Highly Dynamic Jumping with a Knee-less Bipedal Robot SLIDER

Ke Wang[1], Songyan Xin[2], Luyao Zhang[3], Digby Chappell[1], Sethu Vijayakumar[2] and Petar Kormushev[1]

***Abstract*— We propose a planning and control framework for our robot SLIDER to generate highly dynamic jumping motion. SLIDER is a lightweight bipedal walking robot with a novel knee-less leg design. Its non-anthropomorphic straight-legged design reduces the weight of the legs significantly. As a result, most of the weight of the robot is centered within the main body. Such concentrated mass distribution brings benefits for both control and planning. On the control side, a lower inertia leg means that the robot can perform more agile motion which requires higher acceleration capability. On the planning side, the actual robot can be better approximated by a reduced order template model. This greatly reduces the computational cost for the motion planning without introducing much model discrepancy compared with the physical robot. In this paper, we take advantage of both the control and the planning side to realise highly dynamic jumping motions including jumping onto a box with 35 cm height and twisting jump with a yaw orientation change of 90 degrees. The approach is demonstrated in simulation and we plan to apply it to the real robot in the near future.**

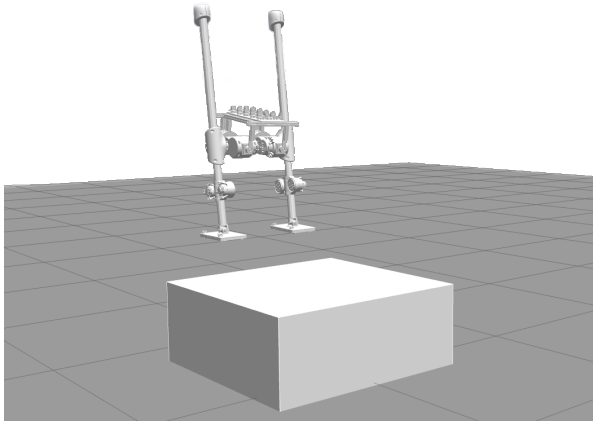


Fig. 1: The bipedal robot SLIDER is jumping onto a box in simulation.

I. INTRODUCTION

Early studies on highly dynamic motion such as jumping and running on legged robots are largely influenced by the heuristic control implemented on Raibert's hoppers [1]. The hopper is able to achieve the dynamic behaviors through simple composition of a set of simple controllers that control hopping height, speed, and posture separately. This is possible due to their prismatic leg design which differs from most humanoid robots with human-like morphology. In fact, the prismatic leg design not only simplifies the control problem but also the motion planning problem which will be demonstrated throughout this paper. Despite its success, heuristic control is quit limited since it needs large amount of work on parameter tuning. More recently, model based approaches are becoming more and more popular. The spring loaded inverted pendulum (SLIP) model is a well recognized template model for running and jumping [2]. An approximated SLIP model has been used to represent the vertical motion, robust running and jumping are planned with model predictive control [3]. 3D-SLIP model has been used to generate high speed running motion [4] and long jump [5]. The trajectories for these motions are all generated with offline nonlinear optimization and then tested in simulation with a full body model robot. However, the SLIP model only considers the point mass dynamics and the angular momentum is ignored, it is difficult to consider motions involving body rotation, such as a twisting jump.

Comparing to humanoid robots, quadruped robots are often built with less degree of freedoms (DoFs) legs and point feet. This leads to light weight legs which can be safely ignored during the motion planning phase or even the control phase without introducing too much modeling error. Dynamic gaits including trot, pronk, bound and pace has been demonstrated on Mini Cheetah through online convex model-predictive control [6], [7]. The convex formulation is made possible due to the small angle assumption for roll and pitch angle of the single rigid body model and predefined footstep locations. In the later work of [8], backflip motion has been realized, however the trajectories is planned for 2D and in an offline fashion. A more versatile gait and trajectory optimization framework is proposed in [9], the single rigid body model has been used to plan the motion. The results show that the simple model is rich enough to express a wide range of motion possibilities. In this work, we adopt the same model and use it to plan highly dynamic motion for our biped robot SLIDER [13] (Fig. 1).

