Design of a Jumping Control Framework with Heuristic Landing for Bipedal Robots

Jingwen Zhang[1], Junjie Shen[1], Yeting Liu[1], and Dennis Hong[1]

Abstract—Generating dynamic jumping motions on legged robots remains a challenging control problem as the full flight phase and large landing impact are expected. Compared to quadrupedal robots or other multi-legged robots, bipedal robots place higher requirements for the control strategy given a much smaller footprint. To solve this problem, a novel heuristic landing planner is proposed in this paper. With the momentum feedback during the flight phase, landing locations can be updated to minimize the influence of uncertainties from tracking errors or external disturbances when landing. To the best of our knowledge, this is the first approach to take advantage of the flight phase to reduce the impact of the jump landing which is implemented in the actual robot. By integrating it with a modified kino-dynamics motion planner with centroidal momentum and a low-level controller which explores the whole-body dynamics to hierarchically handle multiple tasks, a complete and versatile jumping control framework is designed in this paper. Extensive results of simulation and hardware jumping experiments on a miniature bipedal robot with proprioceptive actuation are provided to demonstrate that the proposed framework is able to achieve human-like efficient and robust jumping tasks, including directional jump, twisting jump, step jump, and somersaults.

I. INTRODUCTION

With significant progress being made in recent decades, it has been proven that legged robots have the potential to go anywhere humans can go and do whatever humans can do. To fulfill this great potential, dynamic jumping is another required capability besides walking and running. However, apart from higher actuation requirements for the torque density and speed, dynamic jumping control yet remains a challenging problem since it involves a long flight phase where the floating base is uncontrollable without contacts and a large landing impact is expected which requires a more robust control strategy.

Early studies on legged robotic jumping are significantly influenced by Raibert's single-leg hopping machine with a heuristic controller [1]. Using a similar controller, Hyon and Mita designed another one-legged hopping robot that had an articulated leg composed of three links [2]. More recently, model-based methods gain more and more attention. In [3], [4], the whole-body dynamic model is used in the low-level whole-body control. With commanding the liftoff velocity, simple jumping can be achieved. To generate versatile jumping with a longer horizon, manually finding the trajectory is nearly impossible due to the high degrees of freedom (DoF)

[1]Jingwen Zhang, Junjie Shen, Yeting Liu, and Dennis Hong are with the Robotics and Mechanisms Laboratory, the Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, CA 90095, USA {zhjwzhang, junjieshen, liu1995, dennishong}@ucla.edu
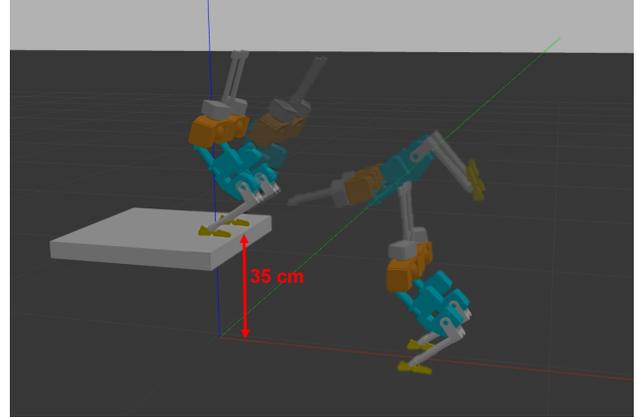
Fig. 1.   Somersault of BRUCE off of a 35 cm platform.

of legged robots, especially bipedal robots. Directly using the whole-body model for planning, the robot can produce more intricate behaviors [5]–[7]. However, due to the complexity of high-dimensional models, these problems sometimes end up being intractable [5], [8], [9]. In [10], [11], the spring-loaded inverted pendulum (SLIP) is accepted as the simplified model to plan running and jumping motions. Despite its success, the point mass is considered with the angular momentum being ignored. Jumping motions involving body rotations, which is fairly normal in animals and humans, is hard to accomplish with it. To mitigate this issue, the single rigid body model (SRBM) is a potential solution. [12] successfully implemented it on a quadruped robot for aerial motion trajectory optimization with the assumption of mass-less legs. Unfortunately, bipedal robots require more actuated joints for each leg and non-point feet for active balancing. Basically, the mass-less leg assumption is easily violated for most bipedal robots. Recently, many approaches [13]–[17] consider a more versatile kino-dynamic planner for bipedal robots based on the centroidal dynamics [18] which is an exact projection of the whole-body dynamics. Centroidal dynamics efficiently introduce the angular momentum into the planner, which benefits arbitrary jumping motion generation.

For landing stabilization, three different methods can be considered: 1) heuristic local damping control [19], [20], 2) unique structure design of feet to handle impact passively [21], 3) actively consider the whole-body dynamics between the robot and the environment. In the last decade, the solution has converged to the last one with a certain class of optimization-based whole-body control [22], which

finds optimal solutions to motor commands online based on real-time feedback via solving a convex quadratic program (QP). For quadrupedal robots, this technique is enough to guarantee the safety [7], [17] if the robot deviates from the optimal trajectory when leaving the ground and is even disturbed in the air since 4 legs provide enough support regions for recovery when landing. For bipedal robots, the tolerance is much lower given a smaller footprint. Besides improving the robustness of the low-level controller, the large landing impact can actually be reduced by adjusting landing locations, which is often ignored in the legged robotic community. The subconscious motion for humans when pushed forward during jumping is moving two legs forward to reduce the influence of the unexpected disturbance on landing. Inspired by this, a heuristic landing planner based on real-time momentum feedback is proposed in this paper. Once the robot leaves the ground, it updates the desired landing locations for feet at high rates using the computed linear and angular momentum as the heuristic feedback.

