Real-time Trajectory Optimization and Control for Ball Bumping with Quadruped Robots

Qiayuan Liao, Zhefeng Cao, Hua Chen, and Wei Zhang

***Abstract*— This paper studies real-time motion planning and control for ball bumping motion with quadruped robots. To enable the quadruped to bump the flying ball with different initializations, we develop a nonlinear trajectory optimization-based planning scheme that jointly identifies the take-off time and state to achieve accurate ball hitting during the flight phase. Such a planning scheme employs a two-dimensional single rigid body model that achieves a satisfactory balance between accuracy and efficiency for the highly time-sensitive task. To precisely execute the planned motion, the tracking controller needs to incorporate the strict time-state constraint imposed on the take-off and ball-hitting events. To this end, we develop an improved model predictive controller that respects the critical time-state constraints. The proposed planning and control framework is validated with a real Aliengo robot. Experiments show that the problem planning approach can be computed in approximately 60 ms on average, enabling successful accomplishment of the ball bumping motion with various initializations in real-time.**

I. Introduction

Jumping and interacting with objects in the air is one of the most amazing behaviors that animals can perform. Quadrupedal animals like leopards can leap and catch birds; dogs can jump in the air and hit the ball using their heads. Usually, this type of acrobatic behavior consists of multiple phases, including jumping, interacting with the target object, and landing phases. Ball bumping motion, which requires the quadruped to jump into the air and hit a falling ball toward a goal location, is one of the most representative acrobatic motions for quadrupeds. Different from standard quadrupedal locomotion on the ground, such a ball bumping motion is highly time-sensitive, i.e., if the juggler does not arrive at the expected position with the specific velocity at a proper time, then the ball can not reach the desired region. In this paper, we aim to tackle the specific ball bumping problem with quadruped robots, which serves as a starting point for exploring more highly dynamic motion planning and control frameworks for future robotic applications.

A. Related Works

Dynamic locomotion for quadruped robots has attracted a considerable amount of research attention during the past decade. Thanks to the rapid advancements of both hardware and algorithms, quadrupedal robots have demonstrated impressive locomotion skills on flat and uneven terrains [1]–[4].

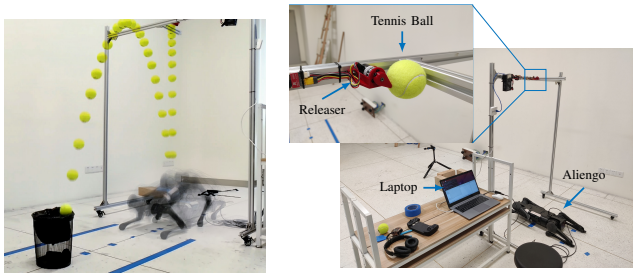

Fig. 1: **Left:** A quadruped robot bumps the ball into a trash can. **Right:** experiment setup.

Fundamentally speaking, achieving jumping motion for quadrupeds can be formulated as trajectory optimization problems in general [5]. MIT Cheetah 3 is capable of jumping onto a desk with a height of 30 inches by an offline trajectory optimization that considers full-body kino-dynamics in a 2D vertical plane [6]. Chigonoli *et al.* [7] proposed a hierarchical planning framework with centroidal dynamics and joint-level kinematics to achieve Omnidirectional jumping, which takes on average 0.55 s to find a reference trajectory plan. Nguyen *et al.* [8] synthesized a full-body trajectory optimization to achieve 3D jump with quadrupeds, which takes several minutes to solve. Among the pioneering works trying to plan and control jumping motions for quadrupeds, Park *et al.* [9], [10] adopted a 2D single rigid body model for quadrupedal dynamics and developed an event-triggered jumping controller that accomplishes jumping over obstacles on MIT Cheetah 2 quadruped. Li *et al.* [11] studied the jumping motion of quadrupeds with only two rear legs and developed a hierarchical planning and control framework that can be implemented in real-time based on a spring-loaded inverted pendulum model [12], which is verified with simulations. As the reversed process of jumping, landing control with quadrupeds has also been considered. Jeon *et al.* [13] developed a supervised learning-based warm start interface for nonlinear landing trajectory planning to improve the performance of quadrupedal landing. How to exploit the conservation of angular momentum to help modulate the robot's configuration during the flight phase has also attracted recent research attentions [14], [15], which further improves the landing performance of quadrupeds. Despite these amazing demonstrations of jumping motion control for quadrupeds, problems studying quadruped jumping with physical interactions with other objects during the flight phase have not been studied adequately in the literature.

More recently, the problem of operating quadrupeds to

All authors are with the Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen, China. liaoqiayuan@gmail.com, 12150041@mail.sustech.edu.cn, chenh6@sustech.edu.cn, zhangw3@sustech.edu.cn. Qiayuan Liao is also with the School of Electromechanical Engineering, Guangdong University of Technology, Guangzhou, China.

physically interact with other objects to achieve more complicated tasks has attracted more research attention. Ji *et al.* [16] considered the soccer shooting problem with a quadruped robot and tackled the problem via a hierarchical reinforcement learning strategy. Shi *et al.* [17] adopted a learning-based approach to enable a quadruped robot to manipulate balls with its legs. All these early investigations focus on scenarios where the target object is static or quasi-static. How to design effective planning and control strategies for interaction with moving objects remains lacking.