On the other end, trajectory optimization based on full order model has demonstrated great potential to ease the whole motion planning process. Highly dynamic maneuvers such as jumping and flips have been generated with the library Crocoddyl [11] for both biped and quadruped robots, but such motions have not been demonstrated on real robot yet. The demonstrated motion is limited to walking on a HRP-2 humanoid with 20 Hz re-planning refresh rate [12]. The underlying technique used to solve the motion planning

[1]Authors are with the Robot Intelligence Lab, Dyson School of Design Engineering, Imperial College London, UK. Email: k.wang17@imperial.ac.uk

[2]Authors are with the School of Informatics, University of Edinburgh, Edinburgh, UK.

[3]Author is with the Department of Electrical & Electronic Engineering, Imperial College London, UK.

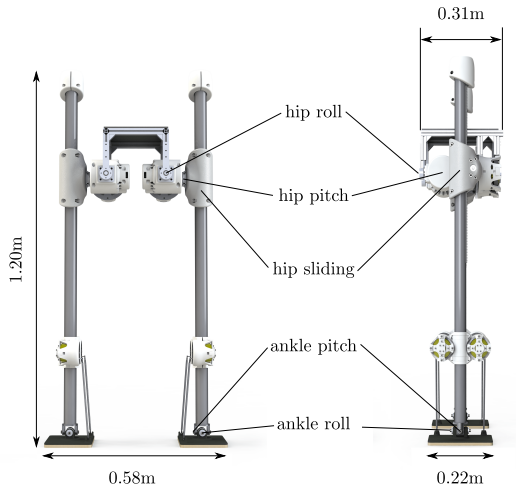


Fig. 2: The dimension and joint configuration of SLIDER.

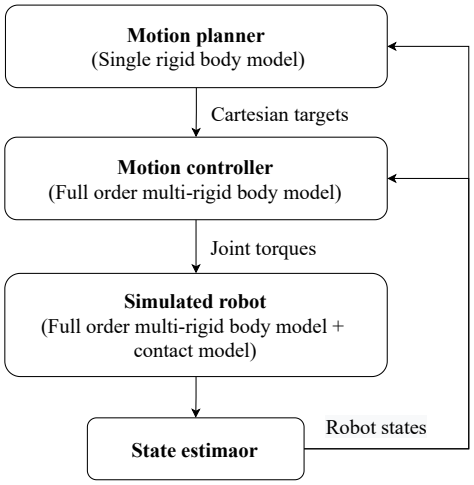


Fig. 3: The system control hierarchy.

problem is differential dynamic programming (DDP) which is an unconstrained optimization approach and therefore is difficult to enforce the system dynamics as hard constraints. In this paper, we use constrained optimization in which the system dynamics can be strictly enforced.

In this paper, we find that the single rigid body model is a good trade off between the model complexity and the motion expression richness. The demonstrated jumping motions verified this. The contribution of this paper is a tightly connected robotic system that combines the software and hardware to achieve highly dynamic agile motions. Here, we would like to highlight the paper:

- The lightweight legs of the SLIDER robot can swing faster so that it allows that robot to perform highly agile motions.
- The SLIDER robot with concentrated mass distribution can be well approximated by a single rigid body model which greatly reduce the model complexity of the motion planning problem.
- With the single rigid body model, the body orientation and angular momentum is well defined, both can be explicitly optimized. It is verified by the generated twisting jump.

II. SYSTEM OVERVIEW

A. The Robot SLIDER

SLIDER is a knee-less bipedal robot designed by the Robot Intelligence Lab at Imperial College London [13]. As shown in Fig. 2, SLIDER is 1.2 m tall and has 10 Degrees of Freedom (DoF), namely hip pitch, hip roll, hip slide, ankle roll and ankle pitch on each leg. The robot is lightweight (14 kg in total) and most of its weight is concentrated in the pelvis because the legs are make of carbon fiber reinforced polymer. The prismatic knee joint design is an unique feature of this robot that differentiate it from many other robots with anthropomorphic design. The sliding joint has a relatively large range of motion. Due to its sliding knee and lightweight leg design, SLIDER can be well approximated with a simple model which can greatly simplify the planning and control problem. The overall light weight and large range of motion leg make the robot suitable for agile locomotion.

B. The Control Hierarchy

The system has a hierarchical structure which mainly includes the motion planner and motion controller as shown in Fig. 3. The high-level user inputs to the motion planner are the initial and final states, contact positions for non-flight phases. The motion planner outputs the trajectories for the CoM, the body as well as the feet. The motion controller will find out all joint torques for the robot to track these Cartesian targets while considering all physical constraints. The state estimator fuses the inertial measurement unit (IMU) and joint encoder readings to estimate the robot state and feed it back to the planner and controller for next loop update.

