Tuning-Free Contact-Implicit Trajectory Optimization

Aykut Özgün Önol[1], Radu Corcodel[2], Philip Long[3], and Taşkın Padır[1]

***Abstract*— We present a contact-implicit trajectory optimization framework that can plan contact-interaction trajectories for different robot architectures and tasks using a trivial initial guess and without requiring any parameter tuning. This is achieved by using a relaxed contact model along with an automatic penalty adjustment loop for suppressing the relaxation. Moreover, the structure of the problem enables us to exploit the contact information implied by the use of relaxation in the previous iteration, such that the solution is explicitly improved with little computational overhead. We test the proposed approach in simulation experiments for non-prehensile manipulation using a 7-DOF arm and a mobile robot and for planar locomotion using a humanoid-like robot in zero gravity. The results demonstrate that our method provides an out-of-the-box solution with good performance for a wide range of applications.**

I. Introduction

The introduction of contacts into a trajectory optimization problem leads to non-smooth dynamics and thus precludes the use of gradient-based optimization methods in a variety of robot manipulation and locomotion tasks. Therefore, much work has focused on mitigating the discrete nature of contacts by developing appropriate models which enable the optimization to reason about contacts. For this purpose, [1, 2] propose to solve a nonlinear program with complementarity constraints. In [3], complementarity- and penalty-based contact models are used in a similar numerical scheme. [4, 5] present hierarchical strategies to increase the computational efficiency through warm starting. In [6, 7], the complementarity condition is relaxed, and methods are presented to improve the integration accuracy of the dynamics. Mordatch et al. [8, 9] solve a convex program with soft constraints that model the dynamics with contacts by compromising physical realism. In [10], a time-stepping scheme with a smoother variant of the complementarity constraints is proposed, and iterative linear quadratic regulator (iLQR) [11] is employed to solve the problem near real-time. Similarly, [12] proposes a bi-level optimization based on iLQR with implicit hard-contact constraints. iLQR is also used in [13, 14] but with an explicit smooth contact model, in which the contact force is a function of the distance, so that dynamic motions for a quadruped robot can be planned in real-time.

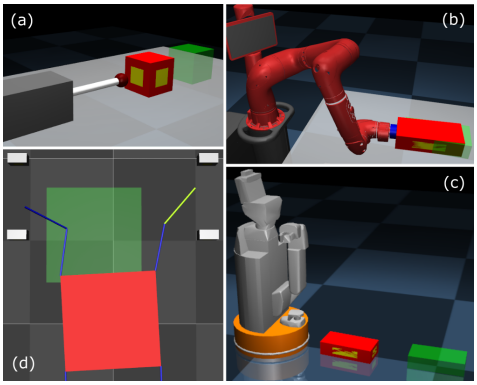

Fig. 1. Applications considered in this study: (a) 1-DOF pusher-slider system for a visual analysis of the problem, (b) a 7-DOF robot arm pushing a box, (c) a mobile robot pushing a box, and (d) locomotion in zero gravity.

While smooth models facilitate convergence, they also lead to physical inaccuracies and are quite difficult to tune. With this in mind, we previously proposed a variable smooth contact model (VSCM) [15] in which virtual forces acting at a distance are injected to the underactuated dynamics with rigid-body contact mechanics. The virtual forces are exploited to discover contacts and minimized throughout the optimization. Consequently, physically-feasible motions are obtained while maintaining fast convergence. Additionally, shooting methods, such as iLQR, are typically sensitive to the initial guess [16]; thus, in [17], we proposed a variant of the successive convexification algorithm that was originally proposed in [18, 19] and showed that it outperforms iLQR for a contact-implicit trajectory optimization (CITO) problem. The use of the VSCM and the successive convexification algorithm (SCVX) significantly mitigates the sensitivity to the initial guess and the burden of tuning by reducing the number of tuning parameters to one, namely a penalty on the virtual stiffness. Nevertheless, it may be required to tune this penalty when the task or the robot is changed; and without extra tuning, abrupt changes may occur in the planned motions even with minor task modifications. Moreover, the resulting contacts are usually impulsive due to the structure of the contact model.

In order to address these issues, we introduce a penalty loop approach for CITO that is analogous to state-of-the-art trajectory optimization methods for collision avoidance such as TrajOpt [20] and GuSTO [21]. In these methods, the penalty on constraints is gradually increased so that the optimization can be initialized with an infeasible trajectory that is in collision. The robot links are eventually pulled

This material is partially based upon work supported by National Science Foundation under Grant Nos. 1451427, 1544895, 1928654. The contribution outlined in this paper was implemented while A.Ö. Önol was an intern at Mitsubishi Electric Research Labs, and R. Corcodel's work was fully supported by Mitsubishi Electric Research Labs. The authors would like to thank Scott Jordan of UMass Amherst and Carlos Jose Nohra Khouri of Carnegie Mellon University for helpful discussions.