In addition to quadrupeds, acrobatic motions for bipedal robots have also been studied extensively in the literature. In particular, investigations on jumping control with bipeds date back to Raibert's seminal works [18], [19]. Hierarchical planning and control frameworks have been long employed in the synthesis of acrobatic bipedal motions. Wensing *et al.* [20] proposed a conic optimization-based task-space control to achieve motions like ball kicking and standing broad jump with a humanoid robot. Xiong *et al.* [21] developed an optimization-based controller for biped hopping with biped Cassie. Chigonoli *et al.* [22] employed the trajectory optimization-based approach for synthesizing dynamic acrobatic motions with a humanoid robot. Reinforcement learning-based approaches have also demonstrated impressive performance in achieving agile biped behaviors [23], [24]. Poggensee *et al.* [25] implemented ball-juggling on biped Cassie, the motion is a periodic orbit, and there is no flying phase while bounding.

B. Contributions

The main contributions of this work are summarized as follows. First, to incorporate the challenging time-state constraint in the ball hitting problem, we develop a real-time optimization scheme to find the critical time-state pair based on a two-dimensional single rigid body model. Such an online planning scheme allows us to find the appropriate take-off state and time to achieve precise ball hitting, which addresses the main challenge in accomplishing the ball bumping problem. Second, to enable the quadruped to accurately execute the planned motion, we develop an improved model predictive controller that actively adjusts the ground reaction forces based on the time remaining for the jumping motion. Such a modification effectively guarantees the execution of the time-restricted motion, making the ball bumping motion practically achievable. Third, we demonstrate the effectiveness and performance of the proposed framework on a real quadruped platform, successfully accomplishing the ball bumping motion with various initializations.

II. Problem Statement

Our goal is to control the quadruped to bump the falling ball to the desired landing point. Throughout this paper, we assume the initial state of the ball is given, including the released position, velocity, and the released time t_s. As depicted in Fig. 2, the whole process can be split into three phases. The first phase is takeoff, in which the quadruped starts to jump at t_j according to the released time t_s of the ball, then executes the online planned jumping motion and finally takes off at t_t. The second is a flying phase, and the quadruped will fly off the ground based on the takeoff velocity, then collide with the ball at t_h in order to let the ball reach the desired landing region. The last one is the landing phase, where the quadruped will keep balance after contact with the ground.

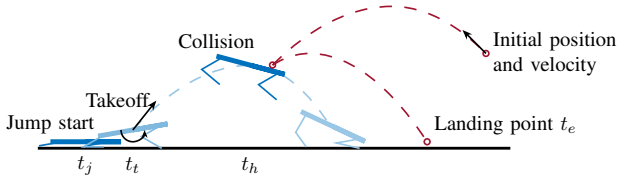

Fig. 2: The process of ball bumping with the quadruped robot.

The ball bumping motion studied in this paper requires an active real-time plan of a feasible trajectory for the quadruped in response to the falling ball's state and accurate execution of the planned trajectory. Such a problem calls for a proper balance between the efficiency and accuracy of the proposed approach. In this section, we provide our modeling of the quadruped-ball system used for planning the ball bumping task and then give an overview of the proposed framework.

A. Modeling of Jumping Quadruped and the Falling Ball

1) quadruped model: In order to simplify the planning of jumping motion while reserving acceptable accuracy, we consider a 2D planar single rigid body model(2D-SRBM) for quadruped in the sagittal plane. The state is the position of the center of mass(COM) and the pitch angle of the quadruped, denoted as $x_q \in SE(2)$. The dynamics are given as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{x}_q\\ \dot{\mathbf{x}}_q\end{bmatrix} = \begin{bmatrix}\mathbf{0}_3 & \mathbf{1}_3\\ \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_q\\ \dot{\mathbf{x}}_q\end{bmatrix} + \begin{bmatrix}\mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 2}\\ \mathbf{1}_2/m & \mathbf{1}_2/m\\ I^{-1}[\mathbf{r}_f]_\times & I^{-1}[\mathbf{r}_r]_\times\end{bmatrix}\begin{bmatrix}\mathbf{f}_f\\ \mathbf{f}_r\end{bmatrix} + \begin{bmatrix}\mathbf{0}_{4\times 1}\\ g\\ 0\end{bmatrix} \quad (1)$$

where $[\mathbf{a}]_\times$ is defined as an operator such that $[\mathbf{a}]_\times \mathbf{b} = \mathbf{a}_1\mathbf{b}_2 - \mathbf{a}_2\mathbf{b}_1$ for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$, $\mathbf{0}_n$ and $\mathbf{0}_{m\times n}$ are the n by n and m by n zeros matrices respectively. $I \in \mathbb{R}^{2\times 2}$ is the inertial matrix in x-z plane. As shown in Fig. 3, $\mathbf{r}_f$, $\mathbf{r}_r$ are the vectors from the COM to the front and rear feet respectively. $\mathbf{f}_f$, $\mathbf{f}_r$ are the corresponding ground reaction forces.

