2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Dynamic Legged Manipulation of a Ball Through Multi-Contact Optimization

Chenyu Yang, Bike Zhang, Jun Zeng, Ayush Agrawal and Koushil Sreenath

***Abstract*— The feet of robots are typically used to design locomotion strategies, such as balancing, walking, and running. However, they also have great potential to perform manipulation tasks. In this paper, we propose a model predictive control (MPC) framework for a quadrupedal robot to dynamically balance on a ball and simultaneously manipulate it to follow various trajectories such as straight lines, sinusoids, circles and in-place turning. We numerically validate our controller on the Mini Cheetah robot using different gaits including trotting, bounding, and pronking on the ball.**

I. Introduction

A. Motivation

Dynamic legged manipulation is an important strategy for humans and animals to interact with environments. For example, manipulation tasks like dribbling a ball, walking on stilts, and playing soccer, all require dexterous legged manipulation skills with dynamic interaction with the manipulated object [9], [12], [15]. Not only does this repertoire extend the scope of robotic manipulation, it also paves the way to achieve agile locomotion on extremely difficult terrain [17], for instance, walking on rolling boulders or toppling stepping stones. Enabling legged robots to manipulate objects using their legs shows highly dynamic motion ability and pushes the limits of robots' agility and dexterity.

B. Challenges

Dynamic manipulation using legs places several challenges with regards to control design: 1) The problem involves designing feedback controllers for legged robots to be able to interact with objects. 2) The manipulated object is usually unactuated, which increases the degree-of-underactuation of the legged robot. 3) In addition to just interacting with the object, the robot is often required to manipulate the object along a reference trajectory or to a desired pose. 4) The manipulation of the object occurs through the locomotion of the robot which is governed by the unilateral and friction contact constraints between the robot and the object. 5) Furthermore, the problem combines the challenges of legged locomotion, including hybrid and nonlinear dynamics with high degree-of-underactuation, as well as challenges of non-prehensile manipulation.

The authors are with the Department of Mechanical Engineering, University of California, Berkeley, California, CA 94720, USA {yangcyself, bikezhang, zengjunsjtu, ayush.agrawal, koushils}@berkeley.edu.

This work was partially supported through National Science Foundation Grant CMMI-1944722.

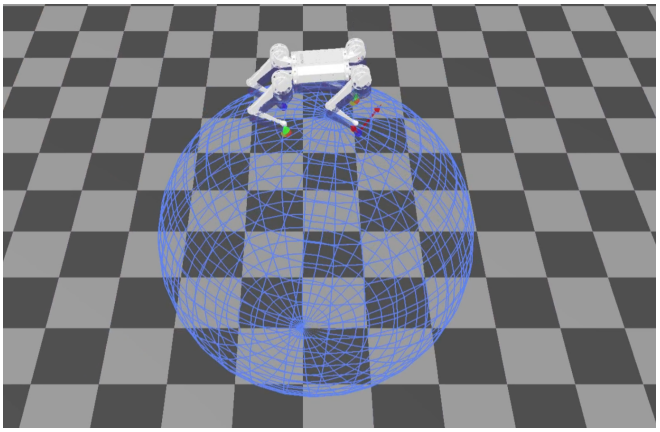


Fig. 1: Simulation snapshot of Mini Cheetah dynamically balancing on and manipulating a ball. The ball has a rigid surface and the contact force between Mini Cheetah and the ball is represented by red arrows. Simulation video is at https://youtu.be/rIVkfudC4_8.

In order to address these challenges, we choose a typical scenario for analysis in this paper: a quadrupedal robot dynamically manipulating a ball to follow different trajectories while balancing on it, shown in Fig. 1. A Mini Cheetah robot [7] and a rigid ball are used and the interaction between them is only through contact.

C. Related Work

There are several quadrupedal robot platforms developed for different tasks including manipulation [5], [6], [10], [13]. These manipulation tasks are usually achieved by attaching a manipulator on a legged robot [1], [11]. However, this approach does not exploit the ability of legs for manipulation tasks.

Legs of a quadrupedal robot are used for both manipulation and locomotion in [14], wherein the legs statically manipulate a box by holding it on both sides. In other words, two legs function as two manipulators and they do not simultaneously achieve manipulation and locomotion tasks.

A dynamic legged manipulation task has been partially implemented in [16] for a bipedal robot balancing on and manipulating a ball. It consists of a balance controller and a footstep planner but only for 2D implementation. Our proposed method aims to provide a comprehensive control framework for 3D dynamic legged manipulation with application to Mini Cheetah manipulating a ball.

D. Contribution

The contribution of this paper is fourfold:

1) Dynamic legged manipulation: We formulate the legged manipulation task of a quadrupedal robot dynamically manipulating a ball in a systematic way by decoupling the whole system via contacts.

2) Interaction model: We develop a simplified interaction model between a quadrupedal robot and a ball based on the concept of equivalent generalized mass.

3) Reaction-force-oriented MPC (R-MPC): We design a MPC strategy by taking contact forces into account to achieve the goal of dynamic ball manipulation along a given trajectory.

4) Foot placement controller: We present a constrained quadratic program-based foot placement controller which adapts to a spherical surface.

E. Paper Structure

This paper is organized as follows. In Sec. II, we introduce the assumptions of the ball and analyze the full dynamics with the ball and the simplified dynamics. The proposed control design is illustrated in Sec. III. In Sec. IV, we present simulation results for a Mini Cheetah robot. In Sec. V, we discuss advantages and limitations of our work, and we summarize the paper in Sec. VI.

II. Dynamics

With the goal of dynamically manipulating a ball using legs, we analyze the interaction between the robot and the ball in this section. We first make assumptions of the ball model, and then introduce the dynamical model of the robot with the ball as well as the simplified model.

We consider a rigid body model of the ball with the following assumptions:

1) The ball does not deform under the influence of external contact forces.
2) We know the physical properties of the ball including its radius, inertia, and friction parameters.
3) We know the ball's states including its position and velocity.
4) There is no slip between the ball and the foot, and the ball and the ground.

