Centralized Model Predictive Control for Collaborative Loco-Manipulation

Flavio De Vincenti and Stelian Coros

***Abstract*—In this work, we extend the model predictive control methods developed in the legged robotics literature to collaborative loco-manipulation settings. The systems we study entail a payload collectively carried by multiple quadruped robots equipped with a mechanical arm. We use a direct multiple shooting method to solve the resulting high-dimensional, optimal control problems for trajectories of ground reaction forces, manipulation wrenches, and stepping locations. To capture the dominant dynamics of the system, we model each agent and the shared payload as single rigid bodies. We demonstrate the versatility of our framework in a series of simulation experiments involving collaborative manipulation over challenging terrains.**

I. Introduction

The automation of heavy labor through collaborative, all-terrain mobile manipulators would have a tremendous impact on ameliorating human work conditions worldwide. Hardware capable of such a feat has existed for a long time in the form of humanoid and arm-endowed quadruped robots [1]. The coordination and control of these inherently high-dimensional systems—let alone teams thereof—demand great computational resources and efficient algorithms, the complexity of which calls for exceptional software architecture and dynamics modeling efforts. By instilling cooperative loco-manipulation skills in multi-robot teams, the safety and productivity of workers would improve in numerous labor-intensive industries, including construction, mining, search and rescue, etc.

Our goal is to design a computational framework for the simultaneous planning and execution of object maneuvering tasks at the hands of multiple, one-armed quadruped robots, as shown in Figure 1. Versatility and ease of adaption to different environments are of great practical interest to any business benefiting from autonomous walking manipulators. We specifically value the capability of a collaborative loco-manipulation (CLM) implementation to manage seamlessly heterogeneous agents and cope with uneven terrains.

Due to their optimization-based reasoning about state and input constraints, model predictive control (MPC) methods appear to be an adequate tool for CLM. However, implementing a trajectory optimization (TO) algorithm for coordinating potentially arbitrary numbers of legged machines comes with significant challenges. Indeed, the collaborative multi-robot setup naturally translates into very high-dimensional optimal control problems (OCPs). The states and inputs associated with each robot directly affect the motion of the manipulated payload, making the whole system tightly coupled and highly nonlinear. Furthermore, real-time rates are necessary to guarantee robust MPC performance, thus requiring careful software design and, most importantly, a meditated trade-off between model accuracy and expressive power.

We address the above problems by proposing an efficient, centralized, multiple-shooting MPC scheme. Our approach optimizes the interaction wrenches and motion trajectories of all robots and the payload over a given time horizon within a single, sparse TO problem. We achieve efficient solution times by modeling every agent, including the shared object, as single rigid bodies; this device allows us to significantly reduce

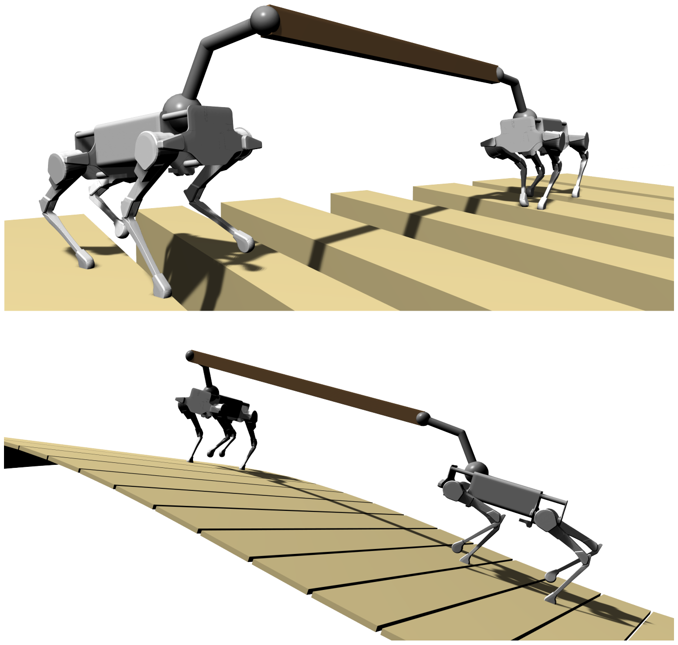

Fig. 1. Two simulated *Laikago* robots [2] manipulating a wooden beam on rough terrains. Since all robots influence the object's motion and, indirectly, each other's, a planning algorithm must be able to reason holistically about the system. Our centralized MPC naturally fulfills this requirement and, by incorporating stepping locations as optimization variables, enables locomotion on ditches (**top**) and slopes (**bottom**). Recordings of all the experiments can be found in the accompanying video.

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR DFAB) and through Grant No. 200021 200644, and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant No. 866480).

We thank Roi Poranne, Dongho Kang, Yijiang Huang, Oliver Stark, and Camilla Casamento Tumeo for their invaluable support and assistance in creating the figures and the accompanying video.

The authors are with the Computational Robotics Lab, ETH Zurich, Switzerland. flavio.devincenti@inf.ethz.ch

the size of the resulting OCPs as opposed to methods using the articulated rigid-body dynamics [12, 30, 31]. Our model choice further grants faster derivative computations compared to centroidal dynamics-based implementations [20, 34], and we find its level of abstraction well justified in the scope of CLM. Also, to handle any possible payload rotation without incurring singularities, we parameterize the orientations via unit quaternions and adopt a Lie group time-stepping scheme to integrate angular velocities. This strategy embeds the unit-norm constraints implicitly into the discretized system dynamics and makes variation-based linearizations an unnecessary complication [9, 15]. Moreover, by including the stepping locations of each robot in the optimization variables, we make the system more robust against external perturbations, as well as internal disturbances induced by other agents. Finally, we conduct simulation experiments featuring multiple robots transporting a payload on rough and non-flat terrains to show the potential and adaptability of our formulation.

II. Related Work

Most existing works on the collaborative control of multi-agent systems are focused on aerial [8, 29, 32, 36, 38, 39] and ground [11, 17, 28, 33] vehicles with shared payloads. Despite a large number of publications about multi-robot systems (MRSs), which attests to their great significance in the robotics community, only a few focus on cooperative legged machines, and even fewer explore CLM. This fact can be attributed to the extraordinary computational challenge of controlling these interconnected high-dimensional systems and, on a practical level, the lack of suitable, commercially available hardware.

The closest works in the direction of collaborative loco-manipulation study one-armed quadruped robots in isolation or the locomotion of rigidly connected four-legged machines. Sleiman et al. [34] adopt a differential dynamic programming (DDP) method to optimize the trajectories of a four-legged manipulator interacting with the environment. They base their implementation on the centroidal dynamics model together with the dynamics of the manipulated object, and complement it with self-collision avoidance constraints in a later publication [10]. In this work, we employ a similar state augmentation strategy, but we elect to represent all agents and the payload using the single rigid body dynamics (SRBD) model to keep a centralized CLM algorithm computationally tractable.

Kim and Hamed [24] tackle the cooperative locomotion problem by decoupling it into a centralized and a distributed part. The former describes the dominant dynamics of each robot using a modified linear inverted pendulum model (LIPM), and then it plans trajectories for the whole system by solving a single optimization problem in an MPC scheme; the optimized trajectories are eventually tracked by the distributed whole-body controllers running on the robots. The authors also show a loco-manipulation example where multiple robots carry a lightweight payload together. Although pioneering, their CLM demonstration is limited by the simplicity of LIPM since it does not allow reasoning about torques and angular momenta.

