Violation Oracle, without Spatial Disturbance and Spatial Disturbances, introduces the closest point only at the current time step. The Time Smoothing technique with $n_s = 1$, demonstrated in (d), extends constraint imposition to the closest points identified at adjacent time steps. (e) presents the Spatial Disturbance technique applied at a specific time step, with only disturbed rotation illustrated.

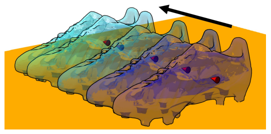
TABLE I: Success rates of STOCS for varying choice of Oracle.

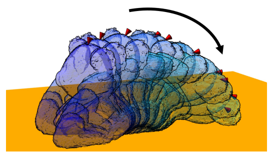
Oracle	MVO	TAMVO	TAMVO+SD	TAMVO+TS	TAMVO+SD+TS
Success Rate	0.75	0.42	0.92	0.58	1.0

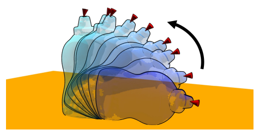
presented in Table II. We set $n_t = 1$ and $N_s = [1e^{-2}]$ as the default parameters for TAMVO. $\Delta T = 0.1\ s$, $\mu_{mnp} = 1.0$ and $\mu_{env} = 1.0$ are used for all the experiments. For all experiments, $T = 10$ is used except in the tasks of pushing a basket on a shelf and sliding a plate on another plate, where $T = 5$ is used. To model the robot manipulator as having a patch contact, 3 to 5 object vertices in the neighborhood of the indicated cone are allowed to be used as contact points.

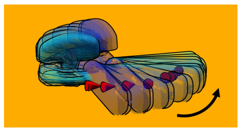
Following the initial assessments, we further evaluated the efficacy of the TAMVO alongside the SD and TS techniques through a set of comparative experiments. These experiments utilized STOCS to plan trajectories for the same set of tasks, with the primary variation being the specific oracle employed in each scenario.

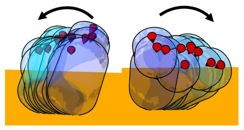
Table 1 presents the success rates of all tested Oracles. The data shows that the MVO achieves a 75% success rate. In contrast, TAMVO without SD and TS exhibits worse performance than MVO; this is particularly evident in 3D scenarios where relying solely on the nearest object-to-environment point is inadequate for fulfilling the object's balance constraints. The SD and TS techniques, introduced to address this challenge,

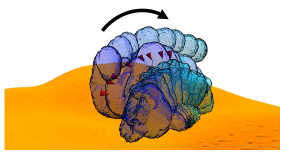

(a) Shoe Pushing on Plane

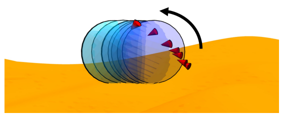

(b) Koala Pivoting on Plane

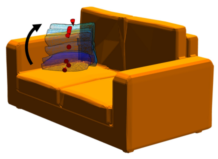

(c) Mustard Pivoting on Plane

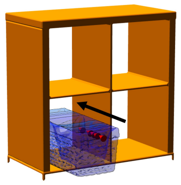

(d) Tool Rotating on Plane


(e) Drug Bottle Rolling on Plane


(f) Koala Pivoting on Curve


(g) Sphere Rolling on Curve


(h) Pillow Pivoting on Sofa


(i) Basket Pushing on Shelf

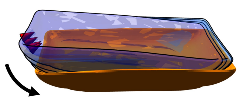
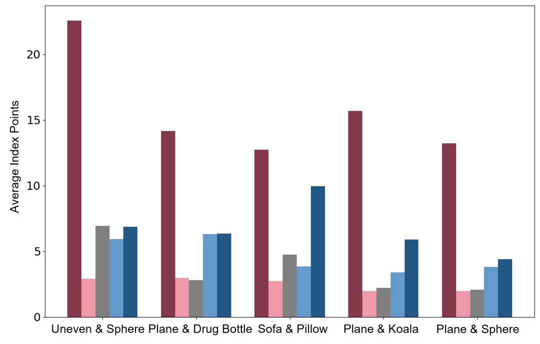
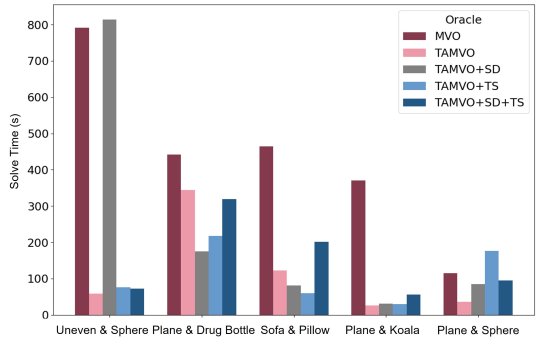

(j) Plate Sliding on Plate

Fig. 3: Trajectories planned by STOCS. Progress along the trajectory is indicated by color (dark to light). The black arrow indicates the object's movement direction. The red cone marks the contact location of the manipulator on the object.



