Fast Multi-Contact Motion Planning Based on Best-Neighbor Search of Contact Sequences

Yuichi Tazaki[†]

***Abstract*— This paper presents a computationally efficient method for planning dynamic multi-contact motion. A low-dimensional dynamical model of a robot equipped with multiple contact points is developed, and a motion planning problem is formulated as a optimal control problem with continuous and discrete variables. An extension to the differential dynamic programming (DDP) framework enables efficient computation of optimal cost for a large number of locally modified contact sequences. Based on this finding, a novel algorithm for multi-contact motion planning is developed, and its performance is evaluated in simulations.**

I. Introduction

In order for a humanoid robot to perform various loco-manipulation tasks in open and changing environments, it must be able to plan and execute multi-contact motion in real time. In traditional motion planning problems such as bipedal walking, only limited number of parts (e.g., feet) make contact, and the order contact switching is fixed and predefined. General multi-contact planning problems, in contrast, pose a number of challenges that make the problem much harder, both theoretically and computationally. Notable challenges are:

- Non-coplanar contact: traditional reduced-order models that assume coplanar contact cannot be applied directly.
- Contact between arbitrary combination of body parts and contact surfaces must be considered.
- The order of contact switching is not specified; instead, it must be found by the planner.

Existing methods for multi-contact planning can be broadly categorized into three classes: search-based, optimization-based, and schedule-based. Search-based methods generates a search tree of contact state transitions, and finds a feasible sequence of contact transitions together with continuous trajectory that satisfies a set of constraints including reachability and contact stability [1], [2], [3], [4], [5], [6]. A major technical issue of this approach is combinatorial explosion of search space. One effective way for reducing computation cost is to define a reference trajectory of the center-of-mass (or the base link) and find valid contacts in the reachable region of this trajectory. [7], [8], [9].

Optimization-based methods formulate multi-contact planning problem as a certain class of optimization problem. Notable classes are mathematical programming with complementarity constraints (MPCC) [10], [11] and mixed integer programming (MIP) [12], [13]. A major issue of MPCC is its poor convergence caused by complementarity constraints. Some studies use soft contact models instead of rigid contact models to improve the convergence [14], [15], [16]. Recent studies investigate application of bi-level optimization technique for reducing computation cost and improving convergence [17], [18], [19].

Schedule-based methods directly parameterize contact timing as decision variables. In this manner, complementarity of contact is satisfied by definition. Winkler et al. formulated a nonlinear optimal control problem in which the time instants of contact switching each end-effector are explicitly defined as decision variables. One shortcoming of this method is that the number of contact switches must be specified in advance.

This paper proposes a novel optimization-based method for multi-contact motion planning. The main contribution of this paper is two-fold. First, a multi-contact linear inverted pendulum mode (mc-LIPM), a natural extension of the conventional LIPM in the multi-contact setup, is proposed and used for trajectory generation. Just like the conventional LIPM, the mc-LIPM yields a closed-form solution of the dynamical equation, which enables planning of long trajectories with a small number of decision variables. Second, a new contact sequence optimization technique based on local search of contact states is developed. The proposed method is conceptually motivated by discrete convex analysis, which claims that for discrete functions with a special convex property, one can obtain a global minimum just by repeating local minimization in a carefully defined neighborhood [20]. At the moment, the proposed method provides no theoretical guarantee for global optimality. Nevertheless, it has been found in simulation experiments that the proposed local-search strategy can find reasonable contact sequences in many practical examples. We also utilize a property of differential dynamic programming DDP that updating the minimum cost when the cost function of only one step is altered can be computed very efficiently. The proposed method is applied to various multi-contact planning problems of a humanoid robot model and verified in simulations.

II. Linear Centroidal Dynamics in Multi-Contact Setup

Let us start from the well-known centroidal dynamics equation shown below.

$$m\ddot{\boldsymbol{p}}(t) = \boldsymbol{f}(t) - m\boldsymbol{g} \tag{1}$$

$$\dot{\boldsymbol{L}}(t) = \boldsymbol{\tau}(t) \tag{2}$$

[†]The author is with Faculty of Engineering, Department of Mechanical Engineering, Kobe University, 1-1 Rokkodai, Nada-ku, Kobe, Japan. tazaki@mech.kobe-u.ac.jp

Here, $\boldsymbol{p}$ is the position of the CoM, $\boldsymbol{L}$ is the angular momentum around the CoM, and $\boldsymbol{f}$ and $\boldsymbol{\tau}$ are the translational and rotational component of the net external wrench, respectively., Moreover, m is the total mass and $\boldsymbol{g}$ is the gravitational acceleration. The robot can make contact with the environment with n_{e} end-effectors. Let us denote the translational and rotational components of the contact wrench applied to the l-th end-effector ($l = 1, 2, \ldots, n_{\mathrm{e}}$) by $\boldsymbol{f}_l$ and $\boldsymbol{\tau}_l$, respectively. Then the total contact wrench is expressed as follows.

$$\boldsymbol{f}(t) = \sum_l \boldsymbol{f}_l(t) \tag{3}$$

$$\boldsymbol{\tau}(t) = \sum_l \left((\boldsymbol{p}_l(t) - \boldsymbol{p}(t)) \times \boldsymbol{f}_l(t) + \boldsymbol{\tau}_l(t) \right) \tag{4}$$

Although the equations (1) and (2) are linear, the equation (4) includes cross product, which makes the overall dynamics nonlinear. To derive a closed-form solution, let us introduce the following restriction on the translational component of contact forces.

$$\boldsymbol{f}_l(t) = m\lambda_l^2(t)(\boldsymbol{p}(t) - \boldsymbol{p}_l(t)) \tag{5}$$