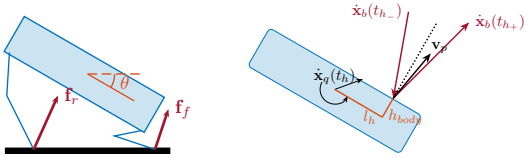

Fig. 3: **Left:**2D single rigid body model; **Right:** collision model.

2) flying ball/quadruped model: Because of the lightweight legs and the short duration of the flying phase, it is hard to adjust the angular momentum of the torso in the air. Therefore, we can consider the position trajectory of the quadruped torso between time t_t and t_h as a ballistic trajectory. As for the

rotation of the quadruped, we believe the robot will keep a constant angular velocity after taking off.

As for the falling ball, if the chosen ball has high-density mass and the velocity is low, then we can neglect the aerodynamic effect. Hence, the motion of the falling ball can also be modeled as a ballistic trajectory before and after the collision.

$$\mathbf{x}_{b/q}(t) = \begin{cases} \mathbf{x}_{b/q}(0) + \dot{\mathbf{x}}_{b/q}(0)t + \frac{1}{2}\mathbf{g}_{b/q}t^2 & , t < t_h, \\ \mathbf{x}_{b/q}(t_h) + \dot{\mathbf{x}}_{b/q}(t_{h_+})(t - t_h) & \\ \qquad + \frac{1}{2}\mathbf{g}_{b/q}(t - t_h)^2 & , t > t_h, \end{cases} \tag{2}$$

where $\mathbf{x}_b \in \mathbb{R}^2$ is the position of the ball in the x-z plane, $\mathbf{x}_q \in SE(2)$ is the 2D state of quadruped, and $\mathbf{g}_b = [0\ g]^T$, $\mathbf{g}_q = [0\ g\ 0]^T$ are gravity vectors.

3) quadruped-ball collision model: The collision model is used to dictate the velocity-changing rules for both the ball and quadruped while the collision occurs. As shown in Fig. 3, we assume the collision will not affect the motion along the tangential direction of the contact surface. According to the conservation of momentum and considering the enormous difference between the mass of the ball and the quadruped, we assume that the quadruped's state will not change after the collision, while the velocity of the ball after the collision is determined as follows

$$\dot{\mathbf{x}}_b(t_{h_+}) = \mathbf{R}_h \left(\begin{bmatrix} 1 & 0 \\ 0 & -e \end{bmatrix} \mathbf{R}_h^T \dot{\mathbf{x}}_b(t_{h_-}) + \begin{bmatrix} 0 & 0 \\ 0 & e \end{bmatrix} \mathbf{R}_h^T \mathbf{v}_p \right),$$

where t_{h_-} and t_{h_+} are the time instants before and after the collision, $\mathbf{R}_h \in \mathbb{R}^{2\times 2}$ represents the orientation from the body to the world frame at t_h, e represents the coefficient of restitution, which is set as 0.8 in our implementation, $\mathbf{v}_q$ is the contact point's velocity of the robot, which is given by

$$\mathbf{v}_p = \begin{bmatrix} \mathbf{1}_2 & \mathbf{R}_h \begin{bmatrix} h_{body} \\ l_h \end{bmatrix} \end{bmatrix} \dot{\mathbf{x}}_q(t_h), \tag{3}$$

where $\mathbf{1}_n$ is the $n \times n$ identity matrix, l_h is the distance between the collision point and COM in the tangential direction of the torso, and h_{body} is the body height.

B. Overview of the Proposed Framework

The whole ball bumping motion is achieved through four modules: jumping motion planning, jumping controller, flying controller, and landing controller. Because of the agility of the jumping motion, a nonlinear program is formulated to get a dynamically feasible trajectory. The details of the formulation will be illustrated in III. Then the model predictive control is used for the jumping tracking controller. Finally, when foot contact is detected, the system will use the landing controller to help the quadruped to keep balance. An overview of this framework is shown in Fig. 4.

III. Real-time Planning of Ball Bumping

The key point of this ball bounding problem is whether the quadruped can reach the desired velocity when it takes off according to the ball's desired landing point. Also, the robot needs to react quickly after detecting the ball, so the planning time needs to be as short as possible. A straightforward way of real-time implementation is to find a pre-collision state of the quadruped leading the post-collision ball to fly along its ballistic trajectory into the desired landing point with the minimum takeoff twist criterion. However, if only given the take-off state and time by the ball's landing point, because of the high-resolution requirement of jumping motion, it is hard for a controller to reach the desired state accurately.