(a) Average Index Points



(b) Solve Time

Fig. 4: The average index points selected at each time step by the different Oracles (a) and the solve time (b) for tasks that were successfully solved by all Oracles.

both demonstrated enhanced performance when combined with TAMVO, surpassing the success rate of TAMVO alone. Also, the integration of SD and TS with TAMVO consistently achieved successful trajectory planning for all tasks.

Figure 4 displays the average number of index points selected at each time step by the different Oracles assessed in our study, focusing on tasks that were successfully solved by all Oracles. As depicted in Fig. 4(a), the MVO selects a larger number of points than TAMVO and all its variations. Notably, TAMVO combined with SD and TS can successfully plan trajectories for all tasks while selecting fewer index points compared to MVO. These findings validate our hypothesis regarding TAMVO: an index point identified at time step t is most valuable within a temporal vicinity of t. Furthermore, these results substantiate our rationale for introducing SD and TS, affirming that a localized exploration in both temporal and spatial dimensions offers a more efficient strategy than incorporating index points identified at distant time steps.

TABLE II: Numerical results of STOCS. Number of points in the object's representation (# Point), solve time (Time), outer iteration count (Outer iters), and average active index points for each iteration (Index points) are reported.

Environment	Object	Task	# Point	Outer iters.	Index points	Time (s)
Plane	Box	Push	764	3	5.75	24.84
	Shoe	Push	17890	6	4.95	144.71
	Koala	Pivot	67359	4	5.89	37.93
	Mustard	Pivot	8424	3	8.58	59.37
	Sphere	Roll	2362	6	4.39	94.42
	Tool	Rotate	8316	6	5.09	99.85
	Drug	Roll	5533	10	6.35	319.72
Curve	Koala	Pivot	67359	9	13.47	676.01
	Sphere	Roll	2362	4	6.86	72.66
Sofa	Pillow	Pivot	13316	7	9.95	201.39
Shelf	Basket	Push	71961	7	23.10	421.30
Plate	Plate	Slide	67283	3	24.67	54.02

Figure 4(b) presents the solve times for STOCS employing various Oracles across all successful tasks. The results indicate that the quantity of index points selected by an Oracle does not necessarily correlate with the solve time. For instance, in the sphere rolling on plane task, TAMVO+SD+TS chooses a larger number of index points than TAMVO+TS, yet the solve time for TAMVO+SD+TS is much faster than that of TAMVO+TS. Similarly, in the case of the drug bottle rolling on plane task, it can be seen that both TAMVO+SD+TS and TAMVO+TS select a comparable number of index points, yet the solve time for TAMVO+TS is much faster than TAMVO+SD+TS. This phenomenon underscores that a larger number of index points does not invariably lead to longer solve time. Although TAMVO+SD+TS provides the best performance in terms of success rate, it is not guaranteed to give the best solve time for all different tasks.

References

[1] URL https://app.gazebosim.org/OpenRobotics/fuel/models/Sofa.

[2] URL https://sketchfab.com/3d-models/burlap-pillow-1ef567278bba4db68ac5488d6b4bc851.

[3] Klampt – Intelligent Motion Laboratory at UIUC. URL http://motion.cs.illinois.edu/klampt/.

[4] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[5] M Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. *Physics-based grasp planning through clutter*. MIT Press, 2012.

[6] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[7] Clemens Eppner and Oliver Brock. Planning grasp strategies that exploit environmental constraints. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4947–4952. IEEE, 2015.

[8] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[9] Kris Hauser. Semi-infinite programming for trajectory optimization with non-convex obstacles. *The International Journal of Robotics Research*, 40(10-11):1106–1122, 2021.

[10] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[11] Zhi-Quan Luo, Jong-Shi Pang, and Daniel Ralph. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[12] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 137–144, 2012.

[13] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[14] Aykut Özgun Önol, Radu Corcodel, Philip Long, and Taşkın Padır. Tuning-free contact-implicit trajectory optimization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1183–1189. IEEE, 2020.

[15] Amir Patel, Stacey Leigh Shield, Saif Kazi, Aaron M Johnson, and Lorenz T Biegler. Contact-implicit trajectory optimization using orthogonal collocation. *IEEE Robotics and Automation Letters*, 4(2):2242–2249, 2019.

[16] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[17] Gerrit Schultz and Katja Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on mechatronics*, 15(5):783–792, 2009.

[18] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[19] Mengchao Zhang, Devesh K Jha, Arvind U Raghunathan, and Kris Hauser. Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning. *arXiv preprint arXiv:2306.06465*, 2023.