[1]Institute for Experiential Robotics, Northeastern University, Boston, MA [2]Mitsubishi Electric Research Labs, Cambridge, MA [3]Irish Manufacturing Research, Dublin, Ireland {onol.a, t.padir}@northeastern.edu, corcodel@merl.com, philip.long@imr.ie

out of collisions by following the gradients. For the CITO problem, an infeasible trajectory corresponds to a motion that completes the task by using non-physical forces that act from a distance, namely the virtual forces. Hence, we develop a method that automatically adjusts the penalty on the relaxation parameters until a motion that completes the task using only physical forces is found. In the CITO case, the solution can be further improved by explicitly exploiting the contact information from the relaxation. In other words, the residual virtual forces indicate the position, time, and magnitude of contact forces required to complete the task. For this purpose, we develop a computationally-cheap post-processing step that improves the solution.

We consider non-prehensile manipulation applications using a 1-degree-of-freedom (DOF) pusher, a 7-DOF arm, and a holonomic mobile robot and a planar locomotion application in zero gravity, see Fig. 1. We test the proposed approach for various goal positions to demonstrate the robustness of our framework. In all cases, the exact same configuration of the pipeline is used with a trivial initial guess, in which the robot stands still. To the best of our knowledge, this is the first attempt to generalize CITO which is the main contribution of this paper to the theory of optimization-based planning of contact-interaction trajectories.

II. Methodology

A. Dynamic Model

The dynamics of an underactuated system with n_a actuated DOF and n_u unactuated DOF, subject to external forces due to frictional rigid-body contacts and virtual forces generated by the contact model is given by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}_a^T\tau + \mathbf{J}_c^T(\mathbf{q})\lambda_c + \mathbf{S}_u^T\lambda_v, \tag{1}$$

where $\mathbf{q} \triangleq [\mathbf{q}_a^T, \mathbf{q}_u^T]^T \in \mathbb{R}^{n_a+n_u}$ is the configuration vector; $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(n_a+n_u)\times(n_a+n_u)}$ is the mass matrix; $\mathbf{c}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n_a+n_u}$ represents the Coriolis, centrifugal, and gravitational terms; $\mathbf{S}_a = [\mathbb{I}_{n_a\times n_a}\ \mathbf{0}_{n_a\times n_u}]$ is the selection matrix for the actuated DOF and $\mathbf{S}_u = [\mathbf{0}_{n_u\times n_a}\ \mathbb{I}_{n_u\times n_u}]$ is the selection matrix for the unactuated DOF; $\tau \in \mathbb{R}^{n_a}$ is the vector of generalized joint forces; $\lambda_c \in \mathbb{R}^{6n_c}$ is the vector of generalized contact forces at n_c contact points and $\mathbf{J}_c(\mathbf{q}) \in \mathbb{R}^{6n_c\times(n_a+n_u)}$ is the Jacobian matrix mapping the joint velocities to the Cartesian velocities at the contact points and $\lambda_v \in \mathbb{R}^{n_u}$ is the vector of generalized contact forces on the unactuated DOF generated by the contact model. In this paper, for n_f free bodies in $SE(3)$ (*e.g.* objects or the torso of a humanoid), we set $n_u = 6n_f$. The state of the system is represented by $\mathbf{x} \triangleq [\mathbf{q}^T\ \dot{\mathbf{q}}^T]^T \in \mathbb{R}^n$ where $n = 2(n_a+n_u)$.

It should be noted that there are two types of contact mechanics in this system: (*i*) frictional contact forces due to physical contacts in the simulated world (*i.e.* contacts detected by the physics engine) which are effective on all DOF; and (*ii*) virtual forces due to the contact model which are exerted only on the unactuated DOF, as explained in II-B.

In this study, the generalized joint forces are decomposed as $\tau = \tau_u + \tilde{\mathbf{c}} - \tilde{\mathbf{J}}_c^T\tilde{\lambda}_c$, where $\tilde{\mathbf{c}}$, $\tilde{\mathbf{J}}_c$, and $\tilde{\lambda}_c$ are the estimations of $\mathbf{c}(\mathbf{q},\dot{\mathbf{q}})$, $\mathbf{J}_c(\mathbf{q})$, and λ_c; and $\tau_u \in \mathbb{R}^{n_a}$ is the vector of control variables associated with the joint forces. This helps to center the optimization problem in terms of the joint forces by directly relating the control term to the acceleration.

B. Contact Model

The contact model considers n_p pairs of predefined contact candidates on the robot (*e.g.* end-effector links) and in the environment (*e.g.* the surfaces of an object). For each contact pair, the magnitude of the virtual force normal to the surface $\gamma \in \mathbb{R}_+$ is calculated by $\gamma(\mathbf{q}) = ke^{-\alpha\phi(\mathbf{q})}$ using the virtual stiffness $k \in \mathbb{R}_+$, the signed distance between the contact candidates $\phi \in \mathbb{R}$, and the curvature $\alpha \in \mathbb{R}_+$. The corresponding generalized virtual force acting on the free body associated with the contact pair $\lambda_v \in \mathbb{R}^6$ is calculated by $\lambda_v(\mathbf{q}) = \gamma(\mathbf{q})[\mathbb{I}_3\ -\hat{\mathbf{l}}(\mathbf{q})]^T\mathbf{n}(\mathbf{q})$, where $\mathbb{I}_3$ is 3×3 identity matrix, $\mathbf{l}$ is the vector from the center of mass of the free body to the nearest point on the contact candidate on the robot, $\hat{\mathbf{l}}$ is the skew-symmetric matrix form of $\mathbf{l}$ that performs the cross product, and $\mathbf{n} \in \mathbb{R}^3$ is the contact surface normal. The net virtual force acting on a free body is the sum of the virtual forces corresponding to the contact candidates associated with that body. As a result, the virtual forces provide a smooth relationship between the dynamics of the free bodies and the configuration of the system.