Therefore, we also need to consider how the robot can reach the desired takeoff state. Some works used 3D single rigid body model as the dynamics of the quadruped for the motion planning, but considering the real-time requirement of the algorithm, it can not satisfy our requirement. Because of the specificity of our jumping motion, which is bilaterally symmetrical, we choose the 2D single rigid body model instead. To further simplify the problem to accelerate the computation, we also set the jumping start time t_j manually according to the ball released time to guarantee the planning can be computed before the robot begins to move.

Finally, in order to find one possible take-off state configuration and a dynamically feasible trajectory in real time, we formulate the following nonlinear optimization problem

$$\min_{\mathbf{t},\mathbf{p},\dot{\mathbf{x}}_q(t_h),\mathbf{x}[\cdot],\mathbf{u}[\cdot]} \quad \sum_{k=1}^{N-1} \omega_k \mathbf{u}[k]^T \mathbf{u}[k] \quad \text{s.t.} \tag{4a}$$

$$\text{(Dynamics)} \quad \mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k], \delta t) \tag{4b}$$

$$\text{(Ball's landing point)} \quad \mathbf{x}_b^{des} = p(t_h, \mathbf{x}_b(0), \dot{\mathbf{x}}_b(0), \dot{\mathbf{x}}_q(t_h)) \tag{4c}$$

$$\text{(Collision point)} \quad \underline{l} \le l_h \le \bar{l} \tag{4d}$$

$$\text{(Takeoff Time)} \quad t_j \le t_t \le t_h \tag{4e}$$

$$\text{(Friction cone)} \quad \mathbf{u}[k] \in \mathbf{F}(\mathbf{u}[k]) \tag{4f}$$

$$\text{(Initial conditions)} \quad \mathbf{x}[1] = \mathbf{x}_0 \tag{4g}$$

$$\text{(Takeoff conditions)} \quad \underline{\mathbf{x}} \le \mathbf{x}[N] \le \bar{\mathbf{x}} \tag{4h}$$

where decision variables $\mathbf{x}[k] = [\mathbf{x}_q[k], \dot{\mathbf{x}}_q[k]]^T \in \mathbb{R}^6$ and $\mathbf{u}[k] = [f_f^T[k], f_r^T[k]]^T \in \mathbb{R}^6$ is the state and input of the 2D-SRBM respectively, and $\mathbf{x}[\cdot], \mathbf{u}[\cdot]$ is the state and input trajectories between time t_j and t_t. Here $\mathbf{t} = [t_t, t_h]^T$ is the time frames which need to be optimized, $\mathbf{p} = [l_h, \theta_h]^T$ is the collision configuration waited to be optimized. There are totally N timesteps for this jumping motion, ω_i is the cost weight, $f(\mathbf{x}[k], \mathbf{u}[k], \delta t), p(t_h, \mathbf{x}_b(0), \dot{\mathbf{x}}_b(0), \dot{\mathbf{x}}_q(t_h)), \mathbf{F}(\mathbf{u}[k])$ are the functions for the discrete 2D-SRBM, the ball's landing position, and the friction cone, where $\delta t = \frac{t_t - t_j}{N-1}$ is the timestep for dynamics. And the collision position and takeoff state is bounded by $\underline{l}, \bar{l}$ and $\underline{\mathbf{x}}, \bar{\mathbf{x}}$ respectively.

A. Ball's Landing Point Design

From the above formulation, we optimized the collision time t_h and collision state $\dot{\mathbf{x}}_q(t_h)$ at the same time. And they are associated by the desired landing point of the ball.

After the collision, the ball will fall only under the effect of gravity as described at II-A.2. With the velocity $\dot{\mathbf{x}}_b(t_{h_+})$ and position $\mathbf{x}_b(t_{h_+})$ of the ball after the collision, we can get the landing point of the ball, which is what we need.

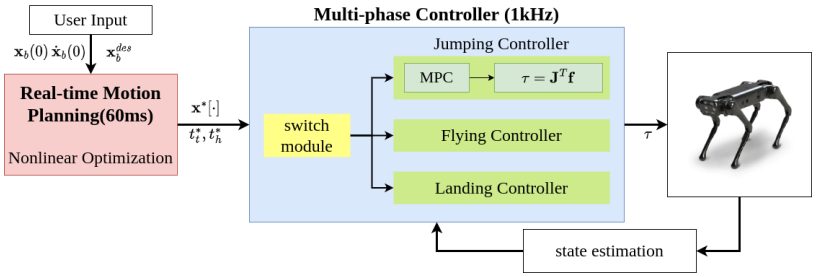

Fig. 4: Motion planning and control framework for ball bumping problem

With z axis value of the velocity $\dot{\mathbf{x}}_{b,z}(t_{h_+})$ and position $\mathbf{x}_{b,z}(t_{h_+})$ of the ball after the collision, we can easily get the time t_e when the ball touch the desired ground, the height of the ground is denoted as $\mathbf{x}_{b,z}(t_e)$, then t_e can be computed

$$t_e = \frac{1}{g}\left(gt_h - \dot{\mathbf{x}}_{b,z}(t_{h_+}) + \sqrt{\begin{array}{l}(\dot{\mathbf{x}}_{b,z}(t_{h_+}) - gt_h)^2 \\ -2g(\mathbf{x}_{b,z}(t_h) - \dot{\mathbf{x}}_{b,z}(t_{h_+})t_h \\ +\frac{1}{2}gt_h^2 - \mathbf{x}_{b,z}(t_e))\end{array}}\right), \tag{5}$$