Fawcett et al. [18] and Kim et al. [25] improve on the centralized/distributed idea by adopting the SRBD model instead in a data-driven and purely model-based fashion, respectively. Fawcett et al. synthesize template models for enabling efficient planning of squads of holonomically constrained quadruped robots, while Kim et al. describe both a centralized and a decentralized MPC architecture for collaborative locomotion, and show real-time performance for two interconnected quadruped robots over a $25\,\mathrm{ms}$ time horizon. Despite a valuable insight into the performance of learning-based and distributed control schemes compared to centralized ones, their work remains in the scope of multi-agent locomotion, and it does not involve any cooperative manipulation experiments. Moreover, the short time horizon makes their MPC demonstrations similar to those of a reactive controller. Finally, neither of the above works optimizes for the stepping locations, and all rely on prefixed gaits. In this work, we overcome these limitations and formulate the CLM problem as robot-, payload-, and gait-agnostic. By additionally adopting a custom, sparsity-exploiting sequential quadratic programming (SQP) solver, we manage to coordinate up to two robots and a manipulated object at real-time rates through a single TO.

III. Collaborative Loco-Manipulation

We formulate CLM as the following nonlinear, discrete-time trajectory optimization problem:

$$\min_{\mathbf{U},\mathbf{X}} \; \mathcal{L}(\mathbf{U},\mathbf{X}) := \sum_{k=0}^{N-1} l_k(\mathbf{x}_k,\mathbf{u}_k) + l_N(\mathbf{x}_N) \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}_m\,, \tag{1b}$$

$$\mathbf{x}_{k+1} = \mathbf{d}_k(\mathbf{x}_k,\mathbf{u}_k)\,, \tag{1c}$$

$$\mathbf{g}(\mathbf{X},\mathbf{U}) = \mathbf{0}\,, \tag{1d}$$

$$\mathbf{h}(\mathbf{X},\mathbf{U}) \leq \mathbf{0}\,, \tag{1e}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ are the states and inputs of the system at time step k, and $\mathbf{x}_m$ is the current measured state; we define the stacked state and input vectors as:

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_0^\top & \mathbf{x}_1^\top & \dots & \mathbf{x}_N^\top \end{bmatrix}^\top \text{ and}$$
$$\mathbf{U} := \begin{bmatrix} \mathbf{u}_0^\top & \mathbf{u}_1^\top & \dots & \mathbf{u}_{N-1}^\top \end{bmatrix}^\top .$$

The scalar functions $l_k(\cdot)$ and $l_N(\cdot)$ denote the running and final costs, while $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ are general state-input equality and inequality constraints, respectively. Finally, $\mathbf{d}_k(\cdot)$ is the time-varying discrete dynamics function describing the evolution of the system at time step k.

By solving (1) over a receding horizon in an MPC fashion, we seek to realize robust tracking performance in the face of disturbances and uncertainty. Our modeling choices are geared toward making a centralized implementation of CLM as shown in Figure 2 as tractable as possible, while enabling highly dynamic optimized motions. In the following sections, we describe in detail the various components of (1).

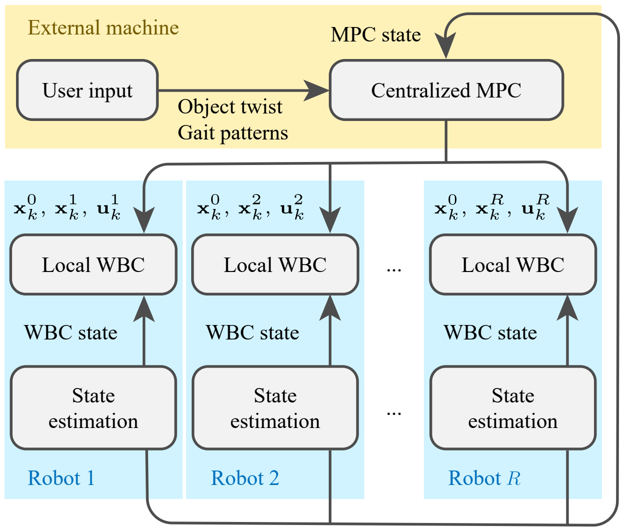

Fig. 2. Schematic overview of the proposed method. Our solution entails a centralized planner interfacing with all agents engaged in CLM. The user commands specify both the desired linear and angular velocities of the manipulated object, as well as the gait pattern for each robot. Based on this information and the measured states of all agents, our centralized MPC formulates a single optimization problem to compute the discrete-time trajectories of payload states $\mathbf{x}_k^0$, and robots' states $\mathbf{x}_k^i$ and inputs $\mathbf{u}_k^i$ for $i \in \{1, 2, \ldots, R\}$. The resulting high-level plans are then individually tracked by all agents using onboard whole-body control (WBC) implementations [3, 10, 34]. To ensure that the manipulated object closely tracks the output CLM plans, the end effector poses and manipulation wrenches must be treated as hard constraints by the robots' local controllers. This paper focuses on the centralized MPC strategy at the core of our approach.

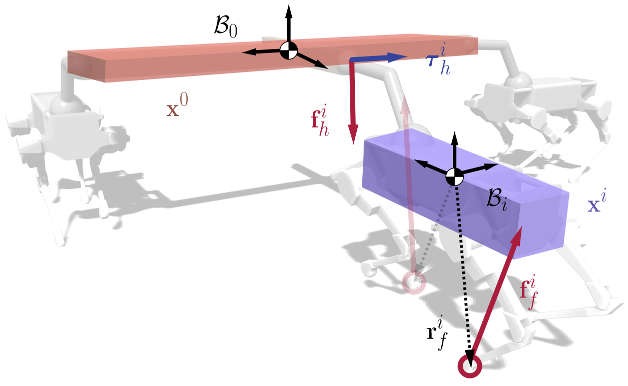

Fig. 3. Graphical description of the CLM system. All robots and the manipulated object are modeled as single rigid bodies with local frames $\mathcal{B}_i$ and states $\mathbf{x}^i$. Each robot interacts with the environment through ground reaction forces $\mathbf{f}_f^i$ and with the payload through a manipulation force $\mathbf{f}_h^i$ and torque $\boldsymbol{\tau}_h^i$. We additionally model as input the position $\mathbf{r}_f^i$ of each foot relative to its robot's center of mass.

A. System Dynamics

Given the high dimensionality inherent to CLM, adopting the articulated rigid-body dynamics of the whole interconnected system for a centralized implementation is out of the question. We conjecture that the widespread single rigid body dynamics [7] provide a good abstraction for representing the various one-armed agents in CLM scenarios. For this purpose, we hypothesize the following:

Assumption 1. *The legs of the robots are lightweight compared to their bodies and we can neglect their inertial effects.*

Assumption 2. *The movements of the arms relative to the robot torsos are restrained and do not significantly deviate from a default pose, so torso and arm act like a single body.*

While the former assumption is prevalent in most state-of-the-art MPC-based locomotion controllers [6, 15, 16, 19], we find the latter peculiar to CLM. If we picture two people collaboratively carrying a heavy object, we can expect that their torsos and hands will keep a constant relative pose most of the time. Indeed, when lifting hefty loads, humans bring their hands closer to their bodies to reduce the momenta weighing on their backs, thus restricting the relative movements between arms and torso [23]. Also, the satisfaction of Assumption 2 corroborates Assumption 1 in that the combined weight of the base and arm would make the legs comparably more lightweight. These observations motivate our decision to model the robots in CLM as single rigid bodies. Nevertheless, if any of the above hypotheses gets sometimes violated, we expect that the MPC feedback loop will reject most model mismatches as disturbances.

Continuous Dynamics: We view CLM as an ensemble of interacting floating rigid bodies: one for the payload and one for each robot. In this unified representation, the robots exert forces on the environment through their point-contact feet and wrenches on the passive, manipulated object through their surface-contact hands. Figure 3 contains a schematic illustration of the system.