In the VSCM, the virtual stiffness values $\mathbf{k} \in \mathbb{R}^{n_p}$ are decision variables of the optimization. Thus, the vector of control variables is $\mathbf{u} \triangleq [\tau_u^T,\ \mathbf{k}^T]^T \in \mathbb{R}^m$ where $m = n_a + n_p$.

C. Trajectory Optimization Problem

A finite-dimensional trajectory optimization problem for N time steps can be written in terms of state and control trajectories $\mathbf{X} \triangleq [\mathbf{x}_1, ..., \mathbf{x}_{N+1}]$ and $\mathbf{U} \triangleq [\mathbf{u}_1, ..., \mathbf{u}_N]$; final and integrated cost terms C_F and C_I; and lower and upper control and state bounds $\mathbf{u}_L$, $\mathbf{u}_U$, $\mathbf{x}_L$, and $\mathbf{x}_U$:

$$\underset{\mathbf{U}}{\text{minimize}}\ C(\mathbf{X},\mathbf{U}) \triangleq C_F(\mathbf{x}_{N+1}) + \sum_{i=1}^{N} C_I(\mathbf{x}_i,\mathbf{u}_i) \tag{2a}$$

subject to:

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i,\mathbf{u}_i) \text{ for } i = 1,...,N, \tag{2b}$$

$$\mathbf{u}_L \le \mathbf{u}_{1,...,N} \le \mathbf{u}_U,\ \mathbf{x}_L \le \mathbf{x}_{1,...,N+1} \le \mathbf{x}_U, \tag{2c}$$

where $\mathbf{x}_{i+1} = f(\mathbf{x}_i,\mathbf{u}_i)$ describes the evolution of the nonlinear dynamics over a control period i.

Locomotion and non-prehensile manipulation tasks can be defined based on the desired torso/object configurations. In this study, we use a weighted quadratic final cost in terms of the deviations of the position and orientation of the free body from the desired pose, p_e and θ_e:

$$C_F = w_1 p_e^2 + w_2\theta_e^2, \tag{3}$$

where w_1 and w_2 are the weights. To suppress all virtual forces, the L^1-norm of the virtual stiffness variables (*i.e.* an exact penalty method [22]) is penalized in the integrated cost:

$$C_I = \omega||\mathbf{k}_i||_1. \tag{4}$$

The penalty ω is adjusted by the penalty loop algorithm, as described in II-E.

D. Successive Convexification

The original successive convexification algorithm proposed in [18, 19] is based on repeating three main steps in successions: (*i*) linearizing non-convex constraints (*e.g.* the nonlinear dynamics) about the trajectory from the previous succession, (*ii*) solving the resulting convex subproblem in a trust region to avoid artificial unboundedness due to linearization, and (*iii*) adjusting the trust-region radius based on the fidelity of the linear approximations.

The convex subproblem is given by:

$$\underset{\delta\mathbf{x}_1,\ldots,\delta\mathbf{x}_{N+1},\delta\mathbf{u}_1,\ldots,\delta\mathbf{u}_N}{\text{minimize}} \quad L \triangleq C(\mathbf{X}^s+\delta\mathbf{X},\mathbf{U}^s+\delta\mathbf{U}) \tag{5a}$$

subject to:

$$\delta\mathbf{x}_{i+1} = \mathbf{A}_i\delta\mathbf{x}_i + \mathbf{B}_i\delta\mathbf{u}_i \text{ for } i=1,\ldots,N, \tag{5b}$$

$$\mathbf{x}_L \le \mathbf{x}_i^s + \delta\mathbf{x}_i \le \mathbf{x}_U \text{ for } i=1,\ldots,N+1, \tag{5c}$$

$$\mathbf{u}_L \le \mathbf{u}_i^s + \delta\mathbf{u}_i \le \mathbf{u}_U \text{ for } i=1,\ldots,N, \tag{5d}$$

$$||\delta\mathbf{X}||_1 + ||\delta\mathbf{U}||_1 \le r^s, \tag{5e}$$

where $(\mathbf{X}^s,\mathbf{U}^s)$ is the trajectory from the succession s; $\mathbf{A}_i \triangleq \partial f(\mathbf{x}_i,\mathbf{u}_i)/\partial\mathbf{x}_i|_{\mathbf{x}_i^s,\mathbf{u}_i^s}$; $\mathbf{B}_i \triangleq \partial f(\mathbf{x}_i,\mathbf{u}_i)/\partial\mathbf{u}_i|_{\mathbf{x}_i^s,\mathbf{u}_i^s}$; $\delta\mathbf{x}_i \triangleq \mathbf{x}_i - \mathbf{x}_i^s$; $\delta\mathbf{u}_i \triangleq \mathbf{u}_i - \mathbf{u}_i^s$; and r is the trust-region radius. Additionally, virtual controls can be added to this problem to prevent artificial infeasibility due to linearization [19].