III. TRAJECTORY OPTIMIZATION

This section describes the mathematical formulation of the single rigid body model by which we are using in offline trajectory optimization and how we formulate trajectory optimization into a Nonlinear Programming (NLP) problem.

A. Single Rigid Body Model

Due to the lightweight leg design of SLIDER, the robot can be abstracted as a single rigid body model as illustrated in Fig. 4. The model captures the translational dynamics as well as the rotational dynamics which is essential for generating twisting jump or backflip.

The Newton-Euler dynamics of the single rigid body with multiple contacts with the environment can be written as:

$$\begin{bmatrix} \dot{\boldsymbol{H}} = \sum_{i=1}^{n} \boldsymbol{f}_i + m\boldsymbol{g} \\ \dot{\boldsymbol{L}} = \sum_{i=1}^{n} (\boldsymbol{p}_i - \boldsymbol{r}) \times \boldsymbol{f}_i \end{bmatrix} \tag{1}$$

where $\boldsymbol{H}$ and $\boldsymbol{L}$ are the linear and angular momentum of the single rigid body, $\boldsymbol{f}_i$ is the ith ground reaction force acting on $\boldsymbol{p}_i$, n forces are assumed. $\boldsymbol{r}$ is the position of the CoM, m and $\boldsymbol{g}$ are the mass of the model and the gravity vector.

The linear momentum is directly related to the translational velocity of the CoM:

$$\dot{\boldsymbol{r}} = \boldsymbol{H}/m \tag{2}$$

For a single rigid body, its angular momentum $\boldsymbol{L}$ can be expressed as:

$$\boldsymbol{L} = \boldsymbol{I}_W \boldsymbol{\omega} \tag{3}$$

where $\boldsymbol{I}_W$ represents the inertia tensor of the body in the world frame, $\boldsymbol{\omega}$ is the angular velocity of the body. In [6] the Euler angles is used to represent the orientation, however this approach suffers from the problem of gimbal lock, so we prefer quaternion representation for the orientation. The dynamics of the quaternion $\boldsymbol{q}_B$ can be expressed as:

$$\dot{\boldsymbol{q}}_B = \frac{1}{2}\boldsymbol{q}_B \circ \boldsymbol{\omega} = \boldsymbol{Q}(\boldsymbol{q}_B)\boldsymbol{\omega} \tag{4}$$

where $\circ$ represents the quaternion product and $\boldsymbol{Q}(\boldsymbol{q}_B) \in \mathbb{R}^{4\times 3}$ is the corresponding matrix representation. If we denote $\boldsymbol{I}_W$ as the body inertia in the world frame, it can be calculated from its local body inertia tensor $\boldsymbol{I}_B$ and body orientation $\boldsymbol{R}_B$ expressed in the world frame:

$$\boldsymbol{I}_W = \boldsymbol{R}_B \boldsymbol{I}_B \boldsymbol{R}_B^T = \boldsymbol{R}(\boldsymbol{q}_B)\boldsymbol{I}_B \boldsymbol{R}^T(\boldsymbol{q}_B) \tag{5}$$

where $\boldsymbol{R}(\boldsymbol{q}_B)$ is a transformation from quaternion to rotation matrix as given in the appendix VII. By associating equation (3), (4), (5), we can get the dynamics of orientation represented by quaternion:

$$\dot{\boldsymbol{q}}_B = \frac{1}{2}\boldsymbol{Q}(\boldsymbol{q}_B)\boldsymbol{I}_W^{-1}(\boldsymbol{q}_B)\boldsymbol{L} \tag{6}$$

So the complete dynamics of the single rigid body with orientation can be summarized as:

$$\begin{bmatrix} \dot{\boldsymbol{r}} = \boldsymbol{H}/m \\ \dot{\boldsymbol{q}}_B = \frac{1}{2}\boldsymbol{Q}(\boldsymbol{q}_B)\boldsymbol{I}_W^{-1}(\boldsymbol{q}_B)\boldsymbol{L} \\ \dot{\boldsymbol{H}} = \sum_{i=1}^{n} \boldsymbol{f}_i + m\boldsymbol{g} \\ \dot{\boldsymbol{L}} = \sum_{i=1}^{n} (\boldsymbol{p}_i - r) \times \boldsymbol{f}_i \end{bmatrix} \tag{7}$$

B. Problem Formulation

The trajectory optimization [18] is a powerful tool to find optimal trajectories for complex tasks that contains nonlinear dynamics and involves multiple phases. In this paper, we employ the *multiple shooting method* to transcript our trajectory optimization problem into a NLP formulation. In the formulation, multiple phases are considered since we are focusing on versatile dynamic jumps that contains multiple pre-defined contact phases: *pre-jump phase*, *flight phase* and *post-landing phase*.