This restriction essentially requires that the contact force acting on each end-effector should always be directed to the CoM. This condition may be difficult to satisfy in general, especially for hand contacts, where the direction of hand contact forces can vary significantly depending on the form of contact (e.g., grasping a hand rail). Nevertheless, we consider that this assumption is acceptable in simple multi-contact cases where the feet make contact mostly with the ground and the hands are used for pushing against walls. Since contact forces are repulsive, we have $\lambda_l(t) \geq 0$. Moreover, let us approximate the inertia matrix around the CoM with a diagonal matrix I. Then the angular momentum around the CoM is given by $\boldsymbol{L} = I\boldsymbol{\omega}$, where $\boldsymbol{\omega}$ is the angular velocity of the base link. The orientation of the base-link is parameterized by Euler angles $\boldsymbol{\theta}$. Here, assuming that the yaw and pitch angles are small enough, $\dot{\boldsymbol{\theta}} \approx \boldsymbol{\omega}$ holds. Furthermore, define $\hat{\boldsymbol{\tau}} = I^{-1}\boldsymbol{\tau}$. As a result, we obtain the following linearized equations of motion.

$$\ddot{\boldsymbol{p}}(t) = \sum_l \lambda_l^2(t)(\boldsymbol{p}(t) - \boldsymbol{p}_l(t)) - \boldsymbol{g} \tag{6}$$

$$\ddot{\boldsymbol{\theta}}(t) = \sum_l \hat{\boldsymbol{\tau}}_l(t) \tag{7}$$

Consider a finite interval of time $[0, T]$ which is further split into N sub-intervals. The k-th interval is given by $[t_k, t_{k+1}]$ ($t_{k+1} = t_k + h_k$). During each interval, the contact state of each end-effector does not change. In other words, contact states may change only at discrete time instants t_k. Furthermore, the stiffness and the contact moment of each end-effector are constant during each interval, and they are denoted by $\lambda_{l,k}$ and $\hat{\boldsymbol{\tau}}_{l,k}$, respectively. Under this assumption, we can analytically integrate (6)(7), which are second-order linear ODEs, to obtain the following.

$$\boldsymbol{p}_{k+1} = \bar{\boldsymbol{p}}_k + C_k\left(\boldsymbol{p}_k - \bar{\boldsymbol{p}}_k\right) + \frac{S_k}{\bar{\lambda}_k}\boldsymbol{v}_k \tag{8}$$

$$\boldsymbol{v}_{k+1} = \bar{\lambda}_k S_k(\boldsymbol{p}_k - \bar{\boldsymbol{p}}_k) + C_k\boldsymbol{v}_k \tag{9}$$

where

$$C_k = \cosh\left(\bar{\lambda}_k h_k\right),\ S_k = \sinh\left(\bar{\lambda}_k h_k\right)$$

$$\bar{\boldsymbol{p}}_k = \frac{\sum \lambda_{l,k}^2 \boldsymbol{p}_{l,k} + \boldsymbol{g}}{\sum \lambda_{l,k}^2},\ \bar{\lambda}_k = \sqrt{\sum \lambda_{l,k}^2}$$

For rotational movement, we have

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{\omega}_k h_k + \frac{1}{2}\sum \hat{\boldsymbol{\tau}}_{l,k} h_k^2 \tag{10}$$

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \sum \hat{\boldsymbol{\tau}}_{l,k} h_k \tag{11}$$

The movement of each end-effector is expressed as

$$\boldsymbol{p}_{l,k+1} = \boldsymbol{p}_{l,k} + \Delta\boldsymbol{p}_{l,k} \tag{12}$$

where $\boldsymbol{p}_{l,k}$ denotes the position of the l-the end-effector and $\Delta\boldsymbol{p}_{l,k}$ denotes its change of position between t_k and t_{k+1}.

Let us define the state variable and the input variable as

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{\theta}_k \\ \boldsymbol{v}_k \\ \boldsymbol{\omega}_k \\ \{\boldsymbol{p}_{l,k}\}_{l\in 1,\ldots,n_{\mathrm{e}}} \\ t_k \end{bmatrix},\quad \boldsymbol{u}_k = \begin{bmatrix} \{\Delta\boldsymbol{p}_{l,k}\}_{l\in 1,\ldots,n_{\mathrm{e}}} \\ \{\lambda_{l,k}\}_{l\in 1,\ldots,n_{\mathrm{e}}} \\ \{\boldsymbol{\tau}_{l,k}\}_{l\in 1,\ldots,n_{\mathrm{e}}} \\ h_k \end{bmatrix} \tag{13}$$

Then, from (8)-(12), the state transition from t_k to t_{k+1} can be expressed as follows.

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{14}$$

III. Formulation of Motion Planning Problem

A. Task-related Costs

The following task-related cost function is defined.

$$\begin{aligned} &L_{\mathrm{task},k} \\ &= w_p^2\|\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{ref}}\|^2 + w_v^2\|\boldsymbol{p}_k - \boldsymbol{p}_k^{\mathrm{ref}}\|^2 \\ &+ w_\theta^2\|\boldsymbol{\theta}_k - \boldsymbol{\theta}_k^{\mathrm{ref}}\|^2 + w_\omega^2\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^{\mathrm{ref}}\|^2 \\ &+ \sum_l \left[w_{p,l}^2\|\boldsymbol{p}_{l,k} - \boldsymbol{p}_{l,k}^{\mathrm{ref}}\|^2 + w_\lambda^2\lambda_{l,k}^2 + w_\tau^2\|\boldsymbol{\tau}_{l,k}\|^2 \right] \end{aligned} \tag{15}$$

Here, $(*)^{\mathrm{ref}}$ are desired values defined by reference trajectories. How to provide reference trajectories of the CoM and those of the end-effectors is problem dependent; for examples shown in Section V, they are defined by cubic spline curves connecting the initial and the goal configurations of the robot. The weight parameters of each term, w_*, must be specified by the designer.