The convex subproblem is a simultaneous problem, and therefore has a larger size but a sparse structure, which can be exploited by a suitable solver. After solving the convex subproblem, we apply only the change of controls instead of applying the changes for both states and controls. The state trajectory is then recalculated by rolling-out the dynamics. This modification prevents the accumulation of defects (*i.e.* $f(x_i,u_i)-x_{i+1}$) that may occur when using the penalty approach in the original method as well as improves the convergence speed by allowing larger trust regions in our experiments. As a result, the modified method combines the numerical efficiency of direct methods and the accuracy of shooting methods. Although it does not have a convergence proof, unlike the original algorithm, it is shown to provide fast and reliable convergence for CITO in [17].

E. Penalty Loop

The SCVX is initialized with large virtual stiffness values to enable the optimization algorithm to reason about contacts. With judicious tuning, the virtual forces vanish yielding a motion that solves the task by using only physical contacts as the optimization converges. However, the penalty on the virtual stiffness ω, plays an important role in this process. While a small penalty may result in physically-inconsistent motions due to leftover virtual forces, a motion that completes the task may not be found if the penalty is too large. Although the tuning of this penalty is fairly straightforward, it hinders the generalization of the method for a wide range of tasks and robots. In order to address this issue, we propose a penalty loop approach that adjusts the penalty similar to a subgradient method [23]. Furthermore, the solution is improved after each iteration through a computationally-cheap post-processing stage by exploiting the contact information embedded in the stiffness variables.

The proposed penalty loop approach is summarized in Algorithm 1. In the first iteration, the SCVX is run with a light penalty value. If the solution satisfies the pose constraints, the weight is increased to reduce the stiffness values; otherwise, the solution is rejected and the weight is reduced by the half of the previous change. Then, a post process is performed on the optimal trajectory to attract the robot links associated with the non-zero stiffness values towards the corresponding contact candidates in the environment using a pulling controller, which is outlined in II-F. This process is carried out only if the average stiffness value is below a threshold, which usually holds after the first iteration in our experiments. The position constraint is normalized with respect to initial position error such that the method scales better to different orders of magnitude (meters for locomotion versus centimeters in the manipulation task).

F. Post Process

After solving the SCVX at each penalty loop iteration, the residual virtual stiffness variables indicate the position, timing, and magnitude of forces required to complete the task. To exploit this information, we develop a controller that pulls the contact candidates on the robot associated with non-zero stiffness variables towards the corresponding contact candidates in the environment.

For a contact pair p and a control period i, the pulling force $\mathbf{f}[p,i] \in \mathbb{R}^3$ is calculated from the distance vector $\mathbf{d}[p,i] \in \mathbb{R}^3$ and the associated virtual stiffness value $k[p,i]$:

$$\mathbf{f}[p,i] = k[p,i]\mathbf{d}[p,i]. \tag{6}$$

Here, $\mathbf{d}$ is the vector from the center of mass of the contact candidate on the robot to the point that is offset from the center of the contact candidate in the environment. The offset is initialized at d_0 for the first penalty iteration and reduced for the following iterations by dividing it by the number of successful penalty iterations. This offset helps reaching occluded surfaces in the environment. Alternatively, a potential field approach may be used with repulsive forces on the surfaces with zero stiffness values.

The corresponding generalized joint force vector $\tau_{pull}[i] \in \mathbb{R}^{n_a}$ is calculated by:

$$\tau_{pull}[i] = \textstyle\sum_{p=1}^{n_p} \mathbf{J}_t^T[p,i]\mathbf{f}[p,i], \tag{7}$$

where $\mathbf{J}_t[p,i] \in \mathbb{R}^{3\times n_a}$ is the translational Jacobian matrix for the center of mass of the contact candidate on the robot.

To prevent the pulling force generating abrupt motions for large stiffness values, a damping controller is applied to keep the joint velocities close to the planned motion:

$$\tau_{damp}[i] = \mathbf{K}_v\mathbf{S}_a^T\mathbf{M}(\mathbf{q}[i])\dot{\mathbf{q}}_e[i], \tag{8}$$

where $\mathbf{K}_v \in \mathbb{R}^{n_a\times n_a}$ is a positive-definite gain matrix, $\dot{\mathbf{q}}_e$ is the deviation of the joint velocities from the planned velocities and $\tau_{damp}[i] \in \mathbb{R}^{n_a}$ is the generalized joint forces for damping. This computation can be done very efficiently by using the sparse form of the inertia matrix.