The system has been discretised into N segments and therefore $N+1$ knots exists. Since m phases are assumed, each phase gets N/m segments. Because timing is important for generating highly dynamic motions we don't fix the time for each phase ΔT. However we keep the time interval dt between knots inside each phase to be the same to reduce

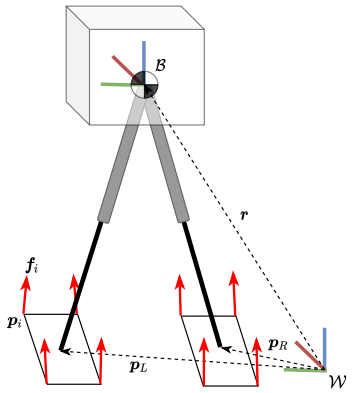


Fig. 4: The SLIDER robot can be well approximated by a single rigid body plus two rectangle feet. The ground reaction forces are located at four corners of the foot with arbitrary directions. $\mathcal{W}$ and $\mathcal{B}$ stand for the world frame and the body frame.

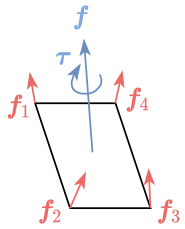


Fig. 5: The four contact forces $[\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4]$ on the foot is equivalent to a contact wrench $[\boldsymbol{f}, \boldsymbol{\tau}]$. With these corner forces, both the linear and angular momentum of the robot can be changed.

the computation cost. For knot k, the state $\boldsymbol{x}[k]$ and control $\boldsymbol{u}[k]$ are defined as:

$$\begin{aligned} \boldsymbol{x}[k] &= [\boldsymbol{r}[k], \boldsymbol{q}_B[k], \boldsymbol{H}[k], \boldsymbol{L}[k]] \\ \boldsymbol{u}[k] &= [\boldsymbol{f}_L[k], \boldsymbol{f}_R[k], \boldsymbol{p}_L[k], \boldsymbol{p}_R[k]] \\ \boldsymbol{\xi}[k] &= [\boldsymbol{x}[k], \boldsymbol{u}[k], dt[k]] \end{aligned}$$

where $\boldsymbol{f}_L[k], \boldsymbol{f}_R[k]$ are the ground reaction forces acting on the left foot and right foot, they collects four corresponding corner forces: $\boldsymbol{f}_L[k] = [\boldsymbol{f}_{L_1}[k], \boldsymbol{f}_{L_2}[k], \boldsymbol{f}_{L_3}[k], \boldsymbol{f}_{L_4}[k]]$, $\boldsymbol{f}_R[k] = [\boldsymbol{f}_{R_1}[k], \boldsymbol{f}_{R_2}[k], \boldsymbol{f}_{R_3}[k], \boldsymbol{f}_{R_4}[k]]$ as shown in Fig. 5. $\boldsymbol{p}_L[k]$ and $\boldsymbol{p}_R[k]$ are the center position of the left foot and right foot respectively, $dt[k]$ is the time interval from knot k to $k+1$.

Given the open parameters $\boldsymbol{\xi}[k]$, the complete trajectory optimization problem can be formulated as follows:

$$\min_{\boldsymbol{\xi}} \sum_{k=0}^{N} l(\boldsymbol{\xi}[k]) + \phi(\boldsymbol{\xi}[N]) \quad \text{(cost function)}$$

$$\text{s.t.} \quad \boldsymbol{x}[k+1] = f(\boldsymbol{x}[k], \boldsymbol{u}[k]) \quad \text{(dynamics)}$$

$$\boldsymbol{x}[0] = \boldsymbol{x}_{ini} \quad \text{(initial states)}$$

$$\boldsymbol{x}[N] = \boldsymbol{x}_{fin} \quad \text{(final states)}$$

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_L[k]\| \leq l_{max} \quad \text{(kinematic limits)}$$