B. Inequality Constraints

The range of the position of each end-effectors relative to that of the CoM is restricted as follows:

$$\boldsymbol{p}_{l,\min} \leq R(\boldsymbol{\theta}_k)^{\mathsf{T}}(\boldsymbol{p}_{l,k} - \boldsymbol{p}_k) \leq \boldsymbol{p}_{l,\max} \tag{16}$$

Here, $R(\boldsymbol{\theta})$ is a rotation matrix equivalent to the Euler angles $\boldsymbol{\theta}$. Similarly, limits on the stiffness and the contact moment are expressed as follows.

$$0 \leq \lambda_{l,k} \leq \lambda_{\max},\ \boldsymbol{\tau}_{l,\min} \leq \boldsymbol{\tau}_{l,k} \leq \boldsymbol{\tau}_{l,\max} \tag{17}$$

These constraints are encapsulated into the following cost function.

$$\begin{aligned} &L_{\mathrm{limit},k} \\ &= w_{\mathrm{lim},p}^2(\| \min(R(\boldsymbol{\theta}_k)^{\mathsf{T}}(\boldsymbol{p}_{l,k} - \boldsymbol{p}_k) - \boldsymbol{p}_{l,\min}, \boldsymbol{0})\|^2 \\ &\qquad + \| \max(R(\boldsymbol{\theta}_k)^{\mathsf{T}}(\boldsymbol{p}_{l,k} - \boldsymbol{p}_k) - \boldsymbol{p}_{l,\max}, \boldsymbol{0})\|^2) \\ &+ w_{\mathrm{lim},\lambda}^2(\min(\lambda_{l,k}, 0)^2 + \max(\lambda_{l,k} - \lambda_{\max}, 0)^2) \\ &+ w_{\mathrm{lim},\tau}^2(\| \min(\boldsymbol{\tau}_{l,k} - \boldsymbol{\tau}_{l,\min}, \boldsymbol{0})\|^2 \\ &\qquad + \| \max(\boldsymbol{\tau}_{l,k} - \boldsymbol{\tau}_{l,\max}, \boldsymbol{0})\|^2) \end{aligned} \tag{18}$$

Here, $\min$ and $\max$ are evaluated componentwise.

C. Contact-related Costs

Let us define a set of boolean variables that expresses the contact state of the end-effectors. Here, $\sigma_{k,l,i}$ ($k = 0, 1, \ldots, N$, $l = 1, 2, \ldots, n_{\mathrm{e}}$, $i = 1, 2, \ldots, n_{\mathrm{c}}$) is a boolean variable which takes 1 if and only if the l-th end effector makes contact with the i-th surface during the time interval $[t_k, t_{k+1}]$, where N is the number of phases considered in planning and n_{c} is the number of contact surfaces. Each end-effector is not allowed to make contact with more than one contact surface simultaneously; thus, $\sum_i \sigma_{k,l,i} \leq 1$ must hold. A boolean vector variable $\boldsymbol{\sigma}_k$ is defined as $\boldsymbol{\sigma}_k = [\sigma_{k,1,1}, \ldots, \sigma_{k,n_{\mathrm{e}},n_{\mathrm{s}}}]$, and $\boldsymbol{\sigma}$ (without subscript) is defined as $\sigma = [\boldsymbol{\sigma}_0, \ldots, \boldsymbol{\sigma}_N]$.

The rigid contact model requires that the complementarity condition is satisfied between contact force and end-effector movement; that is, either of them must be zero at any instant of time. In our formulation, the complementarity condition is imposed to the planning problem by assigning proper weights to the following *complementarity cost*.

$$\begin{aligned} L_{\mathrm{compl},k} = \sum_l \Big[&\sum_i \left(\sigma_{k,l,i} w_{\mathrm{compl}}^2 (\boldsymbol{\eta}_i^{\mathsf{T}}(\boldsymbol{p}_{l,k} - \boldsymbol{o}_i))^2\right) \\ &+ \left(\sum_i \sigma_{k,l,i}\right) w_{\mathrm{compl}}^2 \|\Delta \boldsymbol{p}_{l,k}\|^2 \\ &+ \left(1 - \sum_i \sigma_{k,l,i}\right) w_{\mathrm{compl}}^2 \left[\lambda_{l,k}^2 + \|\boldsymbol{\tau}_{l,k}\|^2\right] \Big] \end{aligned} \tag{19}$$

The first term requires that the distance between the l-th end-effector and the i-th contact surface in the normal direction must be zero, if they are in contact ($\sigma_{k,l,i} = 1$). Here, $\boldsymbol{o}_i$ and $\boldsymbol{\eta}_i$ are the origin and the normal of the i-th contact face, respectively. The second term requires that the end-effector movement must be zero if it is in contact with one of the contact surfaces ($\sum_i \sigma_{k,l,i} = 1$). The third term requires that the stiffness and the contact moment applied to the end-effector must be zero if it is not in contact with any of the contact surfaces ($\sum_i \sigma_{k,l,i} = 0$). The value of the weight w_{compl} is gradually increased during the course of optimization, as described in Section IV.

D. Formulation of Optimal Control Problem

The overall cost function is defined as

$$\begin{aligned} J[\boldsymbol{\sigma}] &= \sum_k L_k[\boldsymbol{\sigma}_k], \\ L_k[\boldsymbol{\sigma}_k] &= L_{\mathrm{task},k} + L_{\mathrm{limit},k} + L_{\mathrm{compl},k}[\boldsymbol{\sigma}_k] \end{aligned} \tag{20}$$

and the planning problem is formulated as the following optimal control problem:

$$\begin{aligned} &\text{find} && \boldsymbol{\sigma}, \boldsymbol{x}, \boldsymbol{u} \\ &\text{minimize} && J[\boldsymbol{\sigma}](\boldsymbol{x}, \boldsymbol{u}) \\ &\text{subject to} && \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{aligned} \tag{21}$$

This problem is hard to solve directly for two reasons. First, J and f both have nonlinearity. Second, it has a combinatorial nature because of the presence of the discrete decision variable $\boldsymbol{\sigma}$. To deal with nonlinearity, we take a strategy known as the differential dynamic programming. For optimizing with respect to $\boldsymbol{\sigma}$, we propose a novel best-neighbor search strategy, which will be described in the next subsection.