Algorithm 1: Penalty Loop

Input : Initial state vector $\mathbf{x}_1$ and initial control trajectory $\mathbf{U}$.
Output: Optimal state and control trajectories.
Data : Initial penalty value $\omega^1 > 0$, penalty step size $\Delta\omega_s > 0$, initial position error p_e^0, position tolerance $\varepsilon_p > 0$, rotation tolerance $\varepsilon_\theta > 0$, avg. stiffness threshold for post processing $k_{threshold}$, max. stiffness tolerance $\varepsilon_k > 0$.

$j = 1$, $\mathbf{U}^j = \mathbf{U}$
repeat
  Step 1 $(\mathbf{X}, \mathbf{U}) \leftarrow \text{SCVX}(\mathbf{U}^j, \omega^j)$.
  Step 2 Calculate position error p_e, rotation error θ_e, max. and avg. stiffness values k_{max} and k_{avg}.
  if $p_e/p_e^0 \leq \varepsilon_p \wedge \theta_e \leq \varepsilon_\theta$ **then**
    $\Delta\omega^j \leftarrow \Delta\omega_s$
    $\omega^{j+1} \leftarrow \omega^j + \Delta\omega^j$
    if $k_{avg}^j \geq k_{avg}^{j-1}$ **then**
      Reject the solution $\mathbf{U}^{j+1} \leftarrow \mathbf{U}^j$, $k_{avg}^j \leftarrow k_{avg}^{j-1}$, $j \leftarrow j+1$, and go back to **Step 1**.
    end
  else
    $\Delta\omega^j \leftarrow -\Delta\omega^{j-1}/2$
    $\omega^{j+1} \leftarrow \omega^j + \Delta\omega^j$
    Reject the solution $\mathbf{U}^{j+1} \leftarrow \mathbf{U}^j$, $k_{avg}^j \leftarrow k_{avg}^{j-1}$, $j \leftarrow j+1$, and go back to **Step 1**.
  end
  if $k_{avg} < k_{threshold}$ **then**
    Step 3 Apply the pulling controller: $(\mathbf{X}_{pp}, \mathbf{U}_{pp}) \leftarrow \text{PC}(\mathbf{x}_1, \mathbf{U})$.
    Step 4 Perform the hill-climbing search: $(\mathbf{X}_{pp}, \mathbf{U}_{pp}) \leftarrow \text{HCS}(\mathbf{x}_1, \mathbf{U}_{pp}^j)$.
    Step 5 Recalculate p_e.
    if $p_e/p_e^0 \leq \varepsilon_p$ **then**
      Accept the post-processing step: $(\mathbf{X}^{j+1}, \mathbf{U}^{j+1}) \leftarrow (\mathbf{X}_{pp}, \mathbf{U}_{pp})$.
    else
      Reject the post-processing step: $(\mathbf{X}^{j+1}, \mathbf{U}^{j+1}) \leftarrow (\mathbf{X}, \mathbf{U})$.
    end
  else
    Accept the solution: $(\mathbf{X}^{j+1}, \mathbf{U}^{j+1}) \leftarrow (\mathbf{X}, \mathbf{U})$
  end
  $j \leftarrow j+1$
until $j > j_{max} \vee (k_{max} \leq \varepsilon_k \wedge p_e/p_e^0 \leq \varepsilon_p)$;
return $(\mathbf{X}^j, \mathbf{U}^j)$.

In Algorithm 1, applying these two steps is referred as the pulling controller (PC). The inputs of the PC are the initial state vector and the optimal control trajectory obtained from the SCVX. The PC adds the pulling and damping forces to the planned joint forces, *i.e.* $\tau = \tau + \tau_{pull} + \tau_{damp}$, and outputs the resulting control and state trajectories. The PC attracts virtually active robot links to the corresponding contact candidates in the environment to facilitate physical contacts. However, as the distances get smaller, the planned stiffness values may lead to excessively large virtual forces. To prevent that, we perform a naive hill-climbing search (HCS) after applying the PC. In this step, non-zero stiffness values are reduced by the change of the final cost divided by the previous change as long as the nonlinear pose error decreases. This step also helps to suppress virtual forces explicitly, *i.e.* independently of the penalty increase.

III. Applications

A. 1-DOF Pusher-Slider System

For a visual analysis of the problem, we evaluate the proposed approach for a simplistic problem: a 1-DOF pusher-slider system with a single time step of 1 s, see Fig. 1(a). The task is to push the slider 20 cm forward (Task 1a), and there is only one contact pair that consists of the tip of the pusher and the front face of the slider.

B. Pushing with a 7-DOF Arm

Similar to [15, 17, 24], we test the method for pushing a box on a table, see Fig. 1(b). For this application, we consider a 7-DOF Sawyer Robot by Rethink Robotics. In addition to pushing the box forward, which is the only task considered in [15, 17, 24], the method is evaluated for side and diagonal pushes as well, which are typically more challenging as the robot must use the occluded faces of the object. In this environment, there are four contact pairs between the side faces of the box and the cylindrical end-effector flange of the robot. The simulation time is 1 s, and the control sampling period is 0.1 s. Three forward pushing tasks are considered to move the box 5 cm, 10 cm, and 30 cm (*i.e.* Tasks 1b, 2b, and 3b). The goal of these tasks is to show that, even with identical initialization conditions, the method performs reliably for diverse problems, *i.e.* tasks that require gentle contact interactions for slight motions or impulsive motions to move the object out of the robot's workspace. Moreover, we evaluate the method to push the box 10 cm left and right (Tasks 4b and 5b) as well as for a diagonal push to move the box 20 cm forward and 20 cm left (Task 6b).