A. Dynamics of Cheetah on Ball

We introduce the robot's dynamical model with the consideration of the interaction with the ball. Rather than solving the dynamical equations of the ball and the robot together, as we will see, we integrate the ball's effect into the robot's model. The dynamics of the robot and the ball can be described as follows,

$$\boldsymbol{A}_r\ddot{\mathbf{q}}_r + \mathbf{b}_r + \mathbf{g}_r = \boldsymbol{\tau} + \boldsymbol{J}_r^T\mathbf{f}, \tag{1}$$

$$\boldsymbol{A}_b\ddot{\mathbf{q}}_b + \mathbf{b}_b + \mathbf{g}_b = -\boldsymbol{J}_b^T\mathbf{f}, \tag{2}$$

where $\mathbf{q}_r \in \mathbb{R}^{6+n_j}$ represents the pose of the floating base and the joints of the robot, and n_j is the number of joints. $\mathbf{q}_b \in \mathbb{R}^3$ represents the pose of the ball, and $\boldsymbol{A}_{r/b}$, $\boldsymbol{b}_{r/b}$, $\boldsymbol{g}_{r/b}$, $\boldsymbol{J}_{r/b}$ are the generalized mass matrix, Coriolis force, gravitation force and contact Jacobian for robot or ball, respectively. The contact force between the robot and the ball is denoted by $\mathbf{f}$, and the generalized torque of the robot actuators is represented by $\boldsymbol{\tau}$. Here, we assume that there is no sliding between the ball and the ground, so that we can describe the ball's motion through its Euler angles as $\mathbf{q}_b \in \mathbb{R}^3$ with the ball's x/y position derived from these angles. Note that, with this assumption, we do not have to consider the force between the ground and the ball, and the gravitation term of the ball can be removed in (2).

For the interaction between the robot and the ball, we also assume that there is no sliding, so that we have the following constraints on the acceleration of the contact point,

$$\dot{\boldsymbol{J}}_r\dot{\mathbf{q}}_r + \boldsymbol{J}_r\ddot{\mathbf{q}}_r = \dot{\boldsymbol{J}}_b\dot{\mathbf{q}}_b + \boldsymbol{J}_b\ddot{\mathbf{q}}_b. \tag{3}$$

Substituting $\boldsymbol{J}_b^\dagger\boldsymbol{A}_b\ddot{\mathbf{q}}_b + \boldsymbol{J}_b^\dagger\mathbf{b}_b + \boldsymbol{J}_b^\dagger\mathbf{g}_b = -\mathbf{f}$ from (2) and $\boldsymbol{J}_b^\dagger\dot{\boldsymbol{J}}_r\dot{\mathbf{q}}_r + \boldsymbol{J}_b^\dagger\boldsymbol{J}_r\ddot{\mathbf{q}}_r = \boldsymbol{J}_b^\dagger\dot{\boldsymbol{J}}_b\dot{\mathbf{q}}_b + \ddot{\mathbf{q}}_b$ from (3) into (1), we have

$$\begin{aligned}\boldsymbol{A}_r\ddot{\mathbf{q}}_r + \mathbf{b}_r + \mathbf{g}_r + \boldsymbol{J}_r^T\boldsymbol{J}_b^{T\dagger}\Big[\boldsymbol{A}_b\boldsymbol{J}_b^\dagger(\boldsymbol{J}_r\ddot{\mathbf{q}}_r \\ + \dot{\boldsymbol{J}}_r\dot{\mathbf{q}}_r) - \dot{\boldsymbol{J}}_b\dot{\mathbf{q}}_b + \mathbf{b}_b + \mathbf{g}_b\Big] = \boldsymbol{\tau},\end{aligned} \tag{4}$$

where the $\boldsymbol{J}_b^\dagger$ is a pseudo inverse of $\boldsymbol{J}_b$.

Ignoring the Coriolis force and gravitation terms of the ball $\mathbf{b}_b$ and $\mathbf{g}_b$, and the terms involving the derivatives of Jacobians, we get the equivalent dynamics based on (4) as follows,

$$\tilde{\boldsymbol{A}}\ddot{\mathbf{q}}_r + \mathbf{b}_r + \mathbf{g}_r = \boldsymbol{\tau} + \boldsymbol{J}_r^T\mathbf{f}, \tag{5}$$

where $\tilde{\boldsymbol{A}}$ represents the *equivalent generalized mass* as follows.

$$\tilde{\boldsymbol{A}} = \boldsymbol{A}_r + \boldsymbol{J}_r^T\boldsymbol{J}_b^{T\dagger}\boldsymbol{A}_b\boldsymbol{J}_b^\dagger\boldsymbol{J}_r. \tag{6}$$

We use this equivalent generalized mass to describe the dynamics and generate torques in the whole body impulse control (WBIC) which will be introduced in Sec. III-D.

B. Simplified Model

Having presented the full dynamics model with consideration of the ball, we now present a simplified model of the robot that will be used in the reaction-force-oriented model predictive control (R-MPC) in Sec. III-B. As introduced in [3], we use the lumped mass model,

$$m\ddot{\mathbf{p}}_r = \sum_{i=1}^{n_c}\mathbf{f}_i + \mathbf{g}, \tag{7}$$

$$\frac{d}{dt}(\boldsymbol{I}\boldsymbol{\omega}_r) = \sum_{i=1}^{n_c}\mathbf{r}_i \times \mathbf{f}_i, \tag{8}$$

where $\mathbf{p}_r$, $\boldsymbol{\omega}_r$, and $\mathbf{g}$ are three dimensional vectors denoting the robot's position, angular velocity, and acceleration due to gravity, all in the global frame. The mass and the moment of inertia of the robot are denoted by m and $\boldsymbol{I}$ respectively. n_c is the number of contacts, and $\mathbf{r}_i$, $\mathbf{f}_i$ are the relative position to the center-of-mass and the contact force of the i-th foot, respectively.