IV. Differential Dynamic Programming with Best-Neighbor Contact Sequence Optimization

A. Backward and Forward Pass of DDP

Let us once put aside the problem of discrete variable $\boldsymbol{\sigma}$, and review the procedure and characteristics of differential dynamic programming (DDP). In the dynamic programming framework, the value function, or the optimal cost-to-go, is defined as follows:

$$V_k(\boldsymbol{x}) = \min_{\boldsymbol{u}_{k,\ldots,N-1}} \sum_{k'=k}^{N} L_{k'} \ \ \boldsymbol{x}_k = \boldsymbol{x} \tag{22}$$

This function gives the minimum cost of all possible sub-trajectories from k to N that starts from $\boldsymbol{x}$. Here, the well-known Bellman equation states that the following recursive relationship holds.

$$V_N(\boldsymbol{x}) = L_N(\boldsymbol{x}) \tag{23}$$

$$Q_k(\boldsymbol{x}, \boldsymbol{u}) = L_k(\boldsymbol{x}, \boldsymbol{u}) + V_{k+1}(f(\boldsymbol{x}, \boldsymbol{u})) \tag{24}$$

$$V_k(\boldsymbol{x}) = \min_{\boldsymbol{u}} Q_k(\boldsymbol{x}, \boldsymbol{u}) \tag{25}$$

Using (23)-(25), one can calculate the value function recursively backward in time, starting from $k = N$. If there is nonlinearity in the state equation and/or the cost function, however, it is generally difficult to calculate the value function analytically. The DDP (differential dynamic

programming) provides a way to compute the value function by means of quadratic approximation.

$$
\begin{aligned}
&V_k(\boldsymbol{x}+\delta\boldsymbol{x}) \approx V_k(\boldsymbol{x}) + V_{k,x}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T} V_{k,xx}\delta\boldsymbol{x} \\
&Q_k(\boldsymbol{x}+\delta\boldsymbol{x},\boldsymbol{u}+\delta\boldsymbol{u}) \approx Q_k(\boldsymbol{x},\boldsymbol{u}) + Q_{k,x}\delta\boldsymbol{x} + Q_{k,u}\delta\boldsymbol{u} \\
&+ \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T} Q_{k,xx}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{u}^\mathsf{T} Q_{k,uu}\delta\boldsymbol{u} + \delta\boldsymbol{u}^\mathsf{T} Q_{k,ux}\delta\boldsymbol{x}
\end{aligned} \tag{26}
$$

Here, $V_{k,x}, Q_{k,x}, Q_{k,u}$ are the first derivatives (i.e., the gradients) of the value function and the action value function, and $V_{k,xx}, Q_{k,xx}, Q_{k,uu}, Q_{k,ux}$ are the second derivatives. Based on this quadratic approximation, the computation of the Bellman equation can be expressed by the following series of linear operations.

$$
\begin{aligned}
Q &= L + V' & V &= Q - \frac{1}{2}Q_u^\mathsf{T} Q_{uu}^{-1} Q_u \\
Q_x &= L_x + f_{k,x}^\mathsf{T} V_x' & V_x &= Q_x - Q_u^\mathsf{T} Q_{uu}^{-1} Q_{ux} \\
Q_u &= L_u + f_{k,u}^\mathsf{T} V_x' & V_{xx} &= Q_{xx} - Q_{ux}^\mathsf{T} Q_{uu}^{-1} Q_{ux} \\
Q_{xx} &= L_{xx} + f_{k,x}^\mathsf{T} V_{xx}' f_{k,x} & & \\
Q_{uu} &= L_{uu} + f_{k,u}^\mathsf{T} V_{xx}' f_{k,u} & & \\
Q_{ux} &= L_{ux} + f_{k,x}^\mathsf{T} V_{xx}' f_{k,u} & &
\end{aligned} \tag{27}
$$

Here, $(*)_k$ and $(*)_{k+1}$ are denoted by $(*)$ and $(*)'$, respectively, to save space. Note that we ignore the second derivatives of the state transition function (i.e., f_{xx}, f_{uu}, and f_{ux}) for simplicity.

The forward pass of DDP computes the update of $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ in the following steps. In the conventional DDP, $\delta\boldsymbol{x}_0$ is simply set as $\mathbf{0}$, but in our case, it is computed to minimize the value function:

$$
\delta\boldsymbol{x}_0 = -V_{0,xx}^{-1} V_{0,x} \tag{28}
$$

The remaining updates are computed recursively as follows.

$$
\begin{aligned}
\delta\boldsymbol{u}_k &= -Q_{k,uu}^{-1}(Q_{k,u} + Q_{k,ux}\delta\boldsymbol{x}_k) \\
\delta\boldsymbol{x}_{k+1} &= f_{k,x}\delta\boldsymbol{x}_k + f_{k,u}\delta\boldsymbol{u}_k
\end{aligned} \tag{29}
$$

In the conventional DDP framework, the value function is computed in the backward pass as described in (27). In a similar manner, it is also possible to derive a forward computation pass. Let us define the optimal cost-so-far function as follows.

$$
U_k(\boldsymbol{x}) = \min_{\boldsymbol{u}_{0,\dots,k-1}} \sum_{k'=0}^{k-1} L_{k'} \;\; \boldsymbol{x}_k = \boldsymbol{x}
$$