Then using the dynamics of the ball (equation (2)), we can get the position function for the ball's landing point $\mathbf{x}_b(t_e)$, which depends on t_h, $\mathbf{x}_b(0)$, $\dot{\mathbf{x}}_b(0)$ and $\dot{\mathbf{x}}_q(t_h)$,

$$\mathbf{x}_b(t_e) = p(t_h, \mathbf{x}_b(0), \dot{\mathbf{x}}_b(0), \dot{\mathbf{x}}_q(t_h)). \tag{6}$$

Then because during the planning, we only consider the trajectory from time t_j to t_t, so here we need to induce $\mathbf{x}_q(t_h)$ by $\mathbf{x}_q(t_t)$ using the model described at II-A.2

$$\mathbf{x}_q(t_h) = \begin{bmatrix}\mathbf{x}_b(t_t) - \mathbf{R}_h\begin{bmatrix}l_h \\ r_h\end{bmatrix} \\ \theta_h\end{bmatrix} - \dot{\mathbf{x}}_q(t_t)\Delta t - 0.5\mathbf{g}_q\Delta t^2 \tag{7}$$

$$\dot{\mathbf{x}}_q(t_h) = \dot{\mathbf{x}}_q(t_t) - \mathbf{g}_q\Delta t \tag{8}$$

where $\Delta t = t_h - t_t$, $\mathbf{g}_q = [0\ g\ 0]^T$ is the gravity.

Here we formulate the ball's landing point task as a constraint not cost, the main reason is that the jumping task is time-sensitive, so it's better to add this task into the constraint to guarantee the reference trajectory can lead the quadruped to reach the desired takeoff state accurately at the desired time instance. Besides, as for the computational cost and success rate, we verified that the success rate of putting the task into the constraint or the cost was roughly the same, and they all spent nearly 60 ms to 80ms.

B. Optimization Objective Design

Because we only need to get a feasible solution, so we only choose the input norm $\mathbf{u}[k]^T\mathbf{u}[k]$ as the optimization cost, which means if the problem has a solution, whatever it is globally optimal, it can be used for our jumping motion.

IV. Multi-phase Controller Design for Ball Bumping

As shown in Fig. 4, there are three main phases in this ball bumping task, in order to achieve the desired landing point accurately and keep the whole process safe, we customize the controllers for all these three phases respectively.

A. Jumping Controller

The jumping motion is the most important part of the ball bumping task, and its focus is different from other classical locomotion tasks of quadrupeds, like walking on narrow terrains. Because jumping motion changes the velocity in small duration, which means it is a high-resolution motion for the controller to track. Besides, the most critical challenge is that jumping motion in the ball bumping task is highly time-sensitive. The quadruped needs to reach the desired takeoff state accurately at a specific time; otherwise, the ball will deviate a lot from the goal. According to this property, there should be a strict time-state constraint imposed on the take-off state.

To fulfill the requirements of the jumping motion, we modify the famous MPC controller [1] for our problem.

$$\min_{\mathbf{x}_c,\mathbf{u}_c} \sum_{i=1}^{K-1} \|\mathbf{u}_c[i]\|_{\mathbf{W}} + \|\mathbf{x}_c[i] - \mathbf{x}_{ref}[i])\|_{\mathbf{Q}} + \|\mathbf{x}_c[K] - \mathbf{x}_{ref}[K]\|_{\mathbf{Q}_p} \tag{9a}$$

$$\text{s.t.} \quad \mathbf{x}_c[i+1] = \mathbf{A}\mathbf{x}_c[i] + \mathbf{B}\mathbf{u}_c[i] \tag{9b}$$

$$\underline{\mathbf{c}} \le \mathbf{C}\mathbf{u}_c[i] \le \bar{\mathbf{c}} \tag{9c}$$

where $\mathbf{x}_c[i] = [\boldsymbol{\Theta}^T\ \mathbf{p}^T\ \boldsymbol{\omega}^T\ \dot{\mathbf{p}}^T\ g]^T$, four ground reaction forces produced by legs is the system input $\mathbf{u}_c[i] = [\mathbf{f}_1^T \cdots \mathbf{f}_4^T]^T$, $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^T$ denotes the Euler angles representing the quadruped's orientation. $\mathbf{A} \in \mathbb{R}^{13\times13}$ and $\mathbf{B} \in \mathbb{R}^{13\times12}$ are the parameter matrices of discrete 3D-SRBM. Compared to the standard formula of MPC, we transfer the strict time-state constraints of the jumping motion into the cost, which is set as a terminal cost for the takeoff state.

1) Effects of the Terminal Cost: The performance of the convex MPC has been validated in the MIT Cheetah 3 [1] for high-speed running. However, for the ball bumping motion, which is time-sensitive, we need to make sure the quadruped body can reach the desired position and velocity at the desired time instance.