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_R[k]\| \leq l_{max} \quad \text{(kinematic limits)}$$

$$t_{min} \leq dt[k] \leq t_{max} \quad \text{(time limits)}$$

$$\|\boldsymbol{q}_B[k]\| = 1 \quad \text{(quaternion norm)}$$

if foot in contact:

$$f_{i_j}^z \geq 0 \quad (i \in \{L, R\}, j \in \{1, 2, 3, 4\}) \quad \text{(pushing force)}$$

$$\boldsymbol{P}\boldsymbol{f}_{i_j} \leq \boldsymbol{0} \quad (i \in \{L, R\}, j \in \{1, 2, 3, 4\}) \quad \text{(friction cone)}$$

$$\boldsymbol{p}_L[k] = \boldsymbol{p}_L^* \quad \text{(given contacts)}$$

$$\boldsymbol{p}_R[k] = \boldsymbol{p}_R^* \quad \text{(given contacts)}$$

if foot in the air:

$$\boldsymbol{f}_{i_j}^z = \boldsymbol{0} \quad (i \in \{L, R\}, j \in \{1, 2, 3, 4\}) \quad \text{(no contact force)}$$

C. Cost Function

The items inside the cost function can be categorized into 3 groups: *control inputs smoothness, energy consumption* and *final state target*.

Control Inputs Smoothness: To generate a smooth motion the difference of ground reaction forces and foot movements between adjacent nodes are minimized:

$$\sum_{k=0}^{N} \left(\dot{\boldsymbol{p}}_L^2[k] + \dot{\boldsymbol{p}}_R^2[k] + \sum_{i}^{\{L,R\}} \sum_{j}^{\{1,2,3,4\}} \dot{\boldsymbol{f}}_{i_j}^2[k] \right) \tag{8}$$

Energy Consumption: We minimize squared momentum to achieve minimal energy consumption:

$$\sum_{k=0}^{N} (\boldsymbol{H}^2[k] + \boldsymbol{L}^2[k]) \tag{9}$$

Final State Target: Though final state targets are formulated as constraints, we find that putting final orientation target into cost function can speed up the solving process. This is especially the case in tasks requiring large change of orientation, like twisting jump. So we put final orientation targets into the cost function term.

$$(\boldsymbol{q}_B[N] - \boldsymbol{q}_B^{fin})^2 + (\dot{\boldsymbol{q}}_B[N] - \dot{\boldsymbol{q}}_B^{fin})^2 \tag{10}$$

D. Kinematics Constraint

Though the slide joint in SLIDER robot has a large range of motion, we still have to prevent the pelvis from going too high to hit the cap or too low to hit the ankle, see Fig 2. Therefore we constrain the distance between CoM and both feet to be within $[l_{min}, l_{max}]$,

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_L[k]\| \leq l_{max} \tag{11}$$

$$l_{min} \leq \|\boldsymbol{r}[k] - \boldsymbol{p}_R[k]\| \leq l_{max} \tag{12}$$

here we set l_{min} = 0.35m and l_{max} = 0.75m.

E. Contact Constraint

We have pre-defined the contact sequence as *pre-jump phase*, *flight phase* and *post-landing phase*. In *pre-jump phase* and *post-landing phase* both feet are in contact with the ground, so all the ground reaction forces would be 0 or pushing against the ground, namely:

$$f_{i_j}^z \geq 0 \quad (i \in \{L, R\}, j \in \{1, 2, 3, 4\}) \tag{13}$$

In jumping tasks, we also pre-define the start and target position of the foot, namely:

$$\boldsymbol{p}_L[k] = \boldsymbol{p}_L^{ini}, \quad \boldsymbol{p}_R[k] = \boldsymbol{p}_R^{ini} \quad \text{(pre-jump phase)}$$

$$\boldsymbol{p}_L[k] = \boldsymbol{p}_L^{fin}, \quad \boldsymbol{p}_R[k] = \boldsymbol{p}_R^{fin} \quad \text{(post-landing phase)}$$

If all feet is in the air, which is the case of *flight phase*, we enforce all ground reaction force to be exactly 0:

$$\boldsymbol{f}_{i_j} = \boldsymbol{0} \quad (i \in \{L, R\}, j \in \{1, 2, 3, 4\}) \tag{14}$$

F. Friction Cone Constraint

When the foot is in contact with the ground, namely in *pre-jump phase* and *post-landing phase*, we ensure no slippage of foot's contact points. The tangential forces are constrained to remain inside the Coulomb friction cone defined by the terrain friction coefficient μ. We approximate the friction cone by the friction pyramid, which is a common approach to make the constraints linear and speeds up the computation. The constraint is given by:

$$-\mu f_{i_j}^z \leq f_{i_j}^x \leq \mu f_{i_j}^z \tag{15}$$

$$-\mu f_{i_j}^z \leq f_{i_j}^y \leq \mu f_{i_j}^z \tag{16}$$

where $f_{i_j}^x$, $f_{i_j}^y$, $f_{i_j}^z$ are components of the ground reaction force $\boldsymbol{f}_{i_j} (i \in \{L, R\}, j \in \{1, 2, 3, 4\})$.