This paper makes the following contributions:

1) A novel heuristic landing planner is proposed to improve the landing stability via taking advantage of the momentum feedback in the flight phase to minimize the influence of uncertainties from tracking error or external disturbance on landing.

2) A complete and versatile jumping framework for bipedal robots is provided with implementation details as shown in Figure 2. Combining the model-based method with centroidal dynamics and the heuristic approach, a more natural jumping behavior can be achieved including squatting before the liftoff, body active rotating to compensate for the unexpected angular momentum, and lowering the body to buffer landing impact.

3) Demonstration of the proposed framework on a miniature bipedal robot that can achieve a variety of jumping tasks, such as directional jump, twisting jump, step jump, and somersaults.

II. SYSTEM OVERVIEW

A. BRUCE

To promote bipedal robotic research and improve the accessibility to bipedal robot platforms with dynamic capabilities, the next-generation miniature Bipedal Robot Unit with Compliance Enhanced (BRUCE) has been developed in our previous work [23] using proprioceptive actuators. For BRUCE, each leg has 5 degrees of freedom (DoF), which includes a spherical hip joint, a knee joint and an ankle joint. To lower the leg inertia, a cable-driven differential pulley system and a linkage mechanism are applied to the hip and ankle joints, respectively. Being a miniature bipedal robot, BRUCE is designed to be approximately 1/3 of an adult male's height, which is around 660 mm. As a result, link lengths of BRUCE and also other major mechanical parameters are summarized in Table I.

To enable BRUCE to detect when the contact between the foot and the ground is created or broken for state
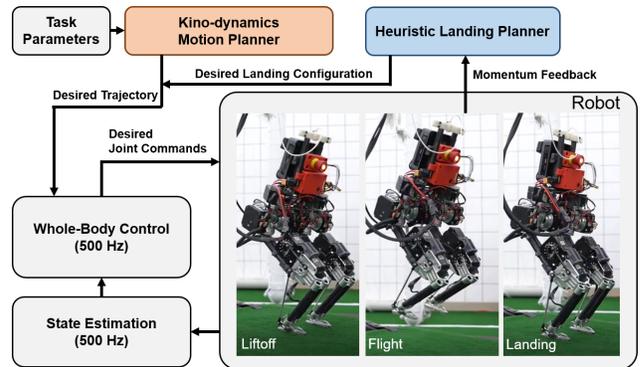

Fig. 2. BRUCE jumping framework. Kino-dynamic motion planner generates desired jumping trajectory based on user-selected task parameters including jumping distance/height, twisting angle, contact sequences, etc. Heuristic landing planner updates desired landing locations with real-time momentum feedback in the air. The low-level whole-body controller feeds desired joint commands to the actual robot with high-level commands.

TABLE I
BRUCE MECHANICAL PARAMETERS

Parameter	Value [Unit]	Parameter	Value [Unit]
Body mass m_b	1687 [g]	Total mass m	5118 [g]
Hip mass m_h	689 [g]	Pelvis length l_p	150 [mm]
Thigh mass m_t	889.5 [g]	Thigh length l_t	175 [mm]
Calf mass m_c	113 [g]	Calf length l_c	169.5 [mm]
Foot mass m_f	24 [g]	Foot length l_f	24 [mm]

estimation purposes, a contact sensor is designed. Tactile switches are embedded into the attached rubber underneath each aluminum foot. To improve the detection performance, one switch is located near the toe with another one near the heel. In this way, contact detection is more robust and sensors are fully protected from the outside environment. Furthermore, all electronics are integrated into the body of BRUCE for fully untethered control. An Intel NUC with Intel Core i5 CPU is used as the onboard PC. The running time is around 20 minutes with a 14.8 V 2200 mAh LiPo battery.

B. Software Architecture

To make BRUCE favorable to dynamic behaviors which require fast response, the overall software framework is developed in a multithreaded environment, which includes a motor communication thread, a state estimation thread combined with robot model computation, and a feedback control thread as illustrated in Figure 2. The control thread is using whole body controller described in Section V, which takes the reference trajectories and robot state feedback and computes desired joint torques at a rate of 500 Hz. The reference trajectories come from high-level planners described in Sections III and IV. Data communication utilizes a custom shared memory library, similar to the setup developed in [21]. All programs are implemented in Python while some parts, including kinematics, dynamics, and state estimation, are precompiled using Numba [24] for acceleration.

Reliable state estimation is crucial to the good performance of legged systems. In the state estimation thread,

a complementarity filter is applied. For body orientation and angular velocity, IMU sensor (ISM330DHCX) readings after proper filters are used. Once the body orientation is determined, IMU accelerometer readings can be integrated to get body velocity and further position. However, these two quantities diverge easily due to sensor noise. Motor encoder readings are introduced as a complement. In specific, the state estimator makes use of the joint encoders for stance leg kinematics to calculate the body position and velocity as a reference. Although Kalman filter is widely used for legged state estimation [25], [26], complementarity filter works as well in practice with a much simpler implementation [27]. Note that this simple approach still comes with some practical issues, e.g., yaw drift, global position inaccuracy.