We assume that each robotic hand is rigidly attached to the object, and its pose with respect to the payload local frame $\mathcal{B}_0$ is fixed; therefore, knowing the poses of all rigid bodies implies knowing the configuration of each mechanical arm through inverse kinematics (IK). Let R be the total number of robots and the index $i \in \{0, 1, \ldots, R\}$ identify the payload if $i = 0$ or the ith robot if $i > 0$. Then, we define the state of each rigid body at time $t \in \mathbb{R}$ as:

$$\mathbf{x}^i := \begin{bmatrix} \mathbf{p}^{i\top} & \mathbf{q}^{i\top} & \dot{\mathbf{p}}^{i\top} & \boldsymbol{\Omega}^{i\top} \end{bmatrix}^\top ,$$

where $\mathbf{p}^i \in \mathbb{R}^3$, $\mathbf{q}^i \in \mathbb{S}^3 \subseteq \mathbb{R}^4$, $\dot{\mathbf{p}}^i \in \mathbb{R}^3$ and $\boldsymbol{\Omega}^i \in \mathbb{R}^3$ represent the position, orientation, linear velocity, and angular velocity of the ith rigid body, respectively. All quantities are defined in an inertial frame $\mathcal{I}$ except the angular velocity $\boldsymbol{\Omega}^i$, which is expressed in the ith body frame $\mathcal{B}_i$.

We associate each foot $f \in \{\text{LF}, \text{RF}, \text{LH}, \text{RH}\}$[1] and each hand h of the robots with a corresponding input, namely:

$$\mathbf{u}_f^i = \begin{bmatrix} \mathbf{f}_f^{i\top} & \mathbf{r}_f^{i\top} \end{bmatrix}^\top ,$$
$$\mathbf{u}_h^i = \begin{bmatrix} \mathbf{f}_h^{i\top} & \boldsymbol{\tau}_h^{i\top} \end{bmatrix}^\top ,$$

where $\mathbf{f}_f^i \in \mathbb{R}^3$ is the ground reaction force, $\mathbf{r}_f^i \in \mathbb{R}^3$ is the foot position with respect to the robot's center of mass expressed in

[1]Throughout this manuscript, LF denotes the left-front limb, RF the right-front, LH the left-hind, and RH the right-hind.

the inertial frame, and $\mathbf{f}_h^i, \boldsymbol{\tau}_h^i \in \mathbb{R}^3$ are the manipulation force and torque, respectively. Since the payload can only move due to the robots' actions, $\mathbf{u}_f^i$ and $\mathbf{u}_h^i$ are defined only for $i > 0$. For convenience, we define the following stacked input vector:

$$\mathbf{u}^i := \begin{bmatrix} \mathbf{u}_{\text{LF}}^{i\top} & \mathbf{u}_{\text{RF}}^{i\top} & \mathbf{u}_{\text{LH}}^{i\top} & \mathbf{u}_{\text{RH}}^{i\top} & \mathbf{u}_h^{i\top} \end{bmatrix}^\top .$$

We use a binary variable $\mathbf{s}_f^i \in \{0, 1\}$ to enforce different gait patterns; similarly to [4], $\mathbf{s}_f^i$ is set to 1 if and only if the fth foot of the ith robot is in contact with the ground at time t. Then, we can describe the dynamics evolution of the ith robot as:

$$\ddot{\mathbf{p}}^i := \frac{1}{m^i}\left(\mathbf{f}_h^i + \sum_f \mathbf{s}_f^i \mathbf{f}_f^i\right) + \mathbf{g}\,, \tag{2a}$$

$$\dot{\boldsymbol{\Omega}}^i := \left(\mathbf{I}^i\right)^{-1}\Bigg[-\boldsymbol{\Omega}^i \times \mathbf{I}^i\boldsymbol{\Omega}^i + \left(\mathbf{q}^i\right)^{-1} * \left(\boldsymbol{\tau}_h^i + \mathbf{r}_h^i(\mathbf{x}^0, \mathbf{x}^i) \times \mathbf{f}_h^i + \sum_f \mathbf{s}_f^i \mathbf{r}_f^i \times \mathbf{f}_f^i\right)\Bigg]\,, \tag{2b}$$

where $m^i \in \mathbb{R}$ is the robot's lumped mass, $\mathbf{I}^i \in \mathbb{R}^{3\times3}$ is its lumped moment of inertia in the body frame $\mathcal{B}_i$, $\mathbf{r}_h^i(\cdot)$ is the position of the hand with respect to the robot's center of mass in the inertial frame $\mathcal{I}$, $\mathbf{g} \in \mathbb{R}^3$ is the gravitational acceleration vector, and $*$ denotes the quaternion multiplication operator. We remark that, because the hands are fixed to the manipulated object, $\mathbf{r}_h^i(\cdot)$ is a function of the object state $\mathbf{x}^0$ and the ith robot state $\mathbf{x}^i$. Analogously, we define the payload dynamics:

$$\ddot{\mathbf{p}}^0 := -\frac{1}{m^0}\sum_i \mathbf{f}_h^i + \mathbf{g}\,, \tag{3a}$$

$$\dot{\boldsymbol{\Omega}}^0 := \left(\mathbf{I}^0\right)^{-1}\Bigg[-\boldsymbol{\Omega}^0 \times \mathbf{I}^0\boldsymbol{\Omega}^0 - \left(\mathbf{q}^0\right)^{-1} * \left(\sum_i \boldsymbol{\tau}_h^i + (\mathbf{p}^i + \mathbf{r}_h^i(\mathbf{x}^0, \mathbf{x}^i) - \mathbf{p}^0) \times \mathbf{f}_h^i\right)\Bigg]. \tag{3b}$$

The object dynamics equations (3) are solely affected by the wrenches applied by all robots, thus making collaboration crucial for completing any desired manipulation task.

Discretization: To embed the continuous dynamics (2) and (3) into the finite dimensional nonlinear programming (NLP) problem (1), we must first convert them into the discrete-time form (1c). Given a time horizon $T \in \mathbb{R}_{>0}$, we sample states and inputs at $N \in \mathbb{N}$ grid points over the prediction interval $[t, t+T)$. We employ a zero-hold parameterization of the control inputs $\mathbf{u}^i$ to integrate the system dynamics equations over each shooting interval of duration $\Delta t \in \mathbb{R}$. We determine the step size as $\Delta t := T/(N+1)$, and we denote a quantity sampled at time $t_k = t + k\Delta t, \forall k \in \mathbb{N}$ with the subscript k; for instance, $\mathbf{x}_k^i$, $\mathbf{u}_k^i$, etc. Then, we discretize the continuous dynamics (2) and (3) using the following numerically stable, Lie group semi-implicit Euler method:

$$\mathbf{p}_{k+1}^i = \mathbf{p}_k^i + \int_{t_k}^{t_{k+1}} \dot{\mathbf{p}}^i \mathrm{d}t \approx \mathbf{p}_k^i + \dot{\mathbf{p}}_{k+1}^i \Delta t\,, \tag{4a}$$

$$\mathbf{q}_{k+1}^i = \mathbf{q}_k^i + \int_{t_k}^{t_{k+1}} \dot{\mathbf{q}}^i \mathrm{d}t \approx \mathbf{q}_k^i * \exp(\boldsymbol{\Omega}_{k+1}^i \Delta t)\,, \tag{4b}$$

$$\dot{\mathbf{p}}_{k+1}^i = \dot{\mathbf{p}}_k^i + \int_{t_k}^{t_{k+1}} \ddot{\mathbf{p}}^i \mathrm{d}t \approx \dot{\mathbf{p}}_k^i + \ddot{\mathbf{p}}_k^i \Delta t\,, \tag{4c}$$

$$\boldsymbol{\Omega}_{k+1}^i = \boldsymbol{\Omega}_k^i + \int_{t_k}^{t_{k+1}} \dot{\boldsymbol{\Omega}}^i \mathrm{d}t \approx \boldsymbol{\Omega}_k^i + \dot{\boldsymbol{\Omega}}_k^i \Delta t\,, \tag{4d}$$

where $\exp\colon \mathbb{R}^3 \to \mathbb{S}^3$ is a Lie-group exponential function which, for unit quaternions, takes the following closed form:

$$\exp(\mathbf{v}) := \begin{cases} \cos(\frac{1}{2}\|\mathbf{v}\|) + \frac{\mathbf{v}}{\|\mathbf{v}\|}\sin(\frac{1}{2}\|\mathbf{v}\|) & \|\mathbf{v}\| \neq 0 \\ 1 & \|\mathbf{v}\| = 0 \end{cases}.$$

