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

Zhang, Mengchao; Jha, Devesh K.; Raghunathan, Arvind; Hauser, Kris
TR2024-100 July 16, 2024

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.

RSS Workshop on Frontiers of Optimization for Robotics (RSS Workshop FOR) 2024
Simultaneous Trajectory Optimization and Contact Selection for Contact-rich Manipulation with High-Fidelity Geometry

Mengchao Zhang¹, Devesh K. Jha², Arvind U. Raghunathan², Kris Hauser¹
¹ University of Illinois Urbana-Champaign ² Mitsubishi Electric Research Laboratories (MERL)

Abstract—Contact-implicit trajectory optimization (CITO) is an effective method for planning 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, 6]. 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 complementarity 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 \( O \) whose geometry, mass distribution, and friction coefficients with both the environment \( \mu_{env} \) and the manipulator \( \mu_{mnp} \) are known. 4) Environment properties: rigid environment \( E \) whose geometry is known. 5) Robot’s contact point(s) with the object: \( c_{mnp} \), 6) A time step \( \Delta t \) and number of time steps \( T \) in the trajectory.

Our method will output a trajectory \( \tau \) 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): \( e_t \). 4) Manipulation force: \( u_t \). 5) Object’s contact points with the environment: \( Y_t \). 6) Contact force at each object-environment contact point: \( z(y_t) \forall y_t \in Y_t \).

In this paper, we treat all objects and environments as rigid bodies and we assume the contact between the manipulator
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 $\hat{Y}$ instantiated at the kᵗʰ outer iteration as $\hat{Y}^k$. 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 $\epsilon$) are excluded in lines 13–18. Given default parameters $n_\alpha = 0$ and $N_\alpha = [0]$, adds only the closest or most deeply penetrating points at the current time step $t$ to $\hat{Y}^t$. 

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, $q_k$ is perturbed with disturbance $n_\beta$ to choose closest points. We choose to perturb along each dimension of

\begin{algorithm}
\caption{STOCS}
\begin{algorithmic}
\Require $q_{\text{start}}, q_{\text{goal}}, c_{\text{mnp}}$
\State $\hat{Y}^0 = [\]$ \Comment Initialize empty constraint set
\State $z_0 \leftarrow \emptyset$ \Comment Initialize empty force vector
\State $x_0 \leftarrow \text{initialize trajectory}(q_{\text{start}}, q_{\text{goal}}, c_{\text{mnp}})$
\For {$k = 1, \ldots, N_{\text{max}}$}
\State $\triangleright$ Update constraint set and guessed forces $z_k$
\State $\triangleright$ Add all points in $\hat{Y}^{k-1}$ to $\hat{Y}^k$, and initialize their forces in $z_k$ with the corresponding values in $z_{k-1}$
\State Call Oracle to add new points to $\hat{Y}^k$, and initialize their corresponding forces in $z_k$
\State $x_k \leftarrow x_{k-1}$
\State $\triangleright$ Solve for step direction
\State $\triangleright$ Set up inner optimization $P^k = \hat{P}^k$
\State Run $S$ steps of an NLP solver on $P^k$, starting from $x_k, z_k$
\State Set $x^*, z^*$ to its solution, and $\Delta x \leftarrow x^* - x_k$, $\Delta z \leftarrow z^* - z_k$
\State Do backtracking line search with at most $N_{\text{max}}/S$ steps to find optimal step size $\alpha$ such that $\phi(x_k + \alpha \Delta x, z_k + \alpha \Delta z; \mu) \leq \phi(x_k, z_k; \mu)$
\State $\triangleright$ Update state and test for convergence
\State $x_k \leftarrow x_k + \alpha \Delta x$, $z_k \leftarrow z_k + \alpha \Delta z$
\State if Convergence condition is met then \Return $x_k, z_k$
\State $\triangleright$ Not CONVERGED
\EndFor
\EndAlgorithm
\end{algorithm}
Algorithm 2 Time-Active Maximum-Violation Oracle

```
Input $q_0:T$, $\mathbb{Y}^{k-1}$, max object-environment distance $d_{max}$, contact uniqueness threshold $\epsilon$, time smoothing step $n_t$, spatial disturbances $N_s$
Output $Y^k$
1: $\hat{Y}^k \leftarrow \mathbb{Y}^{k-1}$
2: $Y' \leftarrow [[0], [1], \ldots, [T]]$
3: for $t = 0, \ldots, T$ do
4:     $y' = \arg\min_{y \in Y_t} g(q_t, y)$
5:     $d^* = g(q_t, y')$
6:     if $d^* < d_{max}$ then
7:         add $y'$ to $Y'[t]$
8: for $t = 0, \ldots, T$ do
9:     for $\hat{y} \in \hat{Y}^k$ 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 $Y'[t]$
13: for $t = 0, \ldots, T$ do
14:    for $\hat{t} = t - n_t, \ldots, t + n_t$ do
15:        if $0 \leq \hat{t} \leq T$ then
16:            for $y' \in Y'[\hat{t}]$ do
17:                if $||y' - y|| > \epsilon \forall y \in \hat{Y}^k_t$ then
18:                    add $y'$ to $Y^k_t$
```

$q_t$ 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 provided by Drake.

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]. Klampf [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 presented in Table 1.

<table>
<thead>
<tr>
<th>Oracle</th>
<th>MVO</th>
<th>TAMVO</th>
<th>TAMVO+SD</th>
<th>TAMVO+TS</th>
<th>TAMVO+SD+TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate</td>
<td>0.75</td>
<td>0.42</td>
<td>0.92</td>
<td>0.58</td>
<td>1.0</td>
</tr>
</tbody>
</table>

We set $n_t = 1$ and $N_s = [1\epsilon^{-2}]$ 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,
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.

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.

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.

### TABLE II: Numerical results of STOCS

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

*This table shows the numerical results for STOCS, including the 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.*
REFERENCES