III. MOTION PLANNING WITH CENTROIDAL MOMENTUM

Bipedal robots have limited supporting regions to recover themselves when landing due to their small footprint. This requires the jumping trajectory must match well with the robot. As a result, the model selection for the planning is crucial. The principle of selection is to make it as simple as possible while keeping versatility to some extent since too complicated models like the full-body dynamics may increase the computational cost significantly and even lead to an intractable problem. In order to achieve a variety of jumping motions, including twisting jumps and somersaults where large body rotations may be required, the point-mass model like the Spring-loaded Linear Inverted Pendulum (SLIP) model is not considered here.

For comparison, the Single-Rigid-Body Model (SRBM) is used at first. By lumping all the link inertia together to get the fixed local inertia of the represented single rigid body, BRUCE fails to accomplish a stable landing due to the unexpected rotation of the body in the air. It turns out that the unexpected angular momentum comes from the thigh rotation when trying to lift off from the ground but it is ignored inside the SRBM. To tackle this issue, we accept centroidal dynamics [18] for motion planning. Since centroidal dynamics consider the full-body mass and inertia distribution, the optimized jumping motion can compensate for the unexpected angular momentum from leg movements.

A. Decision Variables

For the full-body mass and inertia distribution, the joint configuration must be considered. As a result, after including the extra 6 DoFs from the floating body in the joint states, the decision variables are determined for the planner with centroidal momentum as

$$\mathbf{\Gamma} = \{\boldsymbol{q}[k], \boldsymbol{v}[k], \boldsymbol{r}[k], \dot{\boldsymbol{r}}[k], \ddot{\boldsymbol{r}}[k], \boldsymbol{F_j}[k], \boldsymbol{h}[k],$$
$$\text{for all time instances } k\} \tag{1}$$

where $\boldsymbol{q}$ and $\boldsymbol{v}$ denote joint positions and velocities including the floating base. For BRUCE with 10 active joints, $\boldsymbol{q} \in \mathcal{R}^{7+10}$, and $\boldsymbol{v} \in \mathcal{R}^{6+10}$ where $\boldsymbol{q}$ included 7 variables for the floating-base since quaternion is chosen to represent the orientation which requires 4 instead of 3 to avoid the gimbal locking issue. $\boldsymbol{r}, \dot{\boldsymbol{r}}, \ddot{\boldsymbol{r}}$ represent CoM states

including positions, velocities, and accelerations. $\boldsymbol{F_j}$ is the contact forces for the jth contact point. Note that only point contact is considered here. To avoid losing generality, any type of contact can be represented with the point contact. For example, the line contact or square face contact can be divided into 2 or 4-point contacts on the edge. $\boldsymbol{h}$ describes the angular components in the centroidal momentum as defined in [18] which is the exact projection of all the link momentum on the CoM coordinate.

Other works [13], [15] might consider additional decision variables, such as the contact location, the unscheduled contact sequence, and the time step dt. Here, they are fixed for simplification.

B. Constraints

As a kino-dynamics planner, both dynamics and kinematics are considered. The motion of equations for the model with centroidal momentum is written as:

$$m\ddot{\boldsymbol{r}}[k] = \sum_j \boldsymbol{F_j}[k] + m\boldsymbol{g} \tag{2}$$

$$\dot{\boldsymbol{h}}[k] = \sum_j (\boldsymbol{c_j}[k] - \boldsymbol{r_j}[k]) \times \boldsymbol{F_j}[k] \tag{3}$$

$$\boldsymbol{h}[k] = \boldsymbol{A}(\boldsymbol{q}[k])\boldsymbol{v}[k] \tag{4}$$

where $\boldsymbol{c_j}$ denotes the pre-specified contact location for the jth contact point. As shown in Equation (4), the centroidal angular momentum is connected with the joint states with the Centroidal Momentum Matrix (CMM) as defined in [18]. Although being nonlinear, the computation of CMM can be achieved efficiently [28]. Due to the introduction of joint states, the full-body kinematic must be added to ensure kinematic consistency.

$$\boldsymbol{r}[k] = f_{com}(\boldsymbol{q}[k]) \tag{5}$$
$$\boldsymbol{c}[k] = f_{contact}(\boldsymbol{q}[k]) \tag{6}$$

where the function $f(\cdot)$ represents the corresponding forward kinematics for the CoM positions and contact locations. To ensure the consistency of the generated trajectory, integration constraints are formulated as follows:

$$\boldsymbol{q}[k] - \boldsymbol{q}[k-1] = \boldsymbol{v}[k]dt \tag{7}$$
$$\boldsymbol{h}[k] - \boldsymbol{h}[k-1] = \dot{\boldsymbol{h}}[k]dt \tag{8}$$
$$\boldsymbol{r}[k] - \boldsymbol{r}[k-1] = \dot{\boldsymbol{r}}[k]dt \tag{9}$$
$$\dot{\boldsymbol{r}}[k] - \dot{\boldsymbol{r}}[k-1] = \ddot{\boldsymbol{r}}[k]dt \tag{10}$$

When the contact is active for the jth contact point, the contact force is limited inside the friction cone to avoid sliding,

$$\sqrt{(\boldsymbol{F_j})_x^2 + (\boldsymbol{F_j})_y^2} \leq \mu(\boldsymbol{F_j})_z^2, (\boldsymbol{F_j})_z \geq 0 \tag{11}$$