We remark that (4a) and (4b) adopt the quantities $\dot{\mathbf{p}}_{k+1}^i$ and $\boldsymbol{\Omega}_{k+1}^i$ computed for time step $k+1$; these are obtained from the expressions (4c) and (4d), respectively. Equation (4b) integrates the angular velocity $\boldsymbol{\Omega}_k^i$ from the starting orientation $\mathbf{q}_k^i$ without relinquishing the unit quaternion space. Thus, if $\mathbf{q}_k^i$ satisfies the unit norm constraint and represents a valid rotation, $\mathbf{q}_{k+1}^i$ will do as well.

Finally, by grouping the equations (4) for every rigid body i together, we can express the discrete dynamics equation of the system in the form (1c):

$$\mathbf{x}_{k+1} = \mathbf{d}_k(\mathbf{x}_k, \mathbf{u}_k)\,, \tag{5}$$

where

$$\mathbf{x}_k := \begin{bmatrix} \mathbf{x}_k^0 & \mathbf{x}_k^1 & \dots & \mathbf{x}_k^R \end{bmatrix}^\top \text{ and}$$
$$\mathbf{u}_k := \begin{bmatrix} \mathbf{u}_k^1 & \mathbf{u}_k^2 & \dots & \mathbf{u}_k^R \end{bmatrix}^\top$$

are, respectively, the state and input vectors of the entire system as defined in (1). We highlight that the dependency on k of $\mathbf{d}_k$ encapsulates the contributions of the gait-dependent phase variables $s_{k,f}^i$.

B. General Constraints

We codify the general constraints (1d) and (1e) by extending the formulation proposed by Bledt and Kim [4] for four-legged systems to one-armed quadruped robots.

Equality Constraints: Every time a foot ends a swing phase and lands, it must come in contact with the ground. To this end, we represent the terrain as a 2.5D map $z_g\colon \mathbb{R}^2 \to \mathbb{R}$ mapping x- and y-coordinates to the ground height; given the position $\mathbf{p}_{k,f}^i = \mathbf{p}_k^i + \mathbf{r}_{k,f}^i$ of the fth foot of the ith robot at time step k, we can write:

$$\left(s_{k-1,f}^i - 1\right) s_{k,f}^i \left[\mathbf{p}_{k,f,z}^i - z_g(\mathbf{p}_{k,f,x}^i, \mathbf{p}_{k,f,y}^i)\right] = 0\,, \tag{6}$$

where the subscripts x, y and z denote the corresponding coordinates in $\mathbb{R}^3$.

We enforce a contact consistency constraint to ensure that a foot does not move during stance phases:

$$s_{k-1,f}^i\, s_{k,f}^i \left(\mathbf{p}_{k-1,f}^i - \mathbf{p}_{k,f}^i\right) = \mathbf{0}\,. \tag{7}$$

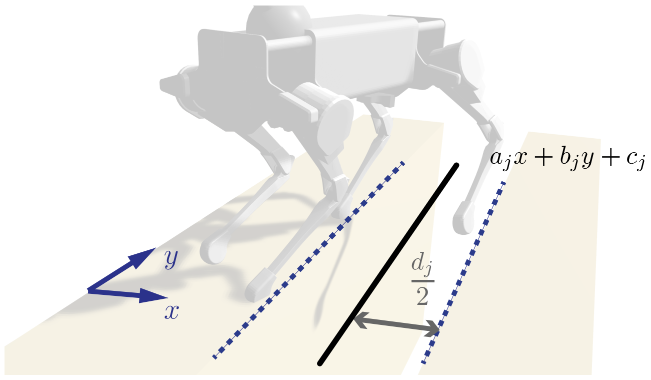

Fig. 4. Geometric representation of a ditch. We model ditches as lines $a_jx + b_jy + c$ on the xy-plane coupled with a width measure d_j.

We assign $\mathbf{p}^i_{-1,f}$ to the current foot position in the real world, so that the enforcement of (7) yields an input $\mathbf{u}^i_{0,f}$ consistent with the measured state of the system. Eventually, we assemble (6) and (7) for all k, $i > 0$, and f to form the general equality constraints (1d) for our TO problem.

Inequality Constraints: Given the ith robot's leg and arm lengths l^i_f and l^i_h, respectively, we enforce kinematic limits for all limbs through the following inequality constraints:

$$\|\mathbf{r}^i_{k,f} - \mathbf{r}^i_{j,f}(\mathbf{x}^i_k)\| - l^i_f \leq 0\,, \tag{8a}$$
$$\|\mathbf{r}^i_{k,h} - \mathbf{r}^i_{j,h}(\mathbf{x}^i_k)\| - l^i_h \leq 0\,, \tag{8b}$$

where $\mathbf{r}^i_{j,f}(\cdot)$ and $\mathbf{r}^i_{j,h}(\cdot)$ are the joint locations with respect to $\mathbf{p}^i_k$ from which the leg and arm stem; they depend only on the robot's state $\mathbf{x}^i_k$.

To prevent a robot from pulling the ground with its feet, we write

$$-s^i_{k,f}\left[\mathbf{f}^i_{k,f} \cdot \mathbf{n}_g(\mathbf{p}^i_{k,f,x}, \mathbf{p}^i_{k,f,y})\right] \leq 0\,, \tag{9}$$

where $\mathbf{n}_g(\cdot)$ is the unit vector normal to the ground where the foot is located.

We enforce linearized friction cone constraints for each foot through the following inequalities:

$$|\mathbf{f}^i_{k,f} \cdot \mathbf{t}^{xz}_g(\mathbf{p}^i_{k,f,x}, \mathbf{p}^i_{k,f,y})| - \mu\mathbf{f}^i_{k,f} \cdot \mathbf{n}_g(\mathbf{p}^i_{k,f,x}, \mathbf{p}^i_{k,f,y}) \leq 0\,, \tag{10a}$$
$$|\mathbf{f}^i_{k,f} \cdot \mathbf{t}^{yz}_g(\mathbf{p}^i_{k,f,x}, \mathbf{p}^i_{k,f,y})| - \mu\mathbf{f}^i_{k,f} \cdot \mathbf{n}_g(\mathbf{p}^i_{k,f,x}, \mathbf{p}^i_{k,f,y}) \leq 0\,, \tag{10b}$$

where $\mu \in \mathbb{R}$ is the static friction coefficient, and $\mathbf{t}^{xz}_g(\cdot)$ and $\mathbf{t}^{yz}_g(\cdot)$ are unit versors tangent to the ground and lying in the xz- and yz-planes, respectively.