C. Pushing with a Mobile Robot

Another application studied here is non-prehensile manipulation with a mobile robot. We consider a Human Support Robot (HSR) by Toyota pushing a box using its velocity-controlled, holonomic base, as shown in Fig. 1(c). There are four contact pairs between the side faces of the box and the cylindrical base of the robot. As the translational and rotational velocities are bounded by ± 2 m/s and ± 2 rad/s, a simulation time of 5 s and a control sampling period of 0.5 s are used for this application. A forward pushing task to move the box 50 cm (Task 1c) and two diagonal pushing tasks are considered. It is observed that when the default friction coefficient of the physics engine ($\mu = 1$) is used, the robot heavily relies on the frictional forces for the diagonal pushes, which seems unrealistic. In order to show that the method is capable of avoiding that as well, we repeat these tasks using $\mu = 0.1$. Hence, Tasks 2c and 3c require to move the box 20 cm forward and 20 cm left, and Tasks 4c and 5c require to move the box 30 cm forward and 10 cm right.

D. Planar Locomotion in Zero Gravity

Lastly, the proposed framework is tested for a locomotion application to demonstrate that it can make and break multiple contacts simultaneously. We consider a planar, humanoid-like robot with a prismatic torso and 2-DOF cylindrical arms and legs. The environment has zero gravity which avoids

stability constraints allowing exact utilization of the proposed framework. The task is specified in terms of the torso's desired pose which the robot can reach by using four static bricks in the environment, as shown in Fig. 1(d). However, as the motion is undamped without contacts, the robot must also use contacts to slow down or stop. In this case, there are 8 contact candidates in the environment that are the front and rear faces of the bricks, and 4 contact candidates on the robot that are the end links of the arms and the legs. These candidates are paired based on the sides, so that there are 16 contact candidates in total. Indeed, the legs are never used in the tasks considered here but are added in order to show that the proposed method can immediately reject superfluous virtual stiffness variables as well as its capability of handling many contact pairs. For this application, the simulation time is 2 s, and the control sampling period is 0.1 s. We consider two tasks to move the torso forward 70 cm and 100 cm, *i.e.* Tasks 1d and 2d. Tasks 3d, 4d, 5d, and 6d require to move the torso 70 cm forward and 20 cm right, 100 cm forward and 20 cm left, 90 cm forward and 25 cm left, and 40 cm forward and 20 cm right, respectively.

IV. Simulation Experiments

A. Software Implementation

We simulate the dynamics in MuJoCo [25] since it is found to be advantageous for rigid-body dynamics with contacts [26] and employs a smooth contact model that facilitates the gradient-based optimization [27]. The large-scale, sparse solver SQOPT [28] is used to solve the convex subproblem in (5) which has a large and sparse equality constraint (5b) and a sparse quadratic cost with only 6 non-zero elements. The distance between contact candidates and the nearest point on the robot are calculated by the Gilbert-Johnson-Keerthi algorithm [29] implemented in FCL [30]. The derivatives of the dynamics are approximated by central differences.[1] However, it is possible to make this process more computationally efficient and accurate using the analytical derivatives developed in [31, 32]. Computations are run on a workstation with Intel Core i7-6700K processor.

B. Parameter Values

In all cases, the same parameter values and initial seed are used. The weights in the final cost (3) are $w_1 = 10^4$ and $w_2 = 1$. We use a trivial initial guess with zero joint force values $\tau_u = \mathbf{0}$ (*i.e.* the robot does not move) and large stiffness values $k = 10$ N/m. The upper bound for the stiffness variables is 20 N/m and $\alpha = 10$. In the penalty loop algorithm, the initial penalty value $\omega_s = 0.1$, the penalty step size $\Delta\omega_s = 1.5$, and the average stiffness threshold is 2 N/m. The tolerance values are selected as $\varepsilon_p = 30\%$, $\varepsilon_\theta = 1$ rad, and $\varepsilon_k = 0.1$ N/m, yet they can be tightened as needed for the application. For the post-process stage, $\mathbf{K}_v = 2.5\mathbb{I}$ and the initial offset distance, d_0 is arbitrarily selected as 5 cm but the procedure is not sensitive to this value. In the HCS, the initial step size is 0.1 N/m, and the step size and cost tolerances are 10^{-3} are 10^{-2}.

[1]The code is available at https://www.merl.com/research/license#CITO.