Until here, all constraints describe the general motion of the model with centroidal momentum. To meet the task-specific requirements, decision variables are constrained in a pre-defined boundary set $\mathcal{Q}$ as boundary conditions.

$$\mathbf{\Gamma} \in \mathcal{Q} \tag{12}$$

In the jumping optimization, besides physical boundaries on numerical values of decision variables, users can define the desired jumping parameters in the boundary condition, for example,

$$\boldsymbol{r}[1] = \boldsymbol{r}_0, \boldsymbol{r}[N] = \boldsymbol{r}_0 \tag{13}$$

$$(\boldsymbol{r}[k])_z \leq h_{nom} \quad \text{when in stance} \tag{14}$$

$$h_{nom} \leq (\boldsymbol{r}[k])_z \leq h_{max} \quad \text{when in flight} \tag{15}$$

$$\dot{\boldsymbol{r}}[k_i] = \boldsymbol{v}_{lo} \quad k_i \text{ is the time the robot lifts off} \tag{16}$$

where $\boldsymbol{r}_0$ denotes the COM positions when the robot is standing still, h_{nom} is the nominal height when the robot is on the ground, h_{max} is the maximum COM height, and $\boldsymbol{v}_{lo}$ denotes the liftoff velocity so that users can change its value to reach different jumping height. Similarly, if a twisting jump is desired, constraints on the body orientation and angular momentum can be added to limit the twisting angle and velocity.

C. Complete Formulation

The complete problem can be formulated as a nonlinear programming (NLP) as follows:

$$\min_{\Gamma} \quad \sum_{k=1}^{N} \Bigg(\|\boldsymbol{q}[k] - \boldsymbol{q_{nom}}[k]\|_{\boldsymbol{Q_q}}^2 + \|\boldsymbol{v}[k]\|_{\boldsymbol{Q_v}}^2 + \|\ddot{\boldsymbol{r}}[k]\|^2$$

$$+ \left\|\dot{\boldsymbol{h}}[k]\right\|_{\boldsymbol{Q_h}}^2 + \sum_{j} \|\boldsymbol{F_j}[k]\|_{\boldsymbol{Q_f}}^2 \Bigg) dt \tag{17}$$

s.t., for each knot point $k = 1, \ldots, N$ and

for each contact point $j = 1, \ldots, M$

$$(2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12) \tag{17a}$$

In the cost function, $\|\cdot\|_{\boldsymbol{Q}}^2$ is the abbreviation for the quadratic cost with the weight matrix as $\boldsymbol{Q} \geq 0$. The terminal cost is ignored. Instead, the final state is put as an additional hard constraint in Constraint (12). And new running cost terms on joint positions and velocities are introduced. The difference between an initial guess $\boldsymbol{q_{nom}}$ and optimized joint positions is considered since it does not make any physical meaning to penalize purely large joint positions. $\boldsymbol{q_{nom}}$ can be defined as a fixed nominal position which can be the safest configuration for the robot or a whole trajectory along the time span from users' educated guesses or other simple planners.

IV. HEURISTIC LANDING

Besides the unexpected angular momentum generated from leg movements, another source of uncertainty comes from the tracking error. The low-level controller may not be able to track the optimized jumping trajectory perfectly in practice. Although in [7] and [17], model predictive control (MPC) is combined with the whole-body controller to improve the tracking performance and disturbance rejection, the stable margin is still small for bipedal robots. To tackle this problem, a landing planner is proposed here. The intuition behind this is that the robot is capable of

adjusting its foot freely once the robot leaves the ground. If a human tries to jump in-situ but is pushed forward in the air, the subconscious reaction is to move both legs forward to catch the landing impact. Similarly, the robot can update the landing locations even if the state of the robot is not exactly the same as planned when leaving the ground or being disturbed in the air.

The ideal approach to update the landing locations may be model-based optimization techniques. However, the flight phase may be often too short to apply models considering momentums like the SRBM and centroidal dynamics. Finding the optimal solution to complex models in a real-time manner still remains an open question. Meanwhile, the point-mass models are too abstract to capture the requirements of computing the optimal landing locations. Inspired by the Raibert heuristic for hopping [1] and the capture point [29], the momentum of the robot in the air is used as the heuristic to update the landing locations. For example, if the robot is leaving the ground with a non-zero angular momentum along the y direction, the foot must be moved forward/backward in the air. The landing locations can be updated based on momentum feedback as follows:

$$p^x = p_{nom}^x + W_l^x l^x + W_k^x k^y \tag{18}$$

$$p^y = p_{nom}^y + W_l^y l^y + W_k^y k^x \tag{19}$$

where p^x and p^y denote the updated foot x and y positions while the z position still follows the optimized trajectory in the air, p_{nom}^x and p_{nom}^y are the nominal positions which can be set by users or obtained from the optimized trajectory, l^x and l^y are the linear momentum feedback along x and y directions while k^x and k^y denote the angular momentum feedback, and W is the heuristic gain.