Finally, in order to cope with unstructured terrains, we model ditches running along lines given by the equation $a_jx + b_jy + c_j = 0$ as shown in Figure 4, where $j \in \mathbb{N}$ identifies the jth ditch. We formulate the corresponding ditch-crossing constraint as:

$$\frac{d_j}{2} - \frac{\left|a_j\mathbf{p}^i_{k,f,x} + b_j\mathbf{p}^i_{k,f,y} + c_j\right|}{\sqrt{a_j^2 + b_j^2}} \leq 0\,, \tag{11}$$

where d_j is the width of the ditch. We note that we can apply and extend (11) to enable CLM over more general terrain classes. For instance, we can represent passable convex areas as intersections of enough half-spaces, each expressed through an affine constraint $\mathbf{a}^\top\mathbf{p}^i_{k,f} + \mathbf{b} \leq 0$ for some $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$. Such a strategy would allow us to model stepping stones, aerial walkways, and staircases [20] and showcases the benefits of treating stepping locations as optimization variables.

Once again, we stack the inequality constraints (8), (9), (10), (11) for all k, $i > 0$, and f in (1e), and we feed them to our optimal control problem (1).

C. Cost Function

We define the running state cost of the ith rigid body as:

$$l^i_{k,x}(\mathbf{x}^i_k, \mathbf{u}^i_k) := \|\mathbf{p}^i_k - \mathbf{p}^{i*}_k\|^2_{\mathbf{W}_p} + \tag{12a}$$
$$w_q \min\left\{\|\mathbf{q}^i_k - \mathbf{q}^{i*}_k\|^2, \|\mathbf{q}^i_k + \mathbf{q}^{i*}_k\|^2\right\} + \tag{12b}$$
$$\|\dot{\mathbf{p}}^i_k - \dot{\mathbf{p}}^{i*}_k\|^2_{\mathbf{W}_{\dot{p}}} + \tag{12c}$$
$$\|\mathbf{\Omega}^i_k - \mathbf{\Omega}^{i*}_k\|^2_{\mathbf{W}_\Omega}\,, \tag{12d}$$

where the superscript $*$ denotes reference states, $w_q \in \mathbb{R}_{>0}$, $\mathbf{W}_p, \mathbf{W}_{\dot{p}}, \mathbf{W}_\Omega \in \mathbb{R}^{3\times3}$ are diagonal, positive definite matrices, and $\|\cdot\|_\mathbf{W}$ computes the weighted norm of a vector with weight matrix $\mathbf{W}$. The orientation cost term (12b) gives values in the range $[0, w_q\sqrt{2}]$ and represents a metric on 3D rotations—i.e., it is equal to 0 if and only if $\mathbf{q}^i_k$ and $\mathbf{q}^{i*}_k$ represent the same orientation in $SO(3)$ [21].

For each robot, we also regularize ground reaction forces, manipulation wrenches, and stepping locations:

$$l^i_{k,u}(\mathbf{x}^i_k, \mathbf{u}^i_k) := \sum_f \left(\|\mathbf{f}^i_{k,f}\|^2_{\mathbf{W}_f} + \|\mathbf{r}^i_{k,f} - \mathbf{r}^{i*}_{k,f}\|^2_{\mathbf{W}_r}\right) + \tag{13a}$$
$$\|\mathbf{f}^i_{k,h}\|^2_{\mathbf{W}_f} + \|\boldsymbol{\tau}^i_{k,h}\|^2_{\mathbf{W}_\tau}\,, \tag{13b}$$

where we set the reference footholds $\mathbf{r}^{i*}_{k,f}$ by projecting the reference hip joint positions—which, in turn, depend on $\mathbf{p}^{i*}_k$ and $\mathbf{q}^{i*}_k$—onto the ground.

Finally, we combine the above terms in (1a) by defining

$$l_k := w^0 l^0_{k,x} + \sum_{i>0}\left(l^i_{k,x} + l^i_{k,u}\right)\,,$$
$$l_N := w_N\left(w^0 l^0_{k,x} + \sum_{i>0} l^i_{k,x}\right)\,,$$

where $w^0 \in \mathbb{R}_{>0}$ gives higher priority to payload tracking objectives, $w_N \in \mathbb{R}_{>0}$ is a final cost weight, and we omit the function arguments for the sake of readability.

Reference Trajectories: We employ an object-centric strategy to model the cost function of our TO. In particular, we compute reference trajectories for the payload by integrating planar velocity commands input by a user after having rotated them according to the local ground orientation [20]. We subsequently determine references for each robot based on the payload targets: this approach makes the trajectories of the robots secondary to the manipulation goals and reduces the user's responsibilities to direct CLM.

Since each robot's hand is rigidly attached to the payload, we can compute their reference pose as a function of $\mathbf{x}_k^{0*}$: let the target position and orientation of the ith robotic hand be $\mathbf{p}_{k,h}^{i*}$ and $\mathbf{q}_{k,h}^{i*}$, respectively. As per Assumption 2, we also define the default position and orientation of the hand relative to the robot's frame $\mathcal{B}^i$ as $\hat{\mathbf{r}}_h^i$ and $\hat{\mathbf{q}}_h^i$. For simplicity, we always define $\hat{\mathbf{q}}_h^i = 1$, the identity quaternion, although our formulation is trivially generalizable.

Given the above information, for each robot $i > 0$, we write its reference orientation as $\mathbf{q}_k^{i*} = \mathbf{q}_{k,h}^{i*}$, and

$$\mathbf{p}_k^{i*} = \mathbf{p}_{k,h}^{i*} - \mathbf{q}_k^{i*} * \hat{\mathbf{r}}_h^i \,, \tag{14a}$$

$$\dot{\mathbf{p}}_k^{i*} = \dot{\mathbf{p}}_k^{0*} + \mathbf{q}_k^{0*} * \left[\mathbf{\Omega}_k^{0*} \times (\mathbf{p}_k^{i*} - \mathbf{p}_k^{0*})\right] \,, \tag{14b}$$

$$\mathbf{\Omega}_k^{i*} = \left(\mathbf{q}_k^{i*}\right)^{-1} * \left(\mathbf{q}_k^{0*} * \mathbf{\Omega}_k^{0*}\right) \,. \tag{14c}$$

Equation (14) treats the entire multi-agent system as a single rigid body where all robots are fixed to the payload at a default relative configuration. Then, it calculates the targets for each robot based on the object reference pose and velocity. Although this approach easily leads to impossible trajectories, it directly encourages the satisfaction of Assumption 2 while burdening the TO problem (1) with finding dynamically feasible optimal solutions.

IV. Results

In this section, we present details about our numerical solver and illustrate the capabilities of the proposed CLM approach in simulation. Notably, we release the source code of our solver as part of the project *Ungar* [37], on which we base our implementation.

A. Implementation Details

We solve the CLM optimal control problem (1) using a custom SQP solver based on the recent work by Grandia et al. [20]. We refer the reader to [20] for an in-depth description of the algorithm and hyperparameters, while we only mention the differences here. More specifically, we adopt a different penalty function to enforce the inequality constraints (1e), we do not project the linearized general equality constraints (1d), and we employ a different back-end quadratic programming (QP) solver. Instead of a relaxed barrier function, we penalize inequality constraint violations through a $\mathcal{C}^2$-continuous function that works well in practice [22], namely:

$$\mathcal{S}(h) := \begin{cases} K\left(h^2 + \frac{\epsilon^2}{3}\right) & h \geq \epsilon \\ K\left(-\frac{1}{6\epsilon}h^3 + \frac{1}{2}h^2 - \frac{\epsilon}{2}h + \frac{\epsilon^2}{6}\right) & -\epsilon \leq h < \epsilon \\ 0 & h < -\epsilon \end{cases} \tag{15}$$

for some constants $K, \epsilon \in \mathbb{R}_{>0}$. Also, we do not perform a projection of the general equality constraints because the contact consistency equation (7) spans two consecutive time steps, thus preventing a block diagonal constraint Jacobian and impeding such a projection. For the same reason, we adopt the general sparsity-exploiting OSQP solver [35] to solve the linearizations of (1), which imposes no requirements on the structure of the constraint matrix.