During the stance phase, as in [3], the MPC makes three assumptions: 1) roll and pitch angles are small, 2) states are close to the reference trajectory, 3) roll and pitch velocities are small and off-diagonal terms of the inertia tensors are

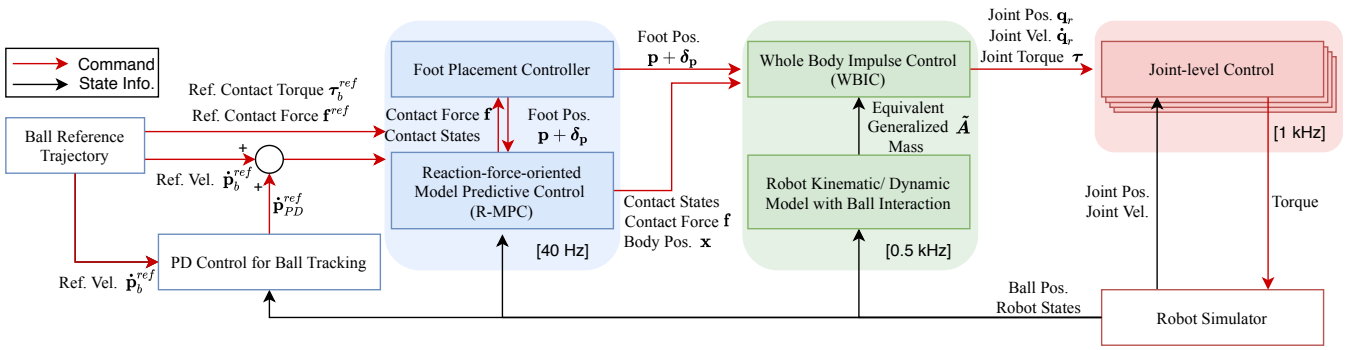


Fig. 2: Control framework for Mini-Cheetah manipulating a ball along a given ball trajectory. Firstly gait types, reference velocity, contact forces and the torque to the ball could be given by user or calculated from ball's reference trajectory. The reaction-force-oriented MPC computes reference contact forces and foot/body position commands, and this allows us to use WBIC to compute joint torque, position, and velocity commands, which is sent to each joint controller. The foot placement controller is added to adjust the planned foot placement, optimizing alternatively with R-MPC. We also consider the interaction between the robot and the ball in the dynamics for WBIC.

small. From [3], the linearized discrete-time dynamics of the system is then expressed as follows,

$$\mathbf{x}(k+1) = \boldsymbol{A}_k\mathbf{x}(k) + \boldsymbol{B}_k\mathbf{f}(k) + \hat{\mathbf{g}}, \tag{9}$$

where $\mathbf{x}$ represents body configurations and velocities, $\mathbf{f}(k)$ and $\hat{\mathbf{g}}$ represent the contact forces from the ball and the robot's gravity. These variables are given by,

$$\mathbf{x} = [\boldsymbol{\Theta}_r^T \quad \mathbf{p}_r^T \quad \omega_r^T \quad \dot{\mathbf{p}}_r^T]^T, \tag{10}$$

$$\mathbf{f} = [\mathbf{f}_1 \quad \mathbf{f}_2 \quad \dots \quad \mathbf{f}_{n_c}]^T, \tag{11}$$

$$\hat{\mathbf{g}} = [\mathbf{0}_{1\times3} \quad \mathbf{0}_{1\times3} \quad \mathbf{0}_{1\times3} \quad \mathbf{g}^T]^T, \tag{12}$$

where $\boldsymbol{\Theta}_r$ is the body orientation. The discrete-time dynamics matrices $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are defined as follows,

$$\boldsymbol{A}_k = \begin{bmatrix} \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{R}_z\Delta t & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3}\Delta t \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} \end{bmatrix}, \tag{13}$$

$$\boldsymbol{B}_k = \begin{bmatrix} \mathbf{0}_{3\times3} & \dots & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \dots & \mathbf{0}_{3\times3} \\ {}_{\mathcal{G}}I^{-1}[\mathbf{r}_1]_\times\Delta t & \dots & {}_{\mathcal{G}}I^{-1}[\mathbf{r}_n]_\times\Delta t \\ \mathbf{1}_{3\times3}\Delta t/m & \dots & \mathbf{1}_{3\times3}\Delta t/m \end{bmatrix}, \tag{14}$$

where ${}_{\mathcal{G}}I$ is the inertia matrix with respect to the global frame, and $\boldsymbol{R}_z$ is the matrix of the body rotation around z-axis. $[\mathrm{x}]_\times \in so(3)$ is defined as the skew-symmetric matrix for cross products. This discrete-time dynamics is then used in the R-MPC in Sec. III-B.

III. Control Design

Having presented the assumptions of the ball as well as the dynamics, we now proceed to present our control design for dynamic legged manipulation.

A. Control Framework

Our proposed control framework extends the control hierarchy in [8] by taking two key issues of ball manipulation into account: 1) how to model the quadruped robot balancing on the ball and 2) how to make use of the contact force and position to drive the ball. The first issue was addressed in Sec. II-A, where we analyzed the interaction model and introduced the *equivalent generalized mass*. For the second issue, we propose a *reaction-force-oriented MPC* (R-MPC) and a *foot placement controller*. The R-MPC is designed to follow the reference state and contact force, see Sec. III-B. The foot placement controller first plans the foot placement according to the Raibert heuristic from [8], and then adjusts the foot placement point to generate a reference torque to the ball, see Sec. III-C. Lastly, the ball reference trajectory and PD control for ball tracking, whose commands are used for R-MPC and the foot placement controller, are described in Sec. III-E.

Our control framework is described in Fig. 2. The ball reference trajectory calculates reference velocity, torque and force of the ball according to different scenarios. A PD controller is added to alter the velocity command, which takes the relative position and velocity of ball and robot into account. The foot placement controller is at the same level with reaction-force-oriented MPC, sharing the commands and the state information. They work together to track the commands of the reference robot state and the reference contact torque and force to the ball. R-MPC and the foot placement controller optimize the foot placement and the contact force profile alternatively. The WBIC tracks the foot position command and the contact force from R-MPC and the foot placement controller using the interaction model. The joint-level control executes commands from WBIC and controls the motors.