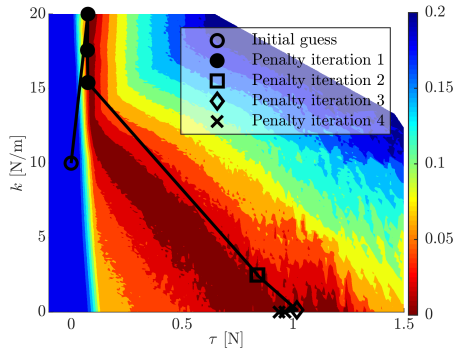

Fig. 2. Change of the cost function over control variables for the 1-DOF pusher-slider system and the progress of the penalty loop algorithm.

C. Visual Analysis

Figure 2 depicts the cost function with respect to the control variables as well as the progress of the proposed method for the 1-DOF pusher-slider system. The large blue band shows that the cost is mostly flat when the physical and virtual forces are zero. In other words, it is difficult for the gradient-based optimization to find a motion that completes the task without a reasonable initial guess. However, the virtual force relaxes the problem by providing steep gradients from the trivial initial guess, and the SCVX accomplishes the task using mostly the virtual force in the first penalty iteration. Moreover, after each iteration the solution is improved substantially by the pulling controller that gets the robot closer to the box and by reducing the virtual stiffness using the HCS. Consequently, a motion that satisfies the task constraints by using only physical forces is obtained.

D. Results

For all applications, except for the pusher-slider system, we evaluate the method for several goal poses in the workspace. Please see the accompanying video[2] for the resulting motions. Here, the progress of the algorithm over penalty loop iterations is presented in terms of the penalty over the relaxation, the average and maximum stiffness values, and the pose error. Figures 3, 4, and 5 demonstrate the results for all applications and tasks. These results show that in all cases, the planner finds a motion that satisfies the pose constraints (*i.e.* the normalized position error is below 30% and the rotation error is below 1 rad) and the stiffness values always converge to zero owing to the exact penalty function in (4). Furthermore, the average stiffness is usually reduced below the threshold in only one iteration.

For the 7-DOF arm tasks, the computation times averaged over penalty loop iterations and tasks for solving the convex subproblems, applying the post process, and calculating the derivatives are 2.05 s, 0.05 s, and 7.23 s, respectively, and the average number of penalty iterations for these tasks is 5. Namely, one can plan a wide range of pushing motions for a 7-DOF arm by using the proposed framework in about 10 s, when the analytic derivatives are integrated.

[2]The video is available at https://youtu.be/_GsuxuQEgPg.

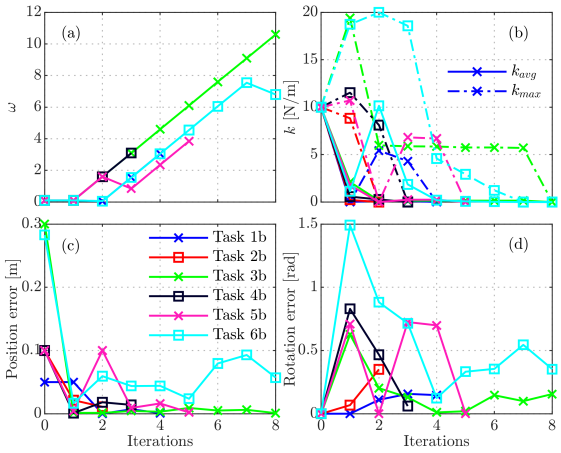

Fig. 3. Changes of (a) the penalty, (b) the average and maximum stiffness, (c) position error, and (d) rotation error over penalty iterations for the 7-DOF robot arm tasks.

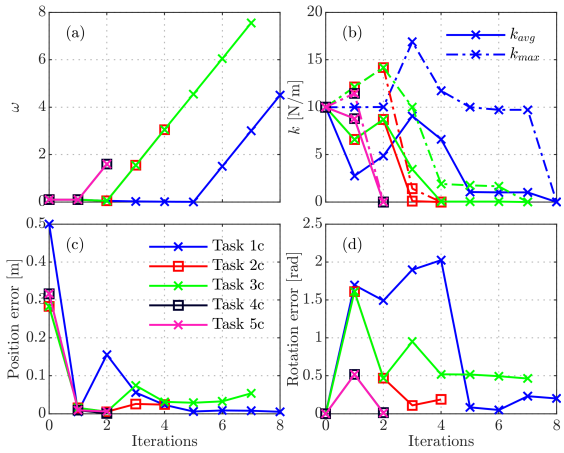

Fig. 4. Changes of (a) the penalty, (b) the average and maximum stiffness, (c) position error, and (d) rotation error over penalty iterations for the mobile robot tasks.

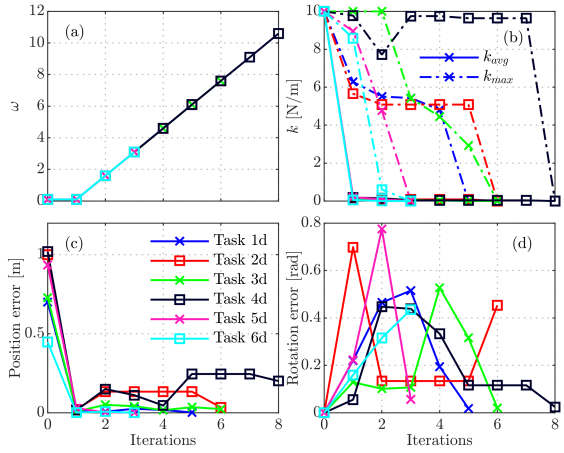

Fig. 5. Changes of (a) the penalty, (b) the average and maximum stiffness, (c) position error, and (d) rotation error over penalty iterations for the planar locomotion tasks.