TABLE I
Parameter Values

m^0	$16\,\mathrm{kg}$	$\mathbf{W}_\Omega$	$\mathrm{diag}(1,1,1)$
$\mathbf{I}^0$	$\mathrm{diag}(1.6, 24, 1.6)\,\mathrm{kg}\cdot\mathrm{m}^2$	$\mathbf{W}_r$	$\mathrm{diag}(10,10,10)$
$m^i,\ i > 0$	$25.2\,\mathrm{kg}$	$\mathbf{W}_f$	$10^{-8}\,\mathrm{diag}(1,1,1)$
$\mathbf{I}^i,\ i > 0$	$\mathrm{diag}(0.4, 1, 1)\,\mathrm{kg}\cdot\mathrm{m}^2$	$\mathbf{W}_\tau$	$10^{-8}\,\mathrm{diag}(1,1,1)$
$l_f^i,\ i > 0$	$0.4\,\mathrm{m}$	w^0	10
$l_h^i,\ i > 0$	$0.4\,\mathrm{m}$	w_N	10
μ	0.8	H	$1.6\,\mathrm{m}$
$\mathbf{W}_p$	$\mathrm{diag}(10,10,42)$	κ	0.8
w_q	36	K	42
$\mathbf{W}_{\dot{p}}$	$\mathrm{diag}(1,1,1)$	ϵ	10^{-4}

We adopt a real-time iteration scheme [14] to take advantage of the close OCP instances typical of MPC applications. To this end, we use the solution computed at the previous iteration shifted by one time step as a warm start. When feeding the current state to (1b), we choose the sign of the measured orientation $\mathbf{q}_m$ so that it is closer to the corresponding warm-started initial state $\mathbf{q}_0$; that is, we make the following reassignment:

$$\mathbf{q}_m \leftarrow \underset{\mathbf{q} \in \{\mathbf{q}_m, -\mathbf{q}_m\}}{\arg\min} \|\mathbf{q}_0 - \mathbf{q}\| \,.$$

Without this change, even if $\mathbf{q}_0$ represented the same orientation of $\mathbf{q}_m$, the sign ambiguity of unit quaternions would spoil the initialization, and the forward propagation through (4b) would lead to catastrophic optimization results. In this regard, we remark that the orientation cost terms (12b) are, by construction, indifferent to the signs of the reference quaternions.

We generate all the required derivatives using the open-source C++ library *CppADCodeGen* [26]. Though entirely serial, our implementation is able to run stably at 30 Hz while optimizing CLM trajectories with 2 robots over a 0.5 s time horizon on a laptop computer with an i7-11800H, 2.30 GHz, 16-core CPU. The frequency drops to ∼25 Hz and ∼18 Hz for time horizons of ∼0.66 s and ∼1.03 s, respectively. However, due to the nature of multiple shooting methods, most parts of our MPC are amenable to parallelization [13], and with aggressive code optimizations we expect to make our centralized approach scale to up to 3 robots and 1 s horizon at fast enough rates for robust control[2].

B. Simulation Experiments

We test our CLM formulation in simulations with 2 robots carrying a payload on challenging terrains using a trot gait. In all our experiments, we adopt a time horizon of $N = 15$ discrete time steps of duration $\Delta t \approx 0.033\,\mathrm{s}$; we list the remaining parameter values of each setup in Table I. In the scope

[2]Grandia et al. [20] achieve the impressive control frequency of 100 Hz with OCPs of around 5000 decision variables, while a CLM with 3 robots, $\Delta t \approx 0.033\,\mathrm{s}$ and a 1 s horizon would consist of "only" 4260. In the literature, rates of 25 Hz are often proven sufficient to achieve reliable MPC performance on quadruped robots [4, 19].

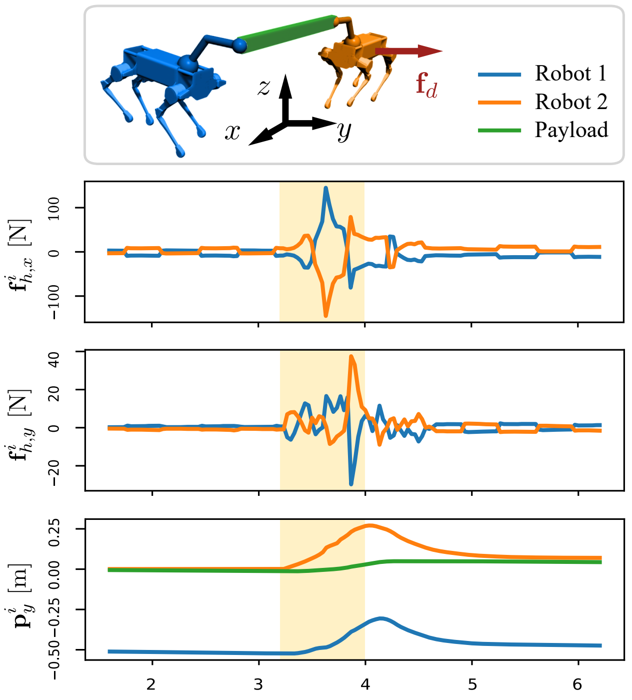

Fig. 5. Manipulation forces and body positions in a two-robot CLM after an unexpected disturbance. At $t = 3.2\,\mathrm{s}$, we push one of the robots (**orange**) with a $164\,\mathrm{N}$ force $\mathbf{f}_d$ acting on the center of mass along the y-direction for $0.8\,\mathrm{s}$. In the top two diagrams, we plot the x- and y-components of the manipulation forces acting on the payload, respectively; at the bottom, we represent the y-coordinate of all rigid bodies' positions. As shown in the accompanying video and in the force plots, both robots pull the object to maintain their balance. From the final diagram, we can see that the unperturbed robot (**blue**) makes a step in the y-direction toward the payload to reduce the moment arm of the hand about its center of mass. Most importantly, the robots successfully keep the payload (**green**) in place throughout the experiment.

of this paper, we use two *Laikago* robots [2] augmented with a four-degree-of-freedom virtual mechanical arm modeled after the *DynaArm* [34]. However, in the accompanying video, we also show the versatility of our controller with three- and four-robot demonstrations involving heterogeneous agents—i.e., *Laikago*, *A1*, *Aliengo* [2]—and different gait patterns—i.e., trot, crawl, pronk.

Finally, to assess the quality of the trajectories discovered by our CLM planner, we employ non-physical environments where the optimal state plans are tracked by IK instead of WBC (cf. Figure 2). To this end, we convert discrete time quantities to continuous time using a simple linear interpolation strategy, and we generate foot trajectories given their stepping locations and timings as cubic splines.

Disturbance Rejection: CLM systems depend on the coordination of different agents. If one of the robots faces an unexpected perturbation, the controller must react and replan all agents' trajectories accordingly. We test our MPC by pushing one of the two robots while commanding them to keep the payload in place. Specifically, we apply a sizable $164\,\mathrm{N}$ force in the y-direction for $0.8\,\mathrm{s}$.

To evaluate the emergence of cooperative behaviors, we plot the manipulation forces and positions of each robot in Figure 5. When the push begins, both agents start pulling the object along its longitudinal dimension, thus helping the attacked robot to regain balance. We also observe that the undisturbed robot steps toward the payload in response to the perturbation. This move reduces the moment arm and, consequently, the torque induced on the base by the manipulation force, which would have been much higher otherwise due to the push. In this context, collaboration and foothold optimization play a fundamental role in preventing the fall of either robot.