Additionally, inspired by the human behavior to increase the y clearance between two legs when being pushed sideways so as to have a larger supporting region for recovery when landing, an additional term can be added to p^y to adjust the y distance between two legs.

$$\Delta y = W_c |l^y| \tag{20}$$

Lastly, it is easy for the direct heuristic planner to find a landing location out of the leg's reachable region. To avoid this, the heuristic landing planner is designed with a saturation function as follows:

$$p_{des}^x = \begin{cases} p^x & \text{if } |p^x| \leq p_{max}^x \\ p_{max}^x \text{sgn}(p^x) & \text{if } |p^x| > p_{max}^x \end{cases} \tag{21}$$

$$p_{des}^y = \begin{cases} p^y + (-1)^i \Delta y & \text{if } |\hat{p}^y| \leq p_{max}^y \\ p_{max}^y \text{sgn}(p^y + (-1)^i \Delta y) & \text{if } |\hat{p}^y| > p_{max}^y \end{cases} \tag{22}$$

where $\hat{p}^y = p^y + (-1)^i \Delta y$, p_{max}^x and p_{max}^y denote the maximum x and y position that the leg can reach, and $i = 0$ for the left leg while $i = 1$ for the right leg. With this landing planner, the robot can handle both the tracking error when lifting off and the disturbance in the air. More stable landing and push recovery in the air can be found in Section VI.

V. JUMPING CONTROLLER

For the low-level control, the goal is to improve the tracking performance along the optimized jumping trajectory. Although the QP-based whole-body controller is able to provide compliant behaviors and strong robustness, it heavily depends on the high quality of the dynamic model which is often difficult to obtain in practice. Additionally, jumping is a highly dynamic motion that requires significant acceleration and control of the fast leg movement is typically hard. However, inverse kinematics approaches only require the robot kinematic model which is much easier to make accurate. Joint-level PD control can benefit humanoid control due to its modeling error compensation and high updating frequency [30], [31]. Based on that, a combined low-level jump controller for each joint is utilized to improve the tracking performance as follows:

$$\tau^{des} = \tau_{ff} + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}), \quad (23)$$

where K_p/K_d is the P/D feedback gain for each joint. In Equation (23), the last two terms serve as the joint-level feedback terms where q_d and $\dot{q}_d$ are desired joint position and velocity that can be obtained by solving the leg inverse kinematics. The first term is treated as the feedforward term from the operational-space controller. In jumping control, the feedforward term is computed separately for different phases. When the robot is in the air without contact, the robot is following the ballistic trajectory under gravity. The PD feedback terms in Equation (23) are enough to control the foot position as commanded by the heuristic landing planner in Section IV. τ_{ff} is set as 0 in the air accordingly.

For the liftoff and landing phase where the ground contact is active for the robot, a weighted hierarchical whole-body controller is formulated as a quadratic programming (QP) as follows:

$$\min_{\ddot{q}, f_j} \quad \sum_{i=1}^{N_t} \left\| J_i \ddot{q} + \dot{J}_i \dot{q} - \ddot{x}_i^{des} \right\|_{W_i}^2$$
$$+ \sum_{j=1}^{N_c} \| f_j \|_{W_f}^2 + \| \ddot{q} \|_{W_{\ddot{q}}}^2 \quad (24)$$

$$\text{s.t.} \quad H_b \ddot{q} + C_b \dot{q} + G_b - \sum_{j=1}^{N_c} J_{c_j,b}^\top f_j = 0, \quad (24a)$$

$$f_j \in \mathcal{C}_j, \; j = 1, \cdots, N_c, \quad (24b)$$

where J_i is the ith task Jacobian and N_t is the number of tasks. As we can see, the ith operational task is set as a QP cost with priority implicitly being enforced with weight W_i. In addition to the task costs, regularization costs are added to the decision variables $\ddot{q}$ and f_j with small weights $W_{\ddot{q}}$ and W_f respectively to ensure the overall QP cost is strictly positive definite even when the task Jacobians contain singularities, which avoids potential numerical issues. In Problem (24), only the floating-base components of the full-body dynamics are used. After solving the optimal accelerations and forces, the joint torques can be retrieved using the joint

components of the dynamics as follows:

$$\tau_{ff} = H_j \ddot{q}^* + C_j \dot{q} + G_j - \sum_{j=1}^{N_c} J_{c_j,j}^\top f_j^* \quad (25)$$

In this manner, variables for τ can be removed from the decision variables to accelerate the QP solving. But we always assume enough torque that the actuator can provide, i.e., no torque limits. In order to track the optimized jumping trajectory, multiple tasks are prioritized in the following sequence, e.g., the 1st task means the highest priority with the largest task weight W_1).

A. Task 1 - Stance Leg

In general, to ensure the stance leg is nonmoving, besides Constraint (24b) ensures each contact force is bounded and lies within the local friction cone $\mathcal{C}_j$ which is approximated by a square pyramid for linearity, the contact acceleration is also fixed to zero as a hard constraint with the equation $J_{c_j} \ddot{q} + \dot{J}_{c_j} \dot{q} = 0$. However, it was treated as the first task, i.e., a soft constraint with sufficiently large task weight and $\ddot{x}_1^{des} = 0$. This can speed up the QP and give better numerical stability [30].

B. Task 2 - Linear Momentum

In particular, the linear momentum task consists of both feedforward and feedback terms, which are specified in the form of

$$\ddot{x}_2^{des} = a_2^{ref} + K_p(p_2^{ref} - p_2) + K_d(v_2^{ref} - v_2) \quad (26)$$

where a_2^{ref}, v_2^{ref}, p_2^{ref} are the linear acceleration, velocity, and position from the optimized jumping trajectory in Section III, and K_p/K_d is the proportional/derivative (P/D) feedback gain matrix. The linear components of the centroidal momentum matrix (CMM) are used as the task Jacobian.