This function gives the minimum cost of all possible sub-trajectories from 0 to k that terminates at $\boldsymbol{x}$. The following recursive relationship can be derived.

$$
\begin{aligned}
U_0(\boldsymbol{x}) &= 0 \\
U_{k+1}(\boldsymbol{x}') &= \min_{\boldsymbol{u}} \{U_k(\hat{f}(\boldsymbol{x}',\boldsymbol{u})) + L_k(\hat{f}(\boldsymbol{x}',\boldsymbol{u}),\boldsymbol{u})\}
\end{aligned}
$$

Here, $\hat{f}$ is the 'inverse' of f, which satisfies $\hat{f}(f(\boldsymbol{x},\boldsymbol{u}),\boldsymbol{u}) = \boldsymbol{x}$ for any $\boldsymbol{x}$ and $\boldsymbol{u}$. The recursive update rule of the quadratic approximation of the optimal cost-so-far can be derived as follows.

$$
\begin{aligned}
P' &= U + L & & \\
P_x' &= \hat{f}_x^\mathsf{T}(U_x + L_x) & U' &= P - \frac{1}{2}P_u^\mathsf{T} P_{uu}^{-1} P_u \\
P_u' &= \hat{f}_u^\mathsf{T}(U_x + L_x) + L_{k,u} & U_x' &= P_x - P_u^\mathsf{T} P_{uu}^{-1} P_{ux} \\
P_{xx}' &= \hat{f}_x^\mathsf{T}(U_{xx} + L_{xx})\hat{f}_x & U_{xx}' &= P_{xx} - P_{ux}^\mathsf{T} P_{uu}^{-1} P_{ux} \\
P_{uu}' &= \hat{f}_u^\mathsf{T}(U_{xx} + L_{xx})\hat{f}_u + L_{uu} & & \\
P_{ux}' &= (\hat{f}_u^\mathsf{T}(U_{xx} + L_{xx}) + L_{ux})\hat{f}_x & &
\end{aligned} \tag{30}
$$

By using this optimal cost-so-far function, one can also derive a backward computation pass of optimal trajectory which is a reverse version of (29), but it is not shown here because it is not used in the proposed algorithm.

B. Efficient One Step Update of Minimum Cost Using Value Functions

Consider that for a certain cost function J, we have computed backward and forward value functions, $\boldsymbol{V} = \{V_0, V_1, \dots, V_N\}$ and $\boldsymbol{U} = \{U_0, U_1, \dots, U_N\}$. Once value functions are obtained, one can easily compute the minimum value of the cost function by minimizing the sum of V_k and U_k for an arbitrary time index k:

$$
\begin{aligned}
J^* &= \min_{\delta\boldsymbol{x}} \Big\{ U_k + V_k + (U_{k,x} + V_{k,x})^\mathsf{T}\delta\boldsymbol{x} \\
&\qquad + \frac{1}{2}\delta\boldsymbol{x}^\mathsf{T}(U_{k,xx} + V_{k,xx})\delta\boldsymbol{x} \Big\} \\
&= U_k + V_k \\
&\qquad - \frac{1}{2}(U_{k,x} + V_{k,x})^\mathsf{T}(U_{k,xx} + V_{k,xx})^{-1}(U_{k,x} + V_{k,x})
\end{aligned} \tag{31}
$$

Note that, from the definition of backward and forward value functions, the value of J^* is the same for any choice of k from 0 to N.

Next, consider that we would like to compute the minimum of a slightly modified cost function where the cost of one time instant k is replaced by $\hat{L}_k$. A naive way would be to recompute the value functions completely for this modified cost function, but it is computationally expensive, especially if we would like to compute the minimum of many different instances of modified cost functions. In the following steps, one can compute the minimum of the modified cost function more efficiently by utilizing the value functions at hand.

1) Using the existing V_{k+1}, compute a single step of backward value update (27) and obtain V_k. Here, use the modified cost $\hat{L}_k$ instead of the original cost L_k.
2) Compute the modified minimum cost by applying (31) to V_k and U_k.

C. Neighborhood of Contact Sequences

A contact sequence is parameterized by a boolean vector variable $\boldsymbol{\sigma}$. Here, we restrict transition of contact states so that at most one contact switching occurs at each time step. This restriction can be expressed as

$$
\|\boldsymbol{\sigma}_{k+1} - \boldsymbol{\sigma}_k\| \leq 1 \tag{32}
$$

Here, the norm counts the number of non-zeros. Under this restriction, the contact state of at most one end-effector may

Algorithm 1 Main loop

$\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\sigma} \leftarrow$ INIT()
$w_{\mathrm{compl}} \leftarrow w_{\mathrm{compl,min}}$
loop until convergence
  $\delta\boldsymbol{x}, \delta\boldsymbol{u}, \boldsymbol{\sigma} \leftarrow$ SWITCHEDDDP($\boldsymbol{\sigma}$)
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta\boldsymbol{x}$
  $\boldsymbol{u} \leftarrow \boldsymbol{u} + \delta\boldsymbol{u}$
  $w_{\mathrm{compl}} \leftarrow \min(\gamma w_{\mathrm{compl}}, w_{\mathrm{compl,max}})$
end loop
return $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\sigma}$

change at a time. Moreover, an end-effector cannot directly switch contact surfaces, but it has to switch to non-contact state before making a new contact. A contact sequence satisfying (32) is said to be a valid contact sequence. There are two reasons for imposing this restriction. First, simultaneous switching of multiple contacts is difficult to realize on a real robot. Second, by restricting transitions one can reduce the set of possible contact sequences, which effectively reduces computation cost. This restriction is not absolutely necessary for the development of the method; it is theoretically possible to allow transitions between arbitrary contact states, or to impose even more strict restriction on contact transitions. Consider for example a simple case of two end-effectors and one contact surface. In this case, $\boldsymbol{\sigma}_1 = [[1,1],[0,1],[0,0]]$ is a valid 3-step sequence, while $\boldsymbol{\sigma}_2 = [[1,1],[0,1],[1,0]]$ is not valid, since the first end-effector makes contact and the second one breaks contact simultaneously.