To solve this issue, we add a terminal cost $\|\mathbf{x}_c[k] - \mathbf{x}_{ref}[k]\|_{\mathbf{Q}_p}$ to help the robot to reach the desired hitting state in time. Both $\mathbf{Q_p}$ and $\mathbf{Q}$ are diagonal matrices, every diagonal entries of $\mathbf{Q_p}$ need to greater than $\mathbf{Q}$'s relatively. This terminal cost can help the controller finds a sequence of control inputs that will guide the system to reach the terminal state at the exact time t_t rather than ahead of t_t.

2) Dynamic Horizon Tuning: During the trajectory tracking process, we will tune the horizon and timestep dt of the

discrete SRBM in order to make sure $\mathbf{x}_q^*(t_t)$ computed by the previous motion planner is always the terminal reference state for the jumping controller. We also use horizon cutting and dynamic $\mathrm{d}t$ to make sure $\mathbf{x}_q^*(t_t)$ is always the terminal reference state of the controller. This tuning method is described in Algorithm 1.

Algorithm 1: Dynamic horizon tuning

Data: current time t_{now}, takeoff time t_t, horizon h, minimum time interval $\mathrm{d}t_{min}$
Result: $\mathrm{d}t$, h
$\mathrm{d}_t \leftarrow (t_{now} - t_t)/h$;
if $dt < dt_{min}$ **then**
  $\mathrm{d}t \leftarrow \mathrm{d}t_{min}$;
end
for $i \leftarrow 1$ **to** h **do**
  if $t_{now} + idt > t_t$ **then**
    $h \leftarrow i$;
  end
end

Finally, the revised MPC problem can be formulated as a quadratic programming problem and quickly solved. Then with the foot jacobian of each leg $\mathbf{J}_i \in \mathbb{R}^{3\times3}$, joint torques of each leg i can be given by

$$\boldsymbol{\tau}_i = \mathbf{J}_i^T \mathbf{R}^T \mathbf{f}_i. \tag{10}$$

B. Flying Controller

As described in II-A.2, the motion trajectory between the takeoff and collision is considered a parabola. Then, to reduce the effect of the leg motion further, we freeze all legs during this process, which means we use the PD control to keep all joint angles constant.

After the collision t_h and a small duration of delay, in order to adjust the leg configuration for landing, we switch to the falling controller; the controller uses Cartesian impedance control:

$$\boldsymbol{\tau}_i = \mathbf{J}_i^T[\mathbf{K}_p(\mathbf{p}_i^{des} - \mathbf{p}_i) + \mathbf{K}_d(\mathbf{v}_i^{des} - \mathbf{v}_i)] \tag{11}$$

to move the foot to desired locations given by

$$\mathbf{p}_i^{des} = \mathbf{p} + \mathbf{p}_i^0, \tag{12}$$

where $\mathbf{p}$ is the robot COM position in the world frame and $\mathbf{p}_i^0$ is the foot position relative to the COM, its x, y axis values are set manually for a stable landing configuration, and the z axis value is the same as the relative z position at the moment of takeoff.

C. Landing Controller

When any leg contact with the ground is detected by the measurement of the foot contact sensor, the controller uses joint space PD control to damp the robot and finally uses MPC described in [1] to squat and lie on the ground.

V. Experimental Results and Discussions

A. Robot Platform

The whole motion planning and control framework was validated on the real robot platform Aliengo [26]. Aliengo from Unitree Robotics is a quadruped robot with 12 actuators, whose legs are designed as open chains, and it has a mass of about 22 kg with lightweight legs, so this mechanical design is suitable for the single rigid body model (SRBM) in our methodology. The physical parameters of Aliengo can be found at [26].

B. Experiment Setup

The motion planning and control framework was developed with `C++` based on `ROS` and `ros_control` [27], which provide hardware interface and controller manager and make simulation (using Gazebo [28]) and debugging efficiency. `pinocchio` [29] and `Eigen` are used as basic kino-dynamics interfaces in the implementation of the methodology. As for the nonlinear optimization programming for the planner, `CasADi` [30] was used for constructing the trajectory optimization problem, and `IPOPT` solver was used for solving it. Quadruped's body mass m was set to 35 kg for increased gains of the system due to the large latency (about 8 ms loop-back) caused by `unitree_legged_sdk`.

The hardware setup is shown in Fig. 1; the software described above was running on a laptop with an i7-8550U processor and runs Ubuntu Linux with a kernel patched with `CONFIG_PREEMPT_RT`. Aliengo was controlled by UDP through an Ethernet cable with a 1000 Hz control loop. We made a tennis ball releaser to guarantee the ball's released time, position, and velocity. The ball releaser was controlled by the laptop with a USB to CAN cable.