C. Task 3 - Torso Orientation

Controlling the torso orientation is essential for the angular momentum compensation during jumping. The task acceleration for all three angles are described as:

$$\ddot{x}_3^{des} = \alpha_3^{ref} + K_p \text{Log}(R_3^\top R_3^{ref}) + K_d(\omega_3^{ref} - \omega_3) \quad (27)$$

where α_3^{ref}, ω_3^{ref}, R_3^{ref} are the desired angular acceleration, velocity, orientation, and the logarithm operator Log : $\text{SO}(3) \to \mathbb{R}^3$ converts a rotation matrix to its corresponding axis–angle representation. In practice, α_3^{ref} is set to 0 since it is hard to define angular acceleration while ω_3^{ref} and R_3^{ref} can be obtained from the optimized trajectory.

D. Task 4 - Angular Momentum

With active orientation control in Task 3, the angular momentum task is of low priority yet to regularize the rotation of the body. As a result, the angular momentum task is to damp out excessive angular momentum:

$$\ddot{x}_4^{des} = -K_d k \quad (28)$$

The angular components of the CMM are used as the task Jacobian.

VI. RESULTS

In this section, various jumping tasks of simulation and hardware experiments for BRUCE are conducted to verify the capability of the proposed dynamic jumping framework. The open-source simulator Gazebo [32] with the ODE physics engine is used as the simulation environment. All the trajectories of the following jumping tasks are generated using the jumping planner with centroidal momentum in Section III. With warm start techniques, the solving time varies from around 10 sec to 5 mins depending on the complexity of the task using the SNOPT solver [33] in Drake [34]. To update landing locations in the air, the foot trajectory along x and y directions are updated with the interpolation of the desired foot locations from the landing planner in Section IV while it is still following the optimized trajectory along the z direction. The low-level jumping controller tracks the modified trajectory with the off-the-shelf QP solver OSQP [35] which can achieve a 500 Hz updating frequency, sufficient for real-time feedback control. All of the following experiments can be seen in the accompanied video.

A. Basic Jumping

To verify the capability of the jumping motion planner, the trajectories for different basic jumping tasks including in-situ jump, directional jump, and twisting jump are generated offline. For these tasks, only minor changes are required for the Constraint (12) in Problem (17), e.g., the twisting angle, desired COM final positions, etc. As shown in Figures 2 and 3, BRUCE is able to accomplish a natural jumping and land stably. The active rotation of BRUCE's body during the liftoff phase can be noticed. Since the centroidal dynamics consider leg inertia which is dependent on joint configuration, the planner optimizes the body rotation to compensate for the angular momentum generated from leg movements. Actually, this is exactly what human does when jumping. In order to jump higher, humans would lean their body forwards when squatting and then suddenly rotate the body backward to lift off. With the proposed framework, BRUCE is able to achieve a more natural in-situ jumping motion which involves squatting before the liftoff and lowering the body to relieve the impact after landing.

Meanwhile, the QP-based whole-body controller (24) is able to track the COM position and velocity very well in practice with a large linear momentum task weight as shown in Figure 3. However, due to the small task weight on the torso orientation tracking, even with the optimized body rotation for angular momentum compensation, the angular momentum along the y direction is still not negligible leading to decreasing pitch angle of the body in the air. With the proposed landing planner in the air, the robot is still able to land safely with adjusted landing locations. Note that the body pitch angle is first increasing although with a decreasing trend. This is because the landing planner commands both legs to move forwards. In the ideal case, the body orientation can be kept still. However, the inertia of the body is not big enough compared to the leg inertia for BRUCE. As a result,
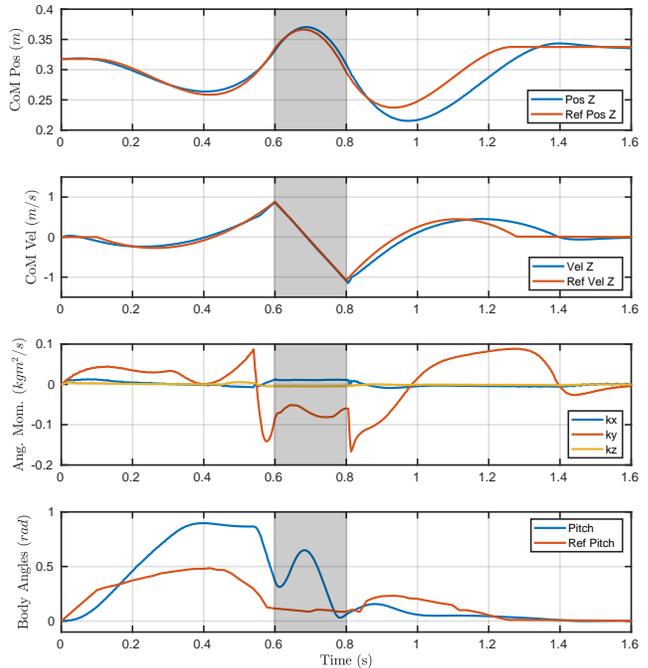

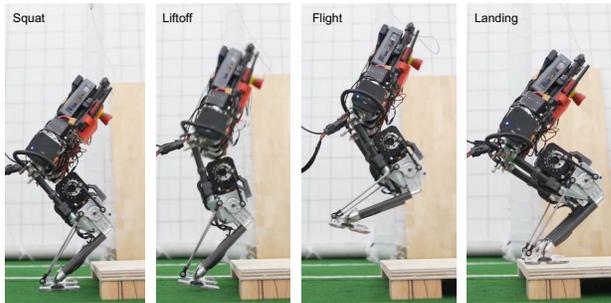Fig. 3. In-situ jumping trajectory for BRUCE. Shaded regions represent the flight phase.