Next, let us define the notion of adjacency of two contact sequences. A valid contact sequence $\boldsymbol{\sigma}$ and another one $\boldsymbol{\sigma}'$ is said to be adjacent to each other if

$$\begin{aligned} &\boldsymbol{\sigma}_0 = \boldsymbol{\sigma}'_0,\ \boldsymbol{\sigma}_N = \boldsymbol{\sigma}'_N, \\ &|\{k \in 1,2,\ldots,N-1 \,|\, \boldsymbol{\sigma}_k \neq \boldsymbol{\sigma}'_k\}| \leq 1 \end{aligned} \tag{33}$$

holds; that is, if their initial and terminal states are the same and they have different contact states in at most one step in between. For a contact sequence $\boldsymbol{\sigma}$, its neighborhood $\mathcal{N}(\boldsymbol{\sigma})$ is defined as the set of all contact sequences adjacent to it. For example, the neighbors of $\boldsymbol{\sigma} = [[1,1],[0,1],[0,0],[1,0]]$ are $[[1,1],[1,0],[0,0],[1,0]$ and $[[1,1],[0,1],[1,1],[1,0]$.

D. Switched DDP

The essential idea behind the proposed algorithm is to evaluate all contact sequences in the neighborhood of the current best sequence, and repeat this procedure until we arrive at an equilibrium. In this manner, we are no longer guaranteed to find a globally optimal solution, but instead we can find a reasonably good contact sequence in much smaller amount of time.

The pseudo-code of the proposed method is shown in Algorithm 1 and 2. Algorithm 1 is the main loop. Here, the complementarity weight w_{compl} is initialized by its minimum value $w_{\mathrm{compl,min}}$, and magnified by a constant rate γ in every iteration until it reaches $w_{\mathrm{compl,max}}$. We found this mechanism helpful for avoiding convergence to an undesirable contact sequence.

Algorithm 2 SWITCHEDDDP

$J^* \leftarrow \infty$
loop
  $\boldsymbol{V} \leftarrow$ VALUEBACKWARD($\boldsymbol{\sigma}$)
  $\boldsymbol{U} \leftarrow$ VALUEFORWARD($\boldsymbol{\sigma}$)
  $\mathcal{N} \leftarrow$ ENUMNEIGHBORS($\boldsymbol{\sigma}$)
  for each $\{k_i, \hat{\boldsymbol{\sigma}}_i\} \in \mathcal{N}$ **do**
    $J_i \leftarrow$ MINCOST($V_{k_i+1}, U_{k_i}, \hat{\boldsymbol{\sigma}}_i$)
  end for
  $i^* = \mathrm{argmin}_i J_i$
  if $J^* \leq J_{i^*}$ **then**
    break
  end if
  $J^* \leftarrow J_{i^*}$
  $L_{k_{i^*}} \leftarrow \hat{L}_{i^*}$
  $\boldsymbol{\sigma}_{k_{i^*}} \leftarrow \hat{\boldsymbol{\sigma}}_{i^*}$
end loop
$\boldsymbol{V} \leftarrow$ VALUEBACKWARD($\boldsymbol{\sigma}$)
$(\delta\boldsymbol{x}, \delta\boldsymbol{u}) \leftarrow$ STATEFORWARD($\boldsymbol{V}$)

Continuous variables are initialized with reference values, where reference values are obtained by cubic interpolation of the initial and the goal configurations, as described in Section III-A. Contact sequence is initialized with a sequence that minimizes the cost $J[\boldsymbol{\sigma}](\boldsymbol{x}, \boldsymbol{u})$ in (21), where $\boldsymbol{x}$ and $\boldsymbol{u}$ are both fixed to the initial trajectory. Since continuous variables are all fixed, this optimization can be done by means of conventional dynamic programming, which is computationally very cheap.

After initialization, it iteratively updates the trajectory until convergence. Algorithm 2 is the minor loop. Here, VALUEBACKWARD computes the backward value function using (27), while VALUEFORWARD computes the forward value function using (30). Next, ENUMNEIGHBORS enumerates all possible local changes applicable to the current contact state sequence. For each candidate change of contact state, the updated minimum cost is evaluated by MINCOST using the procedure described in Section IV-B. Note that this for-loop can be executed in parallel. The best local change is identified and applied to the current contact sequence. The above procedure is repeated until a local equilibrium is reached. Finally, the backward value function is recalculated using the updated contact sequence, and the optimal differential update for the current trajectory is computed by STATEFORWARD using (29).

E. Continuous-time Trajectory Generation

A continuous-time trajectory of the CoM between time instants t_k and t_{k+1} is obtained as the closed-form solution of (6)(7) with initial conditions given by $\boldsymbol{p}_k$, $\boldsymbol{v}_k$.

End-effector trajectories, on the other hand, are not uniquely identified from the outputs of the planner. Here, we define a continuous trajectory of each end-effector between t_k and t_{k+1} by cubic spline interpolation. To avoid the risk of interfering with contact surfaces while moving, a cycloid-like trajectory that moves away from contact surfaces during

TABLE I

SETTING OF WEIGHT PARAMETERS

	$k = 0, N$	otherwise
w_p	100	0.1
w_v	100	1.0
w_θ, w_ω	100	100
w_λ, w_τ	0.01	
$w_{\mathrm{lim},p}$, $w_{\mathrm{lim},\lambda}$, $w_{\mathrm{lim},\tau}$	100.0	
$w_{\mathrm{compl,min}}$	0.01	
$w_{\mathrm{compl,max}}$	100.0	
γ	2.0	

non-contact phases is added to the spline curve.