B. Reaction-force-oriented MPC

After introducing the control framework, we next present the reaction-force-oriented MPC (R-MPC), which plans the contact force with the simplified dynamics from Sec. II-B using the reference robot trajectory and the reference contact force. The R-MPC minimizes the tracking error and the deviation of contact forces, under the friction cone constraints. It is a constrained quadratic programming (QP) problem, and its formulation is described as follows,

Reaction-force-oriented MPC (R-MPC):

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{f}} \sum_{k=0}^{m} & ||\mathbf{x}(k+1) - \mathbf{x}_r^{ref}(k+1)||_{\boldsymbol{Q}} + ||\mathbf{f}(k) - \mathbf{f}^{ref}(k)||_{\boldsymbol{R}} \\
\text{s.t.} \quad & \mathbf{x}(k+1) = \boldsymbol{A}_k \mathbf{x}(k) + \boldsymbol{B}_k \mathbf{f}(k) + \hat{\mathbf{g}}, \\
& |\mathbf{f}_i^x(k)| \leq \mu \mathbf{f}^z(x) \quad i = 1 \dots n_c, \\
& |\mathbf{f}_i^y(k)| \leq \mu \mathbf{f}^z(k) \quad i = 1 \dots n_c, \\
& \mathbf{f}_i^z(k) \geq 0 \qquad\qquad i = 1 \dots n_c.
\end{aligned} \tag{15}
$$

Here $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weight matrices. The friction cone constraint is simplified to a four-side pyramid constraining the x/y direction of the force of each i-th contact $\mathbf{f}_i^x(k)$, $\mathbf{f}_i^y(k)$. The R-MPC problem optimizes robot state $\mathbf{x}$ and contact force $\mathbf{f}$ that appears as the input to the simplified system in (9). Different from [3], we consider the reference contact forces regulated from the ball's reference trajectory in (24), as we hope to control the robot and the ball at the same time. We compute the reference contact force through a PD control from the ball's reference trajectory.

The reference trajectory $\mathbf{x}_r^{ref}$ is similar with [3]. The reference x/y position of the robot are determined by integrating the reference velocities $\dot{\mathbf{p}}_r^{ref}$. The reference yaw and the yaw rate of the robot is from the commanded direction [8]. The z position of the robot is a user defined constant. The other states (roll, pitch, roll rate, pitch rate, and z velocity) are set to 0.

The calculation of the reference velocities $\dot{\mathbf{p}}_r^{ref}$ and the reference contact forces $\mathbf{f}^{ref}(k)$ will be described in Sec. III-E. Besides being passed to WBIC, the solution of $\mathbf{f}(k)$ is then used as inputs to the foot placement controller, which will be described next in Sec. III-C. The output of the foot placement controller affects the relative position of planned foot placement $\mathbf{r}_i$, which consequently affects the dynamics matrix $\boldsymbol{B}_k$ in the next control iteration.

C. Foot Placement Controller

The foot placement contributes to the torque generated on the ball, so we plan to exploit this potential advantage. In order to find a foot placement that yields desired torque on the ball, the foot placement should ideally be a decision variable in the R-MPC. However, this makes the optimization problem nonconvex, which is computationally expensive to implement in real-time.

As a trade-off between the convexity of the optimization problem and the achievement of the control objective, we formulate the R-MPC and foot placement controller as two separate optimization problems with the results of one being used by the other. Specifically, the R-MPC is solved while holding the foot placement as a constant. Then, the foot placement controller is executed to generate the reference foot placement, and the new foot placement will be used in the R-MPC in the next control iteration.

The foot placement controller only changes the planned foot placement of the swing foot. Once a foot is lifted off

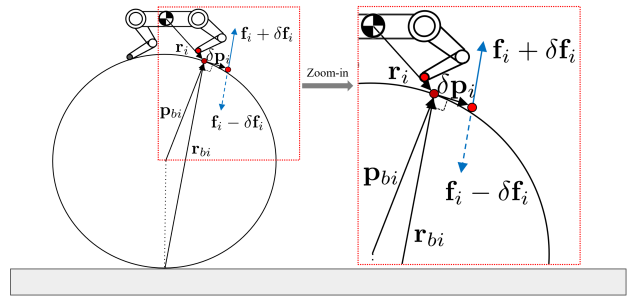


Fig. 3: Mini-Cheetah on the ball and its zoomed view. The foot placement controller adjusts the swing foot placement after R-MPC by calculating a displacement $\delta\mathbf{p}_i$, so that the torque of the contact force at the adjusted foot placement $(\mathbf{r}_{bi} + \delta\mathbf{p}_i) \times (-\mathbf{f}_i - \delta\mathbf{f}_i)$ meets the commanded ball torque.

and becomes a swing foot, its placement is set according to the Raibert heuristic from [8] that is presented as follows,

$$
\mathbf{r}_i^{\text{cmd}} = \mathbf{p}_{\text{shoulder},i} + \mathbf{p}_{\text{symmetry}} + \mathbf{p}_{\text{centrifugal}}, \tag{16}
$$

where,

$$
\begin{aligned}
\mathbf{p}_{\text{shoulder},i} &= \mathbf{p}_r + \boldsymbol{R}_z(\psi_k)\,\mathbf{l}_i, \\
\mathbf{p}_{\text{symmetry}} &= \frac{t_{\text{stance}}}{2}\dot{\mathbf{p}}_r + k\left(\dot{\mathbf{p}}_r - \dot{\mathbf{p}}_r^{\text{ref}}\right), \\
\mathbf{p}_{\text{centrifugal}} &= \frac{1}{2}\sqrt{\frac{h}{g}}\dot{\mathbf{p}}_{\mathbf{r}} \times \omega_r^{\text{ref}}.
\end{aligned} \tag{17}
$$

In (17), $\mathbf{l}_i$ is i-th leg shoulder location with respect to the body frame, and h is the height of the CoM. $\mathbf{p}_{\text{symmetry}}$ is called Raibert heuristic that uses foot placements to stabilize the horizontal CoM dynamics, t_{stance} is the stance duration of the current gait cycle, ω_r^{ref} and $\dot{\mathbf{p}}_r^{\text{ref}}$ are the reference robot angular and linear velocities respectively.