Fig. 4. Screenshots of BRUCE step jumping onto a 5cm step. Motion sequence starts from squatting, then lifting off, to the flight phase, ends with a stable landing.

the body will rotate instead of purely moving legs due to the conservation of angular momentum.

The directional jump and the twisting jump are also conducted in both the simulation environment and the actual robot hardware as shown in the accompanied video. To report here, the maximum jumping height (COM z position change) is around 15 cm (22.7% of its total height). The maximum jumping distance and twisting angle are around 10 cm and 30 deg. The main limitation here is that the desired landing locations cannot be reached with respect to the global frame due to the accompanied body rotation described previously.

B. Step Jumping

By combining the basic jumping motions, BRUCE is also capable of jumping onto a 5cm step and jumping downwards

as shown in the accompanied video with the proposed jumping framework. Note that the step height is chosen to be conservative in this experiment. In Figure 4, BRUCE leans the body forward when squatting and then extends the leg while rotating backward to lift off. As a result, a large portion of the angular momentum generated from the leg movement is compensated while the heuristic landing planner adjusts the landing location in the flight phase to balance out the remaining nonzero angular momentum. When landing, the robot tends to lower the body first to buffer the impact. And eventually, it recovers to the nominal configuration and prepares for the next jump on the step.

C. Push Recovery

To verify the robustness of the landing planner proposed in Section IV, a push recovery test is conducted. In the simulation, BRUCE is commanded to jump in-situ and a constant 70 N pushing force with a duration of 0.01 s is respectively applied to the torso of the robot along x and y directions after 0.03 s in the air. Note that the flight phase is only around 0.2 s. Due to the choice of momentum as the heuristic, the landing planner is able to compute the desired landing positions very fast and leave enough time for the low-level controller to move its legs.

The simulation results are shown in Figure 5. When the robot is pushed along the +x direction, COM x velocity changes immediately and the angular momentum along the y direction increases since the force is applied on the torso leading to a body rotation. As a result, Equation (21) commands a forwarding landing position. Similarly, in Figure 5(b), COM y velocity and the angular momentum along x direction change accordingly when pushed. Since the y-clearance term in Equation (20) is applied to the two legs with a different sign, Equation (22) moves the left foot along +y direction significantly while the right foot is not changed too much. As a result, the distance between the two legs is becoming larger to increase the capability of the robot to stay balanced when landing along the y direction. However, in both cases, although the foot is already in the commanded position with respect to the body frame, the tracking of the foot position with respect to the global frame is not perfect since the rotation of BRUCE's body is inevitable when commanding to move legs in the air due to the comparable inertia of the body and legs.

D. Somersault

To further explore the dynamic capability of BRUCE and the potential of the proposed jumping framework, a somersault is achieved in the simulation using the proposed approach. Due to the quaternion representation of the orientation in Problem (17), the planner is able to deal with the task that requires the body of BRUCE to rotate 360 deg. In Figure 1, BRUCE executes a somersault to jump off of a 35 cm platform. The actual actuator of BRUCE is not powerful enough to support the somersault in-situ. After relaxing the torque limits in the simulation, BRUCE is able to perform a somersault in-situ as shown in Figure 6.
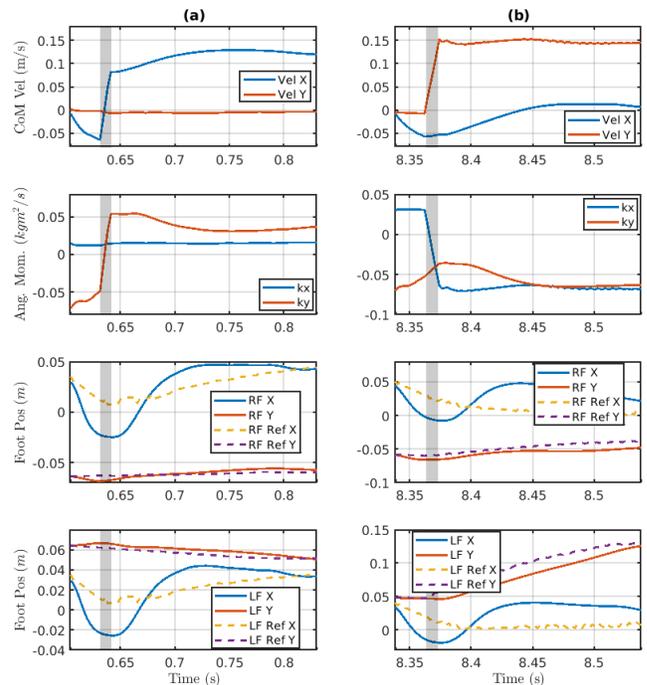