V. SIMULATION EXPERIMENTS

A. Planning and Simulation Results

The proposed method was applied to four example scenes: flat, stairs, gap, and gap with rail. Each motion was tested in rigid body dynamical simulation in which motion planning was performed offline, and the robot tracked the planned trajectory by whole-body control. For both planning and simulation, a computer with AMD Ryzen 9 5950X CPU with 16 cores was used. Choreonoid was used for simulation environment. The robot model used for simulation has 30 joints and its total mass is 44[kg]. A model with four end-effectors corresponding to feet and hands are used for planning. The setting of the weight parameters is shown in Table I.

Planned reference trajectories are visualized in Figs. 1(a)-(d). For each figure, the thick solid line depicts the trajectory of the CoM, while thinner solid lines depict the trajectories of the end-effectors. We can observe in Fig. 1(a),(b) that bipedal walking gait is automatically generated without any prior knowledge. In Fig. 1(c), jumping motion consisting of a flight phase is generated. Here, physically consistent vertical movement of the CoM is generated by the planner. The gap with rail scene shown in Fig. 1(d) two handrails are placed in the middle of the gap on both sides (left and right). The planner generated motion in which the robot places two hands on the handrail to support its weight while moving over the gap.

Figures 3(a)-(d) visualize how planned contact sequence changed over the course of optimization. The horizontal axis indicates the switching count of contact sequence, while each column indicates the contact sequence at the corresponding iteration cycle. For example, horizontal stripes seen in Fig. 3(a) indicate that the left and the right foot make contact in turn with double support phase in between. In every example, the contact sequence largely settles to a fixed sequence after around 50 switches, but persistent chattering of contact states is observed. This chattering may be occurring between two adjacent contact sequences having almost the same cost. We consider therefore that chattering can be suppressed by introducing certain hysteresis mechnism to contact switching.

Figures 2(a)-(d) show snapshot images of dynamical simulation. We found it quite challenging to track reference trajectories multiple contacts and flight phases. So this time we conducted simulation in an assisted setup in which very large values are assigned to the base link inertia. In this manner, the base link orientation is effectively fixed, and the controller can concentrate on trajectory tracking of the translational movement of the CoM and the end-effectors. Applying more sophisticated whole-body control techniques for achieving better trajectory tracking performance is a subject of future study.

TABLE II

AVERAGE COMPUTATION TIME PER ITERATION, WITH AND WITHOUT PARALLELIZATION.

	w/ parallel [ms]	w/o parallel [ms]
flat	12.16	11.36
stairs	16.73	13.55
gap	12.72	10.87
gap with rail	14.29	12.16

B. Computational Characteristics

Computation time for one iteration (average of 100 iterations) is summarized in Table II. It can be seen that computation time is very small compared to reports in existing studies. Considering that we need about 50 iterations for convergence, the overall computation time for trajectory generation is less than 1 second. Table II compares computation time with and without parallelization. It was against our expectation that it took slightly greater computation time with parallelization. The reason for this result is that the algorithm is fine-grained, so the overhead of thread synchronization was greater than the time gained by parallel execution. We must conclude that OpenMP-based parallelization is not very effective, but it is still possible that tailor-make parallel implementation could achieve further speed up.

VI. CONCLUSION

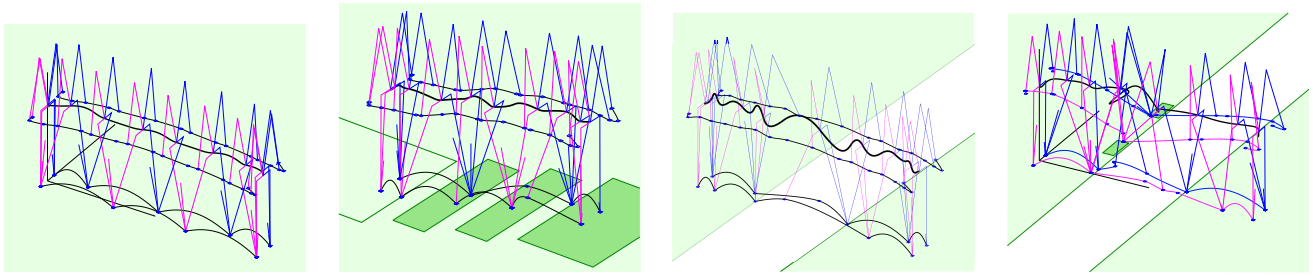
This paper presented a multi-contact motion planning method that is based on discrete local search of contact sequences. The proposed method is able to generate long trajectories by utilizing the closed-form solution of the mc-LIPM. A novel manipulation of forward and backward value functions enabled efficient update of optimal cost, which lead to parallel evaluation of multiple contact sequences. One interesting of future study is to extend the method to motions involving large rotation of base link.

REFERENCES

[1] K. Hauser, T. Bretl, and J. C. Latombe, "Non-gaited humanoid locomotion planning," in *IEEE-RAS Int. Conf. Humanoid Robots*, 2005, pp. 7–12.