C. Experimental Results

1) NLP Test: In order to test the performance of the NLP solver for the planner, we did 10000 test cases with 10 evenly distributed samples per interval over the initial state and desired state $\mathbf{x}_{b,1}(0) \in [0, 0.5]$, $\mathbf{x}_{b,2}(0) \in [1.5, 2.0]$, $\mathbf{x}_{b,1}^{des} \in [0.5, 1.5]$, $\mathbf{x}_{b,2}^{des} \in [0, 0.5]$. All NLPs were solved with a cold start, and the initial guess is set as $\mathbf{t}^0 = [0.2, 0.4], \mathbf{p}^0 = \mathbf{0}_{2\times1}, \dot{\mathbf{x}}_q(t_h)^0 = \mathbf{0}_{3\times1}, \mathbf{x}^0[i] = \mathbf{0}_{3\times1}, \mathbf{u}^0[i] = [0, 0, 175, 0, 0, 175]^T$. The distribution of the solve time is shown in Fig. 5, the solve time of failed solutions is excluded, and the success rate is about 99.84%. The average

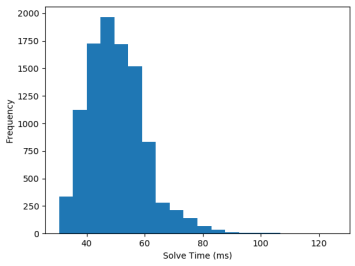

Fig. 5: Solving time of 9984 successful solutions, all cold-start.

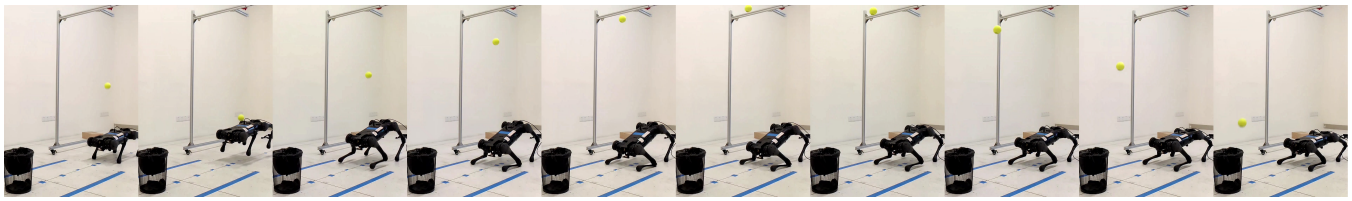

Fig. 6: Time series of the ball bumping experiment

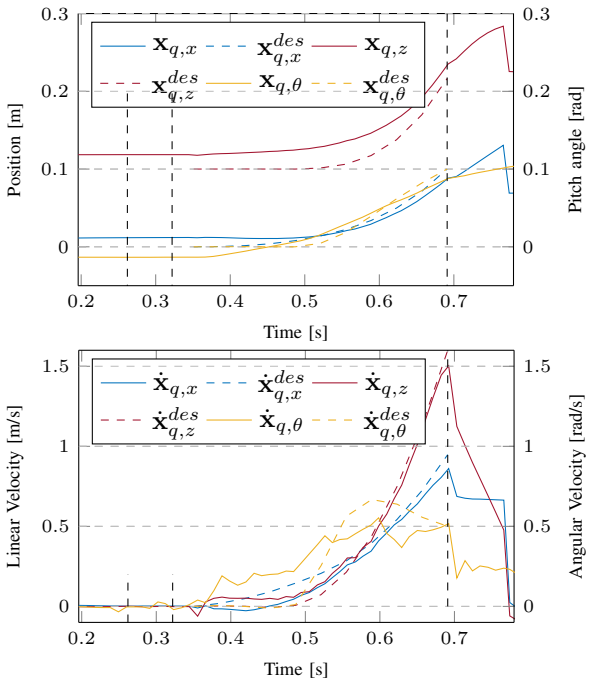

Fig. 7: Planned trajectory and state estimate data in bump ball into a trash can experiment. The initial state of The desired takeoff state is $\mathbf{x}_q^{des}(t_t) = [0.1\ 0.216\ 0.1\ 0.948\ 1.6\ 0.5]^T$, the actual takeoff state is $\mathbf{x}_q(t_t) = [0.088\ 0.235\ 0.087\ 0.86\ 1.5\ 0.516]^T$.

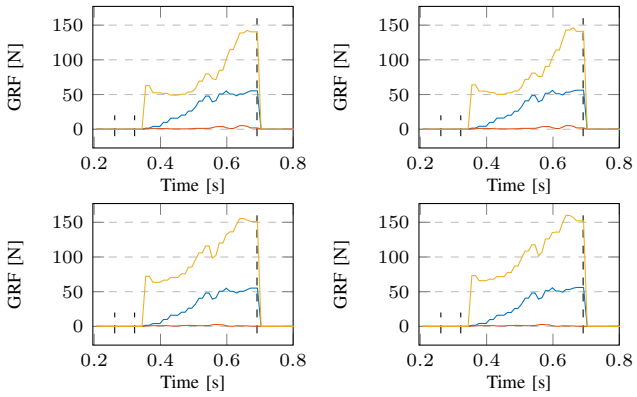

Fig. 8: The command ground reaction force (GRF) computed by jumping controller. the figures from left to right, top to bottom, represent FL, FR, RL, RR respectively. The values of f_x f_y f_z axis are represented by blue, orange, and yellow respectively.

solving time is about 50ms, most of the solve time is around 80ms, and the maximum solve time is under 120ms, allowing for real-time solving after the ball is released.