Fig. 5. Simulation results of push recovery. The shaded region represents the 0.01s duration of the push while the graph only shows the trajectory of the flight phase. (a) Push along +x direction. (b) Push along +y direction.
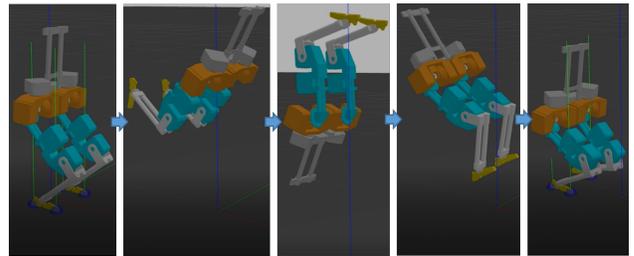

Fig. 6. Somersault of BRUCE in-situ with relaxed torque limits.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, a complete dynamic jumping framework for bipedal robots with a novel heuristic landing planner is presented. Specifically, the high-level jumping planner with centroidal momentum is solving a NLP offline to get the local-optimal jumping trajectory, which is a series of motions similar to human jumping including squatting before liftoff, body lowering after landing, and body rotating to compensate for the angular momentum. To deal with the tracking error when lifting off and possible disturbances in the air for safer landing, the heuristic landing planner updates the landing positions in a real-time manner during the flight phase based on the momentum feedback. To the best of our knowledge, this is the first approach to take advantage of the flight phase to reduce the impact of the jump landing which is implemented in the actual robot. The low-level whole-body controller is finding the required joint torques to best accomplish the operational-space tasks

by solving a small-scale QP, which guarantees the global optimality and ensures a 500 Hz updating frequency. With this framework, a miniature bipedal robot, BRUCE is capable of directional jumps, twisting jumps, jumps with push in the air, and somersaults, which demonstrates the versatility and robustness of the framework.

In the future, a better performance in the actual hardware can expected if the robot, BRUCE can be upgraded with a more optimized mass distribution. The more inertia lumped into the body, the more benefits the proposed framework can get to control the actual hardware. Due to the long solving time of the jumping planner, an offline motion library like [36] is under exploration as well for online dynamic behavior execution.

REFERENCES

[1] M. H. Raibert, "Hopping in legged systems—modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, 1984.

[2] S.-H. Hyon and T. Mita, "Development of a biologically inspired hopping robot-" kenken"," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 4, pp. 3984–3991, IEEE, 2002.

[3] M. De Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM transactions on graphics (TOG)*, vol. 29, no. 4, pp. 1–10, 2010.

[4] 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.

[5] J. Shen, Y. Liu, X. Zhang, and D. Hong, "Optimized jumping of an articulated robotic leg," in *2020 17th International Conference on Ubiquitous Robots (UR)*, pp. 205–212, 2020.

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

[7] C. Nguyen, L. Bao, and Q. Nguyen, "Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control," *arXiv preprint arXiv:2204.01147*, 2022.

[8] C. Mastalli *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2536–2542, 2020.

[9] J. Zhang, X. Lin, and D. W. Hong, "Transition motion planning for multi-limbed vertical climbing robots using complementarity constraints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2033–2039, 2021.

[10] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5134–5140, IEEE, 2013.

[11] P. M. Wensing and D. E. Orin, "Development of high-span running long jumps for humanoids," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 222–227, IEEE, 2014.

[12] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7693–7699, IEEE, 2021.

[13] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, IEEE, 2014.

[14] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 842–849, IEEE, 2016.

[15] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5776–5782, IEEE, 2018.

[16] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[17] 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.

[18] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2, pp. 161–176, 2013.

[19] Y.-D. Kim, B.-J. Lee, J.-H. Ryu, and J.-H. Kim, "Landing force control for humanoid robot by time-domain passivity approach," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1294–1301, 2007.

[20] J. Jo, G. Park, and Y. Oh, "Robust landing stabilization of humanoid robot on uneven terrain via admittance control and heel strike motion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2994–3000, IEEE, 2021.

[21] J. Yu, J. Hooks, X. Zhang, M. Sung Ahn, and D. Hong, "A proprioceptive, force-controlled, non-anthropomorphic biped for dynamic locomotion," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 1–9, 2018.

[22] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *arXiv preprint arXiv:2211.11644*, 2022.

[23] Y. Liu, J. Shen, J. Zhang, X. Zhang, T. Zhu, and D. Hong, "Design and control of a miniature bipedal robot with proprioceptive actuation for dynamic behaviors," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8547–8553, IEEE, 2022.

[24] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A LLVM-based python JIT compiler," in *Workshop on the LLVM Compiler Infrastructure in HPC*, 2015.

[25] M. Bloesch *et al.*, *State Estimation for Legged Robots: Consistent Fusion of Leg Kinematics and IMU*, pp. 17–24. MIT Press, 2013.

[26] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6058–6064, 2013.

[27] Y. Ding *et al.*, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[28] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1550039, 2016.

[29] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*, pp. 200–207, IEEE, 2006.

[30] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the Atlas robot," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 120–127, 2014.

[31] 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.

[32] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, 2004.

[33] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[34] T. Russ and the Drake Development Team, "Drake: Model-based design and verification for robotics." https://drake.mit.edu, 2019.

[35] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[36] M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros, and M. Hutter, "Offline motion libraries and online mpc for advanced mobility skills," *The International Journal of Robotics Research*, vol. 41, no. 9-10, pp. 903–924, 2022.