Ditches: We evaluate the ditch crossing constraint (11) by randomly generating 12 ditches aligned with the y-axis with a maximum width d_j of $32\,\mathrm{cm}$. This allows us to simplify (11) as:

$$\frac{d_j}{2} - \left|\mathbf{p}^i_{k,f,x} - x_j\right| \leq 0\,, \tag{16}$$

where x_j is the x-coordinate of any point along the jth ditch center line. Then, we let the robots carry an object using a variety of gait patterns while avoiding the gaps on the ground. In this experiment, (11) constrains all robots' feet at all time steps, which translates to 2880 inequalities.

Handling inequality constraints through penalty functions requires careful consideration of how the constraints are formulated—see Appendix A. The proposed CLM consistently converges to a feasible solution with minimal tuning of the foot tracking weights $\mathbf{W}_r$ in (13) and the penalty function parameters in (15). We include a snapshot of the experiment in Figure 1; notably, the optimization can make considerable corrections to the regularizing footholds $\mathbf{r}^{i*}_{k,f}$. This demonstration shows how the ability to adjust stepping locations can be crucial if the ground is not trivially flat. Not only is this skill necessary to deal with rough terrains, but the slight displacement of a planned foothold can also result in much higher torques on a robot's base to promptly counteract unexpected pushes and perturbations. These aspects become especially relevant in CLM, where the high dimensionality of the system implies more numerous potential sources of error.

Ramps: By endowing our TO with foothold optimization, we enable CLM to work on virtually any terrain. Indeed, as delineated in Section III-B, we can make our MPC controller aware of the terrain as long as a parameterization or a local approximation thereof is available. We verify the efficacy of our approach by demonstrating CLM on a smooth ramp, as shown in Figure 1. To this end, we model $z_g(\cdot)$ with a logistic function:

$$z_g(\mathbf{p}^i_{k,f,x}) := \frac{H}{1 + \exp\left[-\frac{4}{H}\kappa(\mathbf{p}^i_{k,f,x} - x_g)\right]}\,, \tag{17}$$

where $H \in \mathbb{R}_{>0}$ is the height of the ramp, $x_g \in \mathbb{R}$ is the position along the x-axis of the ramp center, and $\kappa \in \mathbb{R}$ is the slope value at $\mathbf{x}_g$—see Figure 6. The functions $\mathbf{n}_g(\cdot)$, $\mathbf{t}^{xz}_g(\cdot)$ and $\mathbf{t}^{yz}_g(\cdot)$ required by the positive ground reaction force normal and linearized friction constraints (9) and (10), respectively, can be derived from (17) and we omit them for brevity.

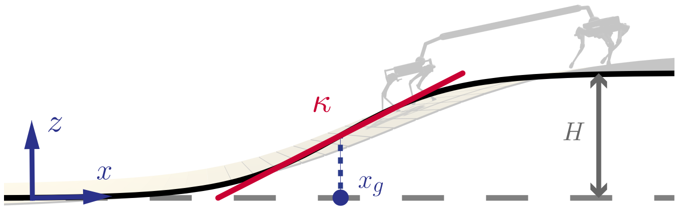

Fig. 6. Geometric representation of a ramp. We employ a logistic function due to its smooth profile and convenient parameterization: H is the maximum height of the ramp, x_g is its location, and κ is its slope at x_g.

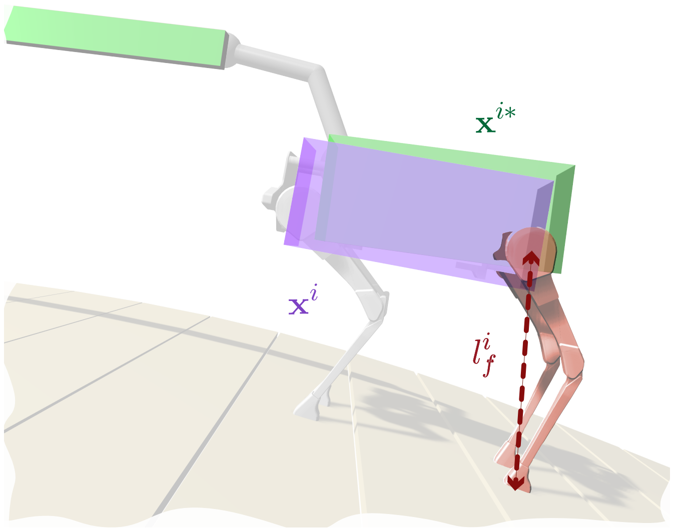

Fig. 7. CLM on the edge of a ramp. When the lowermost robot approaches the peak of the ramp, our method to generate target motions clashes with its leg kinematic limits, and the robot must navigate at the boundary of the corresponding constraint. The reference state $\mathbf{x}^{i*}$ (**green**) lies above the actual state of the robot $\mathbf{x}^{i}$ (**violet**) since the hind legs (**red**) have reached their maximal extension l_f^i.

As described in Section III-C, we project planar velocity commands for the payload onto planes locally tangent to the terrain geometry. In our trials, the CLM system is able to track such targets reliably with slope values up to $\sim 38.7°$ while switching between different gaits. However, near the upper edge of the ramp, the generated reference trajectories for the robots start colliding with the kinematic limits of the legs—see Figure 7. This fact requires precise tuning of the stiffness K of the penalty function (15), so that it successfully meets the constraint (8) without hindering convergence.

C. Limitations

Our OCP is limited by the absence of constraints on robot self-collisions and manipulation forces. While our choice of tracking objectives mitigates both issues, we must carefully address them before migrating our controller to real-world machines. Thus, we will integrate contact wrench cones (CWCs) [16] as an immediate next step, along with a self-collision avoidance formulation adapted to our SRBD.

We further plan to improve how we generate target trajectories for the robots. Although our approach to computing robot references proves effective in most scenarios, the "edge cases" encountered in our ramp experiment advocate the necessity for more complex strategies. To this end, investigating extensions of Raibert's heuristics [5, 20] to irregular and non-flat terrains in cooperative settings would be a valuable research avenue.

The need for real-time motion planning capabilities can be tackled along two threads: on an implementation level, through parallel computing solutions, more efficient QP solvers, or decentralized control schemes; on a modeling level, by employing simpler input parameterizations [22], or different MPC paradigms for high-dimensional systems—e.g., model hierarchy predictive control (MHPC) [27].

For future work, we will couple the resulting MPC with whole-body controllers and deploy it in a proof-of-concept hardware demonstration. This will involve resolving early/late touchdown events [20] and adopting a robust control strategy for object grasping. After successfully integrating these capabilities, we intend to use teams of *ANYmal C* platforms equipped with *DynaArm* manipulators [34] for assisting the onsite fabrication of timber assemblies.

V. Conclusion

In this paper, we presented a principled approach for controlling multiple one-armed quadruped robots toward collaborative loco-manipulation tasks. Our centralized MPC formulation extends the widespread SRBD models developed in the legged robotics literature to interacting, multi-agent systems. By expressing orientations through unit quaternions, we benefit from a compact, singularity-free representation, which can accommodate arbitrary payload trajectories. A Lie group time-stepping method allows us to integrate angular velocities seamlessly while preserving unit norm constraints. Moreover, by including foothold positions in the decision variables, we can apply our approach to non-flat and unstructured terrains, enabling more challenging collaborative tasks than those previously achieved in CLM research. While our serial implementation achieved real-time rates for robot duos over $0.5\,\mathrm{s}$ time horizons, our multiple-shooting strategy is well suited to performance boosts through parallelizations, which will render our framework ripe for hardware realizations.