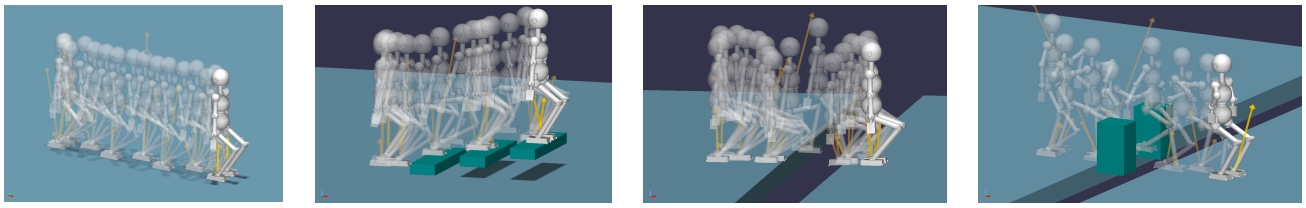
[2] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, "Potential field guide for humanoid multicontacts acyclic motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1165–1170.

[3] K. Bouyarmane and A. Kheddar, "Static multi-contact inverse problem for multiple humanoid robots and manipulated objects," in *IEEE-RAS Int. Conf. Humanoid Robots*, 2010, pp. 8–13.



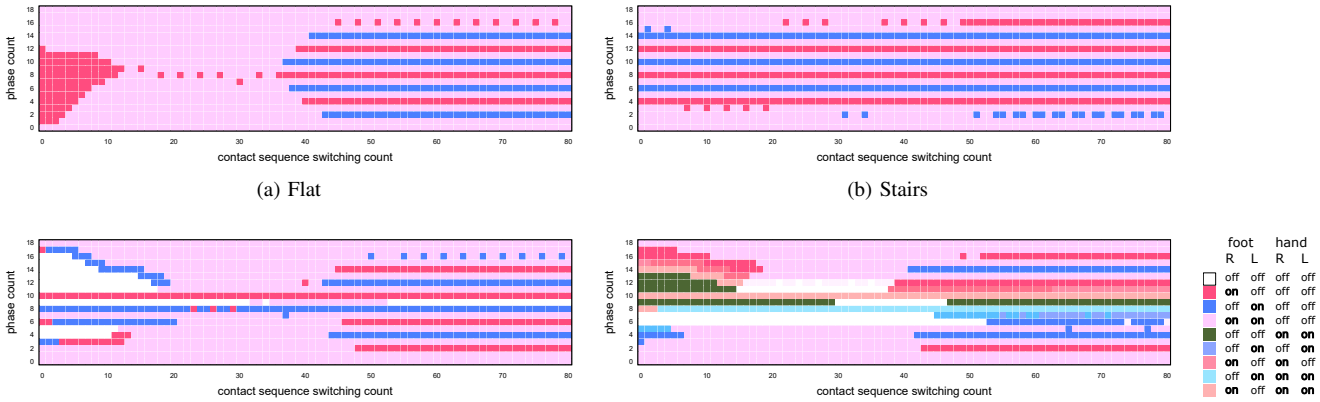
(a) Flat (b) Stairs (c) Gap (d) Gap with rail

Fig. 1. Visualization of generated trajectories. Initial trajectory (top) and after optimization (bottow).



(a) Flat (b) Stairs (c) Gap (d) Gap with rail

Fig. 2. Snapshot images of trajectory tracking in simulation.



(a) Flat (b) Stairs

(c) Gap (d) Gap with rail

Fig. 3. Change of contact sequence during iteration

[4] ——, "Multi-contact stances planning for multiple agents," in *IEEE Int. Conf. Robotics and Automation*, 2011, pp. 5246–5253.

[5] A. Escande and A. Kheddar, "Contact planning for acyclic motion with tasks constraints," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2009, pp. 435–440.

[6] M. Murooka, K. Chappellet, A. Tanguy, M. Benallegue, I. Kumagai, M. Morisawa, F. Kanehiro, and A. Kheddar, "Humanoid loco-manipulations pattern generation and stabilization control," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5597–5604, 2021.

[7] G. C. Thomas and L. Sentis, "Towards computationally efficient planning of dynamic multi-contact locomotion," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3879–3886.

[8] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Trans. Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[9] I. Kumagai, M. Morisawa, S. Hattori, M. Benallegue, and F. Kanehiro, "Multi-contact locomotion planning for humanoid robot based on sustainable contact graph with local contact modification," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6379–6387, 2020.

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

[11] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences," in *IEEE Int. Conf. Robotics and Automation*, 2016, pp. 4636–4641.

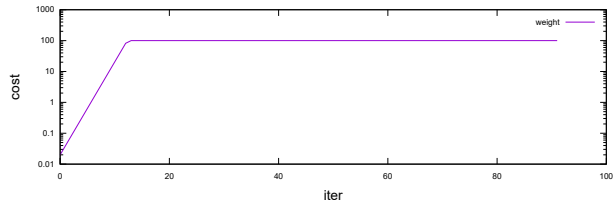
[12] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2018.

[13] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1661–1679, 2021.

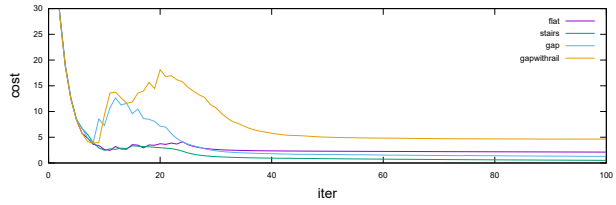
[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012, pp. 4906–4913.

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

[16] 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.



(a) Change of complementarity weight



(b) Change of cost

Fig. 4. Change of complementarity weight and cost during iteration.

[17] I. Chatzinikolaidis, Y. You, and Z. Li, "Contact-implicit trajectory optimization using an analytically solvable contact model for locomotion on variable ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6357–6364, 2020.

[18] Y. Zhu, Z. Pan, and K. Hauser, "Contact-implicit trajectory optimization with learned deformable contacts using bilevel optimization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9921–9927.

[19] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *IEEE International Conference on Robotics and Automation*, 2022.

[20] K. Murota, *Recent Developments in Discrete Convex Analysis*. Springer Berlin Heidelberg, 2009, pp. 219–260.