The video shows that the proposed method can find motions with maintained contacts for sensitive pushing tasks (*e.g.* Task 1b) as well as highly-dynamic behaviors with impact-like contacts (*e.g.* Task 3b), and there are no abrupt changes in the behavior in between. Furthermore, the frictional forces are extensively used for the side pushing tasks although they are not considered in the contact model. This is owing to the fact that once a contact is made, the gradients make it possible to also reason about the tangential forces. Thus, the proposed approach is not limited to the utilization of normal contact forces.

For the mobile robot application, solving the convex subproblems, applying the post process, and calculating the derivatives take 0.91 s, 0.14 s, and 25.81 s in average, and the average number of penalty iterations is 4.6. In this application, the frictional forces are used substantially for the diagonal pushes for the default friction value (*i.e.* Tasks 2c and 4c), as shown in the video. However, lowering the friction coefficient yields similar convergence characteristics, see Fig. 4(a), albeit different motion patterns that mostly rely on normal forces to move the box will occur.

It is noteworthy that the algorithm cannot always find a solution that satisfies the pose constraints in the first iteration (*i.e.* for Tasks 1b, 6b, 1c, 2c, and 3c). However, it overcomes this by reducing the penalty and allowing the virtual stiffness to increase. Additionally, in some cases (*e.g.* Tasks 5b and 6b), the penalty is reduced if the fixed step size $\Delta\omega_s = 1.5$ is too large.

The most computationally expensive application is the planar locomotion with the average times of 38.97 s, 0.57 s, and 51.66 s for solving the convex subproblems, applying the post process, and calculating the derivatives and the average number of iterations of 5.2. The similar computation times for the post process indicates that this step scales well with the size of the problem. However, solving the convex program takes a significantly longer time for the locomotion problem, which can potentially be improved by using a customized convex programming solver. Obviously, the generality of the proposed framework comes at the cost of increased computational cost, and real-time implementation remains unlikely. Nevertheless, it is possible to use the penalty loop approach for offline planning and then executing the resulting trajectory by running SCVX or iLQR in a receding horizon fashion.

V. Conclusion

In this paper, we have presented a generalized contact-implicit trajectory optimization framework that can be used for a wide range of applications and tasks with the same trivial initial seed and without any tuning. The proposed algorithm automatically adjusts the penalty on the relaxation parameters while explicitly improving solutions through a post-processing stage that exploits the contact information indicated by how the relaxation is used in the solution. Our findings indicate that this formulation is generic and applicable for a diverse set of problems as it can generate motions with sensitive contact interactions as well as highly-dynamic motions for a wide variety of tasks and robots. The future work will focus on validating the resulting motions by hardware experiments.

REFERENCES

[1] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using SUMT," in *9th IEEE International Workshop on Advanced Motion Control*. IEEE, 2005, pp. 665–671.

[2] 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.

[3] M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis, "A computational framework for environment-aware robotic manipulation planning," in *Robotics Research*. Springer, 2018, pp. 363–385.

[4] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4636–4641.

[5] T. Marcucci, M. Gabiccini, and A. Artoni, "A two-stage trajectory optimization strategy for articulated bodies with unscheduled contact sequences," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 104–111, 2017.

[6] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2017.

[7] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, 2019.

[8] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[9] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2012, pp. 137–144.

[10] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 4906–4913.

[11] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *Proceedings of 1st International Conference on Informatics in Control, Automation, and Robotics (ICINCO)*, 2004, pp. 222–229.

[12] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[13] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

[14] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[15] A. Ö. Önol, P. Long, and T. Padır, "A comparative analysis of contact models in trajectory optimization for manipulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[16] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[17] A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2447–2453.

[18] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3636–3641.

[19] Y. Mao, M. Szmuk, and B. Acikmese, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint arXiv:1804.06539*, 2018.

[20] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[21] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[22] J. Nocedal and S. J. Wright, "Penalty and augmented Lagrangian methods," *Numerical Optimization*, pp. 497–528, 2006.

[23] N. Z. Shor, *Minimization methods for non-differentiable functions*. Springer Science & Business Media, 2012, vol. 3.

[24] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-implicit trajectory optimization for dynamic object manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[25] 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 (IROS)*. IEEE, 2012, pp. 5026–5033.

[26] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX," in *IEEE International Conference on Robotics and Automation (ICRA), 2015*. IEEE, 2015, pp. 4397–4404.

[27] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6054–6061.

[28] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong, "User's guide for SQOPT 7.7: Software for large-scale linear and quadratic programming," Department of Mathematics, University of California, San Diego, La Jolla, CA, Center for Computational Mathematics Report CCoM 18-2, 2018.

[29] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[30] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[31] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems (RSS)*, 2018.

[32] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.