Released point x_b^0 /m	0.			0.25		0.4	
Desired point x_b^{des} /m	0.5	0.75	1.0	1.0	1.5	1.5	2.0

TABLE I: Ball bumping experiment setting

2) Ball bumping Test: We tested 7 cases (see table I) with the same initial ball release height $z = 1.56m$ and different horizontal positions x. The whole experiment results are included in the supplemental video. The actual ball landing point in all tests is within a circle with a radius of 10 cm centered on the desired landing point.

Fig. 6 shows the snapshot of the case with initial ball position $x = 0$ m,$z = 1.56$ m and desired landing point $x = 2.5$ m, $z = 0$ m, Fig. 7 shows the corresponding pose and twist trajectories given by state estimation module. The ball was released at 0.262 s, and the time instances of a jump start and takeoff were given by solving the nonlinear optimization problem described in section III. The NLP solver spent about 60ms to get the optimized time instances. The quadruped started jumping at 0.342 s and took off at 0.691 s. Finally, the robot reached the desired takeoff state at 0.691 s with the jumping controller described in section IV-A. Fig. 8 shows the foot force computed by MPC in the jumping process. Although the trajectory has not always been tracked very well, the error of takeoff state is acceptable relatively. The state estimation data can not be trusted after the jumping because the foot is off the ground.

VI. Conclusions and Future Work

This paper presented a motion planning and control framework for the ball bumping task that leads the quadruped bump of the ball into the desired landing region. A nonlinear optimization problem is formulated for a dynamically feasible jumping trajectory, which can be solved online. The multi-phase controller is designed for this time-sensitive trajectory, which can reach the desired takeoff state at desired time instance accurately. And the performance has been validated on the Aliengo.

Future work can be devoted to the warm-start interface of the nonlinear optimization problem to get a more reasonable initial guess. A vision detection and tracking system can be developed and run onboard, allowing the dropping of the ball by human hand without the releaser. In addition, the planning for landing motion can be developed for a more steady landing performance.

REFERENCES

[1] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, IEEE, 2018.

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

[3] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 558–564, IEEE, 2016.

[4] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[5] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[6] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. 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)*, pp. 2245–2252, IEEE, 2018.

[7] M. Chignoli, S. Morozov, and S. Kim, "Rapid and reliable trajectory planning involving omnidirectional jumping of quadruped robots," *arXiv preprint arXiv:2111.13648*, 2021.

[8] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3d jumping on quadruped robots," *arXiv preprint arXiv:2110.06764*, 2021.

[9] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *2015 Robotics: Science and Systems Conference, RSS 2015*, MIT Press Journals, 2015.

[10] H.-W. Park, P. M. Wensing, and S. Kim, "Jumping over obstacles with mit cheetah 2," *Robotics and Autonomous Systems*, vol. 136, p. 103703, 2021.

[11] S. Li, H. Chen, W. Zhang, and P. M. Wensing, "Quadruped robot hopping on two legs," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7448–7455, IEEE.

[12] H. Chen, P. M. Wensing, and W. Zhang, "Optimal control of a differentially flat two-dimensional spring-loaded inverted pendulum model," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 307–314, 2019.

[13] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 178–184, 2022.

[14] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 317–328, 2021.

[15] V. Kurtz, H. Li, P. M. Wensing, and H. Lin, "Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 4635–4641, IEEE, 2022.

[16] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," *arXiv preprint arXiv:2208.01160*, 2022.

[17] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2316–2323, IEEE, 2021.

[18] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[19] J. Hodgins and M. H. Raibert, "Biped gymnastics," *Dynamically Stable Legged Locomotion*, vol. 79, 1988.

[20] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3103–3109, IEEE, 2013.

[21] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3821–3828, IEEE, 2018.

[22] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pp. 1–8, IEEE, 2021.

[23] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7309–7315, IEEE, 2021.

[24] F. Yu, R. Batke, J. Dao, J. Hurst, K. Green, and A. Fern, "Dynamic bipedal maneuvers through sim-to-real reinforcement learning," *arXiv preprint arXiv:2207.07835*, 2022.

[25] K. L. Poggensee, A. H. Li, D. Sotsaikich, B. Zhang, P. Kotaru, M. Mueller, and K. Sreenath, "Ball juggling on the bipedal robot cassie," in *2020 European Control Conference (ECC)*, pp. 875–880, IEEE, 2020.

[26] Unitree, "Physical parameters of aliengo model." https://github.com/unitreerobotics/unitree_ros/tree/master/robots/aliengo_description.

[27] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017.

[28] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, IEEE, 2004.

[29] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems." https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[30] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.