After R-MPC plans the contact force profile, the foot placement controller alters the foot placement using the following optimization program,

Foot Placement Controller:

$$
\begin{aligned}
\min_{\boldsymbol{\delta}_{\mathbf{p}},\boldsymbol{\delta}_{\mathbf{f}}} & \|\boldsymbol{\delta}_{\boldsymbol{\tau}r}(\boldsymbol{\delta}_{\mathbf{p}},\boldsymbol{\delta}_{\mathbf{f}})\|_{Q_r} + \left\|\boldsymbol{\tau}_b^{ref}(\boldsymbol{\delta}_{\mathbf{p}},\boldsymbol{\delta}_{\mathbf{f}}) - \boldsymbol{\tau}_b(\boldsymbol{\delta}_{\mathbf{p}},\boldsymbol{\delta}_{\mathbf{f}})\right\|_{Q_b} \\
& + ||\boldsymbol{\delta}_{\mathbf{p}}||_{\boldsymbol{R}_{\boldsymbol{\delta}\mathbf{p}}} + ||\boldsymbol{\delta}_{\mathbf{f}}||_{\boldsymbol{R}_{\boldsymbol{\delta}\mathbf{f}}} \\
\text{s.t.} & \ \boldsymbol{\delta}_{\mathbf{p}i} \times \mathbf{p}_{bi} = 0.
\end{aligned} \tag{18}
$$

Here, Q_r, Q_b, $R_{\boldsymbol{\delta}\mathbf{p}}$, and $R_{\boldsymbol{\delta}\mathbf{f}}$ are positive definite weight matrices, $\boldsymbol{\delta}_{\mathbf{p}}, \boldsymbol{\delta}_{\mathbf{f}}$ are concatenated vectors of the deviations of contact positions $\boldsymbol{\delta}_{\mathbf{p}i}$ and contact forces $\boldsymbol{\delta}_{\mathbf{f}i}$ of each foot. By ignoring the higher order terms of infinitesimal variations of positions and forces $\boldsymbol{\delta}_{\mathbf{p}i}$ and $\boldsymbol{\delta}_{\mathbf{f}i}$, we can express the change of the contact torque to robot $\boldsymbol{\delta}_{\boldsymbol{\tau}r}$, and the contact torque to the ball $\boldsymbol{\tau}_b$ as follows,

$$
\boldsymbol{\delta}_{\boldsymbol{\tau}r}(\boldsymbol{\delta}_{\mathbf{p}},\boldsymbol{\delta}_{\mathbf{f}}) = \sum_{i=1}^{4} \boldsymbol{\delta}_{\mathbf{p}i} \times \mathbf{f}_i + \mathbf{r}_i \times \boldsymbol{\delta}_{\mathbf{f}i}, \tag{19}
$$

$$\boldsymbol{\tau}_b(\boldsymbol{\delta}_{\mathbf{p}}, \boldsymbol{\delta}_{\mathbf{f}}) = \sum_{i=1}^{4} \mathbf{r}_{bi} \times (-\mathbf{f}_i) + \boldsymbol{\delta}_{\mathbf{p}i} \times (-\mathbf{f}_i) + \mathbf{r}_{bi} \times (-\boldsymbol{\delta}_{\mathbf{f}i}). \tag{20}$$

Here, $\mathbf{r}_i$ and $\mathbf{r}_{bi}$ are the relative position from the robot CoM to foot placement and the relative position from ball's ground contact point to the foot placement, respectively. $\mathbf{p}_{bi}$ is the relative position from the ball's center to the foot contact point. The contact force $\mathbf{f}_i$ comes from R-MPC. We illustrate the geometric relation in Fig. 3.

The foot placement controller is a multi-objective quadratic program where the first term minimizes the resulting torque change on the robot, and the second term tracks the reference torque to be applied on the ball. The third and fourth terms ensure that the solution is in the proximal neighbor of the original point. The optimization constraint guarantees that the optimized foot placement is on the surface of the ball. Note that $\boldsymbol{\delta}_{\mathbf{f}}$ is introduced as a slack variable and it is not used anywhere else.

D. Whole-body Impulse Control

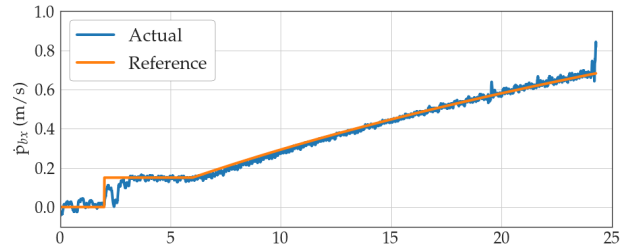
Following the reference from the foot placement controller and the R-MPC, the WBIC tries to incorporate both body posture stabilization and contact force execution with a full dynamics model of the robot. Apart from the use of the equivalent generalized mass $\tilde{\boldsymbol{A}}$, we did not make any other modification to the WBIC described in [8]. The WBIC calculates joint level commands from the reference contact force and body trajectory in the following steps. First, an acceleration command $\ddot{\mathbf{q}}_r^{\text{cmd}}$ of the robot's configuration is computed to execute a set of user-specified prioritized tasks. Then, the following QP is solved,

$$\begin{aligned}
&\min_{\boldsymbol{\delta}_{\mathbf{f}}, \boldsymbol{\delta}_q} \boldsymbol{\delta}_{\mathbf{f}}^\top \boldsymbol{Q}_1 \boldsymbol{\delta}_{\mathbf{f}} + \boldsymbol{\delta}_q^\top \boldsymbol{Q}_2 \boldsymbol{\delta}_q \\
\text{s.t.} \quad & \boldsymbol{S}_f(\tilde{\boldsymbol{A}}\ddot{\mathbf{q}}_r + \mathbf{b}_r + \mathbf{g}_r) = \boldsymbol{S}_f \boldsymbol{J}_r^\top \mathbf{f} && \text{(floating base dyn.)} \\
& \ddot{\mathbf{q}}_r = \ddot{\mathbf{q}}_r^{\text{cmd}} + \begin{bmatrix} \boldsymbol{\delta}_q \\ \mathbf{0}_{n_j} \end{bmatrix} && \text{(acceleration)} \\
& \mathbf{f} = \mathbf{f}^{\text{R-MPC}} + \boldsymbol{\delta}_{\mathbf{f}} && \text{(reaction forces)} \\
& \boldsymbol{W}\mathbf{f} \geq \mathbf{0}\ . && \text{(contact force constraints)}
\end{aligned} \tag{21}$$

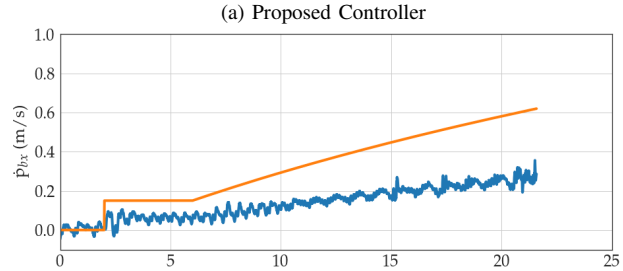
Here, $\boldsymbol{\delta}_{\mathbf{f}}$ and $\boldsymbol{\delta}_q$ are relaxation variables for the reaction forces and the floating base acceleration. $\tilde{\boldsymbol{A}}, \mathbf{b}_r, \mathbf{g}_r, \boldsymbol{J}_r$ are defined in Sec. II-A. The $\boldsymbol{Q}, \boldsymbol{S}_f, \boldsymbol{W}$ are weight matrices of the deviation, the row selection matrix of the floating base, and the matrix of the friction cone and the normal direction of the contact surface. The solution $\boldsymbol{\delta}_{\mathbf{f}}$ is used to adjust the planned contact force $\mathbf{f}$ to satisfy the full dynamics. At last, the torque commands can be computed by plugging the contact forces and the acceleration into the robot's dynamics and passed to the joint level controllers.