Appendix A
Impact of Inequality Constraint Formulations

The effectiveness of our penalty function-based approach is naturally influenced by the formulation of inequality constraints. For instance, with regard to the ditch crossing experiment of Section IV-B, we can write the squared form of (16) as:

$$\frac{d_j^2}{4} - \left(\mathbf{p}_{k,f,x}^i - x_j\right)^2 \leq 0\,. \tag{18}$$

Although (18) is mathematically equivalent to (16), the alternative formulation is more prone to producing unfeasible solutions and requires painstaking hyperparameter tuning to achieve satisfactory performance. This fact is due to the tendency of (18) to yield less steep search directions orthogonal to the constraint surfaces when $\mathbf{p}^{i}_{k,f,x}$ approaches x_j. Nevertheless, the absolute value function in (16) proves highly effective in averting the above shortcomings and guaranteeing feasible solutions.

References

[1] Boston dynamics. https://www.bostondynamics.com/, 1992. [Online; accessed 31-January-2022].

[2] Unitree robotics. https://www.unitree.com/en/, 2016. [Online; accessed 31-January-2022].

[3] Dario Bellicoso, Koen Krämer, M. Stäuble, Dhionis V. Sako, Fabian Jenelten, Marko Bjelonic, and Marco Hutter. Alma - articulated locomotion and manipulation for a torque-controllable robot. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8477–8483, 2019.

[4] G. Bledt and Sangbae Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323, 2019.

[5] G. Bledt, P. Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4102–4109, 2017.

[6] G. Bledt, M. J. Powell, B. Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252, 2018.

[7] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[8] Fabrizio Caccavale, Gerardo Giglio, Giuseppe Muscio, and Francesco Pierri. Cooperative impedance control for multiple uavs with a robotic arm. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2366–2371, 2015.

[9] Matthew Chignoli and Patrick M. Wensing. Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797, 2020.

[10] Jiawei Chiu, Jean-Pierre Sleiman, Mayank Mittal, Farbod Farshidian, and Marco Hutter. A collision-free mpc for whole-body dynamic locomotion and manipulation. *2022 International Conference on Robotics and Automation (ICRA)*, pages 4686–4693, 2022.

[11] Preston Culbertson, Jean-Jacques E. Slotine, and Mac Schwager. Decentralized adaptive control for collaborative manipulation of rigid bodies. *IEEE Transactions on Robotics*, 37:1906–1920, 2020.

[12] Hongkai Dai, Andres Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, 2014.

[13] M. Diehl, H. Bock, H. Diedam, and Pierre-Brice Wieber. Fast direct multiple shooting algorithms for optimal robot control. 2005.

[14] M. Diehl, H. Bock, and J. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control. Optim.*, 43:1714–1736, 2005.

[15] Yanran Ding, A. Pandala, Chuanzheng Li, Young-Ha Shin, and Hae-Won Park. Representation-free model predictive control for dynamic motions in quadrupeds. *IEEE Transactions on Robotics*, 37:1154–1171, 2021.

[16] Yanran Ding, Charles Khazoom, Matthew Chignoli, and Sangbae Kim. Orientation-aware model predictive control with footstep adaptation for dynamic humanoid walking. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 299–305, 2022. doi: 10.1109/Humanoids53995.2022.10000244.

[17] Hamed Farivarnejad, Sean Wilson, and Spring Berman. Decentralized sliding mode control for autonomous collective transport by multi-robot systems. *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1826–1833, 2016.

[18] Randall T. Fawcett, Leila Amanzadeh, Jeeseop Kim, A. Ames, and Kaveh Akbari Hamed. Distributed data-driven predictive control for multi-agent collaborative legged locomotion. *ArXiv*, abs/2211.06917, 2022.

[19] R. Grandia, Farbod Farshidian, Ren'e Ranftl, and M. Hutter. Feedback mpc for torque-controlled legged robots. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737, 2019.

[20] Ruben Grandia, Fabian Jenelten, Shao-Hua Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model predictive control. *ArXiv*, abs/2208.08373, 2022.

[21] Dong Jin Hyun, Sangok Seok, Jongwoo Lee, and Sangbae Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *Int. J. Rob. Res.*, 33(11):1417–1445, September 2014. ISSN 0278-3649. doi: 10.1177/0278364914532150. URL https://doi.org/10.1177/0278364914532150.

[22] Dongho Kang, Flavio De Vincenti, and Stelian Coros. Nonlinear model predictive control for quadrupedal locomotion using second-order sensitivity analysis. *ArXiv*, abs/2207.10465, 2022.

[23] W. Monroe Keyserling. Analysis of manual lifting tasks: a qualitative alternative to the niosh work practices guide. *American Industrial Hygiene Association journal*, 50 3:165–73, 1989.

[24] Jeeseop Kim and Kaveh Akbari Hamed. Cooperative locomotion via supervisory predictive control and distributed nonlinear controllers. *Journal of Dynamic Systems, Measurement, and Control*, 2021.

[25] Jeeseop Kim, Randall T. Fawcett, Vinay R. Kamidi, A. Ames, and Kaveh Akbari Hamed. Layered control for cooperative locomotion of two quadrupedal robots: Centralized and distributed approaches. *ArXiv*, abs/2211.06913, 2022.

[26] João Rui Leal and Brad Bell. Cppadcodegen. 2016.

[27] He Li, Robert J. Frei, and Patrick M. Wensing. Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, 6:3373–3380, 2020.

[28] Toni Machado, Tiago Malheiro, Sérgio Monteiro, Wolfram Erlhagen, and Estela Bicho. Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3111–3117, 2016.

[29] Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay R. Kumar. Cooperative grasping and transport using multiple quadrotors. In *International Symposium on Distributed Autonomous Robotic Systems*, 2010.

[30] Maria Vittoria Minniti, Farbod Farshidian, Ruben Grandia, and Marco Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *IEEE Robotics and Automation Letters*, 4:3687–3694, 2019.

[31] Michael Neunert, Markus Stäuble, Markus Giftthaler, C. Dario Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3:1458–1465, 2018.

[32] Hai-Nguyen Nguyen, Sangyul Park, Junyoung Park, and Dongjun Lee. A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators. *IEEE Transactions on Robotics*, 34:353–369, 2018.

[33] Guilherme A. S. Pereira, Mario Fernando Montenegro Campos, and Vijay R. Kumar. Decentralized algorithms for multi-robot manipulation via caging. *The International Journal of Robotics Research*, 23:783 – 795, 2004.

[34] Jean-Pierre Sleiman, Farbod Farshidian, Maria Vittoria Minniti, and Marco Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6:4688–4695, 2021.

[35] Bartolomeo Stellato, Goran Banjac, Paul J. Goulart, Alberto Bemporad, and Stephen P. Boyd. Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, pages 1–36, 2017.

[36] Andrea Tagliabue, Mina Kamel, Roland Y. Siegwart, and Juan I. Nieto. Robust collaborative object transportation using multiple mavs. *The International Journal of Robotics Research*, 38:1020 – 1044, 2017.

[37] Flavio De Vincenti and Stelian Coros. Ungar – a c++ framework for real-time optimal control using template metaprogramming. https://github.com/fdevinc/ungar, 2023. [Online; accessed 12-May-2023].

[38] Jad Wehbeh, Shatil Rahman, and Inna Sharf. Distributed model predictive control for uavs collaborative payload transport. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11666–11672, 2020.

[39] Hyunsoo Yang and Dongjun Lee. Hierarchical cooperative control framework of multiple quadrotor-manipulator systems. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4656–4662, 2015.