E. Ball Reference Trajectory and Tracking

The ball reference trajectory generates the velocity command of the robot as well as the reference contact force and the reference torque to be applied on the ball. Note that the yaw angle of the ball is not considered in the ball reference trajectory.



(a) Proposed Controller



(b) Baseline Controller

Fig. 4: The comparison of reference speed tracking between (a) our proposed controller and (b) the baseline controller. The blue line represents the actual speed, which tries to catch up with an increasing command shown as the orange line.

The reference x/y velocities of the robot, $\dot{\mathbf{p}}_r^{ref}$, are obtained from the reference x/y velocities of the ball $\dot{\mathbf{p}}_b^{ref}$ and a feedback term, $\dot{\mathbf{p}}_{PD}^{ref}$,

$$\dot{\mathbf{p}}_r^{ref} = \dot{\mathbf{p}}_b^{ref} + \dot{\mathbf{p}}_{PD}^{ref}. \tag{22}$$

Here, $\dot{\mathbf{p}}_{PD}^{ref}$ is a feedback component to minimize the tracking error between the instantaneous ball and the robot positions,

$$\dot{\mathbf{p}}_{\text{PD}}^{ref} = -\boldsymbol{k}_p^{\mathbf{v}}(\mathbf{p}_r - \mathbf{p}_b) - \boldsymbol{k}_d^{\mathbf{v}}\dot{\mathbf{p}}_r. \tag{23}$$

where $\mathbf{p}_b$ and $\mathbf{p}_r$ are the position of the ball and the robot. $\dot{\mathbf{p}}_r$ is the velocity of the robot's CoM.

We compute the reference contact force $\mathbf{f}^{ref}$ and the reference torque $\boldsymbol{\tau}_b^{ref}$ through a PD control from ball's reference trajectory,

$$\mathbf{f}^{ref}(k) = -\boldsymbol{k}_p^{\mathbf{f}}\mathbf{e}_p - \boldsymbol{k}_d^{\mathbf{f}}\mathbf{e}_v, \tag{24}$$

$$\boldsymbol{\tau}_b^{ref} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (-\boldsymbol{k}_p^{\boldsymbol{\tau}}\mathbf{e}_p - \boldsymbol{k}_d^{\boldsymbol{\tau}}\mathbf{e}_v), \tag{25}$$

where $\mathbf{e}_p = \mathbf{p}_b^{ref} - \mathbf{p}_b$ and $\mathbf{e}_v = \dot{\mathbf{p}}_b^{ref} - \dot{\mathbf{p}}_b$ represent ball's position and velocity error. $\boldsymbol{k}_p^{\mathbf{f}}$, $\boldsymbol{k}_d^{\mathbf{f}}$, $\boldsymbol{k}_p^{\boldsymbol{\tau}}$, and $\boldsymbol{k}_d^{\boldsymbol{\tau}}$ are PD gains.

IV. Results

Having introduced our control design for dynamic legged manipulation, we next present simulation results that validate the control strategy in this section.

A. Simulation Setup

The simulation environment is set up using the MIT Cheetah Software[1], and a compliant contact model is utilized

[1] https://github.com/mit-biomimetics/Cheetah-Software

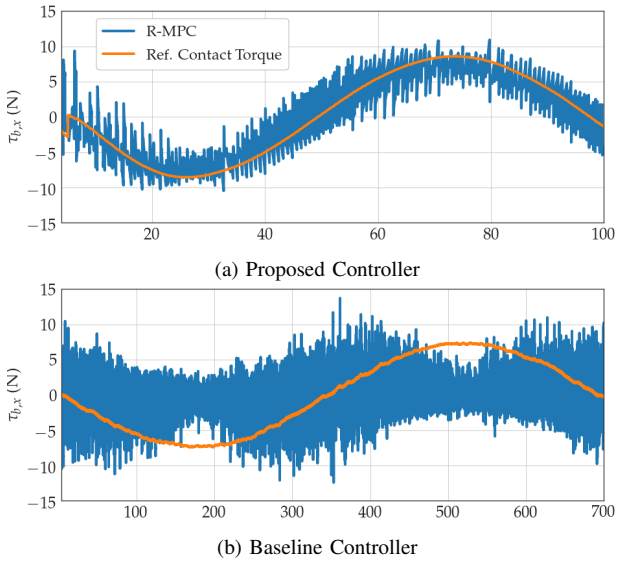


Fig. 5: The comparison of reference ball torque command tracking between (a) our proposed controller and (b) the baseline controller. The blue line shows the torque that can be exerted on the ball from the R-MPC, which is a cross product of current relative contact position and the contact force. The orange line is the ball's torque command from the ball reference trajectory.

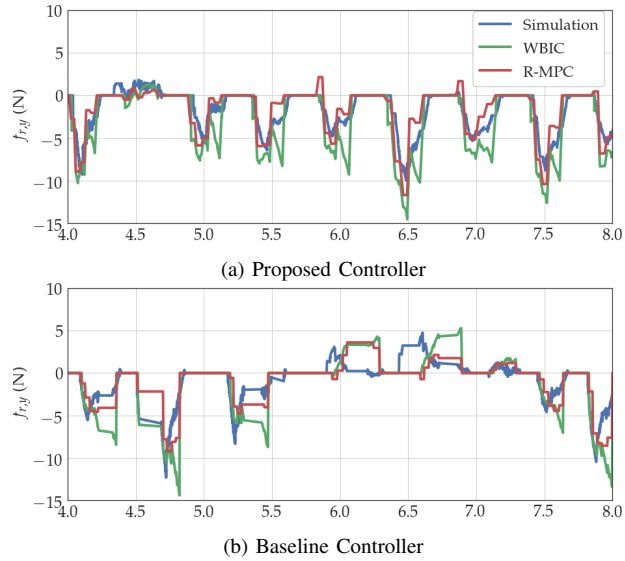


Fig. 6: The comparison of the contact force tracking in y-direction between (a) our proposed controller and (b) a baseline controller. Curves shown in the figure are the planned contact force from R-MPC (red), the contact force solved in WBIC (green), and the actual contact force in simulation (blue)

to compute interaction forces between the robot and the ball based on Featherstone's algorithm [4]. The friction coefficient μ is set as 0.9. The ball's radius is 1m. We use the controller from [8] that is designed to walk on flat ground as our *baseline controller*. Specifically, the baseline controller uses the control framework in Fig. 2 without the foot placement controller, with Robot Kinematic/Dynamic Model of solely the robot, and original MPC instead of R-MPC. Note that the PD Control for Ball Tracking module is included and tuned separately in both our proposed controller and the baseline controller. Also, both the proposed controller and the baseline controller have adapted the direction of gait to the normal direction at the contact point. The ball is created from blender [2] with 2562 points.

B. Performance Evaluation

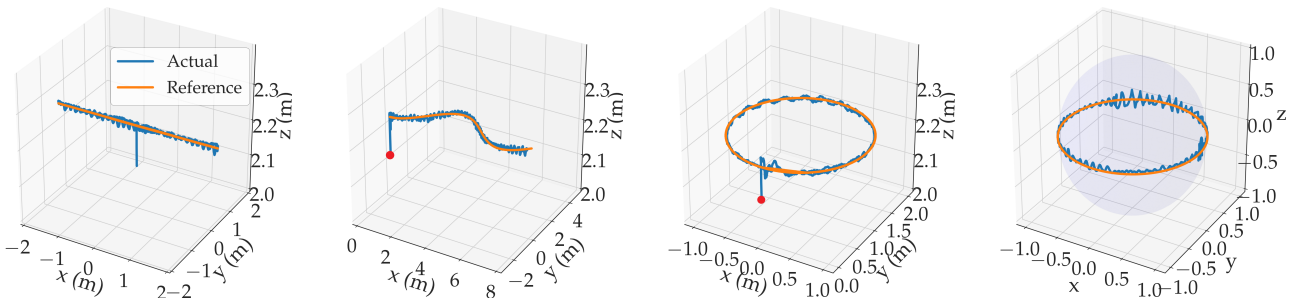
The speed tracking performance is shown in Fig. 4. We set the reference velocity in ball reference trajectory to be a step followed by a ramp, and the yaw rate to be zero. Our proposed controller can track the reference and continuously accelerate until 0.75 m/s. The baseline controller can also track a step input but with more tracking errors. Under mild acceleration, the baseline controller maintains stability as well. However, it can not get close to the 0.75m/s speed limit, and it has a steady state error for tracking a constant acceleration. The proposed control strategy improves the speed tracking performance and extends the range of speed for manipulating a ball.

Fig. 5 shows the effectiveness of our proposed control design for reference ball torque command tracking. The robot manipulates a ball to follow a circular trajectory in this scenario. The orange line is the ball's torque command from the ball reference trajectory, as shown in (25). The blue line shows the torque that can be exerted on the ball from the R-MPC and the foot placement controller, which is a cross product of current relative contact position and the contact force. While the proposed controller keeps tracking the reference commands, the baseline controller loses tracking and is more noisy.

Fig. 6 evaluates how MPC and WBIC track the contact force command in the same scenario of circular trajectory tracking. We only include data from the front left leg, and other legs are similar. In the control hierarchy, the contact force from WBIC should follow the command of R-MPC (proposed controller) or MPC (baseline controller), and by performing the torque commands from WBIC, the real contact force in simulation is at best to be the same as WBIC expected. The closer real contact force to the WBIC's command suggests a more accurate dynamic model. Using our proposed controller with equivalent generalized mass, the contact force in simulation follows the command of WBIC. However, the contact forces of the baseline controller have larger errors in both timing and scale. Using the proposed controller, the mean squared error (MSE) between the real contact force and the WBIC command among four legs of the first 10 seconds is 1810.2, smaller than that of the baseline controller as 2109.4. Note that the commands from R-MPC/MPC are also different, as the baseline controller can not stabilize the robot well.

C. Different Scenarios

Besides comparing the performance with the baseline controller, we implement our controller for different scenarios to further validate our control design.



(a) Tracking a straight line (b) Tracking a sinusoid curve (c) Tracking for a circle (d) Yaw angle tracking

Fig. 7: The 3D tracking performance of Mini Cheetah in different scenarios: (a) tracking a straight line (b) tracking a sinusoid curve (c) tracking a circle (d) orientation tracking. The orange lines are reference trajectories and blue lines are actual trajectories. Red dots indicate the initial position of the robot. The orientation of the robot is shown as point on an unit sphere in (d).

1) Different Gaits: The proposed controller is tested using different gaits. The experimental video shows that trotting, bounding and pronking gaits all work for a Mini Cheetah robot dynamically balancing on a ball.

2) Different Reference Trajectories: Fig. 7 shows the performance of our proposed control design in four different scenarios. In Fig. 7a - 7c, Mini Cheetah manipulates a 2 kg ball to follow a straight line, a sinusoid, and a circle at velocity 0.3 m/s. Note that the reference yaw angle is the angle of the robot, rather than the ball. The reference position and velocity in (24) and (25) are generated from the ball reference trajectory. For example, for the circular trajectory tracking scenario, the ball reference trajectory is given as,

$$\begin{bmatrix} p_{bx}^{ref} \\ p_{by}^{ref} \\ \dot{p}_{bx}^{ref} \\ \dot{p}_{by}^{ref} \end{bmatrix} = \begin{bmatrix} r\sin(tv^{ref}/r) \\ -r\cos(tv^{ref}/r) \\ v^{ref}\cos(tv^{ref}/r) \\ v^{ref}\sin(tv^{ref}/r) \end{bmatrix}, \tag{26}$$

and the robot yaw angle,

$$\psi_r^{ref} = tv^{ref}/r, \tag{27}$$

where $r = 0.7$ m is the radius of the circular trajectory, $v^{ref} = 0.3$ m/s is the reference velocity, and t is the current time. The reference ball velocity $\dot{\mathbf{p}}_b^{ref}$ in (22) is the reference velocity $\dot{p}_{bx/y}^{ref}$ plus a feedback tracking of reference ball position $p_{bx/y}^{ref}$. Notice that the Mini Cheetah starts with position errors from the origin. Fig. 7d shows the orientation of Mini Cheetah following the yaw angle command, which is illustrated by points on a unit sphere. From these plots, we can clearly see that Mini Cheetah tracks these predefined trajectories well.

Fig. 8 shows the detailed tracking errors when Mini Cheetah manipulates a ball to follow a circle: roll, pitch, yaw angle, $\dot{x}$, $\dot{y}$, $\dot{z}$ of Mini Cheetah. We observe that there is a slight tracking delay on tracking yaw angle and robot velocities. In order to make the robot move forward, we need some pitch angle errors to provide torque on the ball and there is a steady state tracking error around 0.2 rad on the pitch angle.

V. Discussion

Ball parameters are important factors affecting the stability and performance of the robot on the ball. The lighter or the smaller the ball is, the more difficult it is to balance on it. Our controller can work with more extreme ball parameters. Without changing the controller parameters, the minimum ball mass and radius for stabilization are 0.5kg and 0.5m, comparing with the 4kg and 0.8m for the baseline controller.

One of the main assumptions that we have made in our control design and simulation is the ball's rigidity. For a more general case, such as stabilizing Mini Cheetah on a deformable object, e.g., a fitness ball, we need to take the deformation of that object into account during the dynamic modeling and control design.

VI. Conclusion

In this paper, we have presented a novel control design for dynamic legged locomotion with an application to a quadrupedal robot manipulating a rigid ball. The control design for ball manipulation consists of a reaction-force-oriented model predictive controller and a foot placement controller, applied to an interaction model and integrated with a nominal WBIC. We numerically verified the control strategy with a variety of scenarios. The proposed controller allows a Mini Cheetah robot to manipulate a ball along straight/sinusoid/circular trajectories and outperforms a baseline controller. Experimental results are envisaged for the future.

Acknowledgement

The authors would like to thank Professor Sangbae Kim and the MIT Biomimetic Robotics Lab for providing the Mini Cheetah simulation software.

References

[1] E. Ackerman, "Boston dynamics' spotmini is all electric, agile, and has a capable face-arm," *IEEE spectrum*, 2016.

[2] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

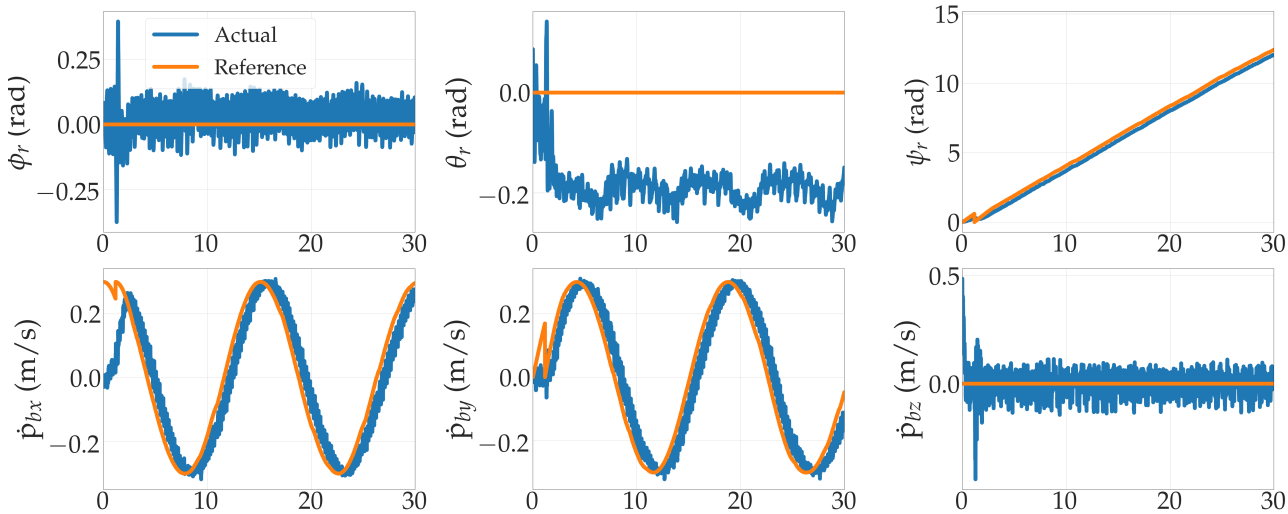


Fig. 8: The simulation results of Mini Cheetah manipulating a ball to track a circle. The orange lines are the reference commands, and the blue lines are the actual values. The first row shows Euler angles (roll ϕ, pitch θ, and yaw ψ), and the second row shows robot velocities ($\dot{x}$, $\dot{y}$, $\dot{z}$).

[3] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[4] R. Featherstone and K. A. Publishers, *Robot Dynamics Algorithm*. USA: Kluwer Academic Publishers, 1987.

[5] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 38–44.

[6] A. M. Johnson and D. E. Koditschek, "Legged self-manipulation," *IEEE Access*, vol. 1, pp. 310–334, 2013.

[7] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 6295–6301.

[8] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[9] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–13, 2017.

[10] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[11] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 3618–3624.

[12] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[13] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of hyq–a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[14] W. Wolfslag, C. McGreavy, G. Xin, C. Tiseo, S. Vijayakumar, and Z. Li, "Optimisation of body-ground contact for augmenting whole-body loco-manipulation of quadruped robots," *arXiv preprint arXiv:2002.10552*, 2020.

[15] J. Z. Wu, S. S. Chiou, and C. S. Pan, "Analysis of musculoskeletal loadings in lower limbs during stilts walking in occupational activity," *Annals of biomedical engineering*, vol. 37, no. 6, pp. 1177–1189, 2009.

[16] Y. Zheng and K. Yamane, "Ball walker: A case study of humanoid robot locomotion in non-stationary environments," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2021–2028.

[17] M. Zucker, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "An optimization approach to rough terrain locomotion," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 3589–3595.