IV. WHOLE-BODY CONTROL

The whole-body controller takes responsibility of computing the joint torques to achieve the desired motions while respecting a set of constraints. In the paper, the tasks of interest are the CoM position, the pelvis orientation, the angular momentum of the robot, the foot positions and orientations. Each task is comprised of a desired acceleration as a feed-forward term and a state feedback term to stabilize the trajectory. Generally, the task for the linear motion can be expressed as:

$$\boldsymbol{J}_{\mathrm{T}} \ddot{\boldsymbol{q}} = \ddot{\boldsymbol{x}}^{\mathrm{cmd}} - \dot{\boldsymbol{J}}_{\mathrm{T}} \dot{\boldsymbol{q}},$$

$$\ddot{\boldsymbol{x}}^{\mathrm{cmd}} = \ddot{\boldsymbol{x}}^{\mathrm{des}} + \boldsymbol{K}_{\mathrm{P}}^{\mathrm{pos}} (\boldsymbol{x}^{\mathrm{des}} - \boldsymbol{x}) + \boldsymbol{K}_{\mathrm{D}}^{\mathrm{pos}} (\dot{\boldsymbol{x}}^{\mathrm{des}} - \dot{\boldsymbol{x}}),$$

where $\boldsymbol{J}_{\mathrm{T}}$ is the translational Jacobian for the task, $\boldsymbol{x}$ is the actual position of the link, and the superscript des indicates the desired motion.

For the task of angular motion, the command can be formulated as:

$$\boldsymbol{J}_{\mathrm{R}} \ddot{\boldsymbol{q}} = \dot{\boldsymbol{\omega}}^{\mathrm{cmd}} - \dot{\boldsymbol{J}}_{\mathrm{R}} \dot{\boldsymbol{q}},$$

$$\dot{\boldsymbol{\omega}}^{\mathrm{cmd}} = \dot{\boldsymbol{\omega}}^{\mathrm{des}} + \boldsymbol{K}_{\mathrm{P}}^{\mathrm{ang}} (\mathrm{AngleAxis}(\boldsymbol{R}^{\mathrm{des}} \boldsymbol{R}^T)) + \boldsymbol{K}_{\mathrm{D}}^{\mathrm{ang}} (\boldsymbol{\omega}^{\mathrm{des}} - \boldsymbol{\omega}),$$

where $\boldsymbol{J}_{\mathrm{R}}$ is the rotational Jacobian for the task, $\boldsymbol{R}$ and $\boldsymbol{R}^{\mathrm{des}}$ denote the actual and desired orientation of the pelvis

link respectively, $\mathrm{AngleAxis}()$ maps a rotation matrix to the corresponding axis-angle representation, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the link.

For the CoM task and angular momentum task, the centroidal momentum matrix [19] is used as the task jacobian. For balancing or walking, angular momentum task are often defined as a damping task that damps out excess angular momentum. However for highly dynamic motion such as twisting jump, angular momentum varies a lot during the process and it plays a vital role to achieve the jumping motion. In this case, the reference angular momentum is needed and it comes from our motion planner. Since the single rigid body model is used in the motion planner, its orientation and associated angular momentum are both well defined.

A. QP formulation

Inspired by [17], the full dynamics can be decomposed into the underactuated part and actuated part:

$$\begin{bmatrix} \boldsymbol{M}_f \\ \boldsymbol{M}_a \end{bmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{H}_f \\ \boldsymbol{H}_a \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{S}_a \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} \boldsymbol{J}_f^{\mathrm{T}} \\ \boldsymbol{J}_a^{\mathrm{T}} \end{bmatrix} \boldsymbol{f},$$

where $\boldsymbol{M}$, $\boldsymbol{H}$, $\boldsymbol{S}_a$, $\boldsymbol{\tau}$, $\boldsymbol{J}$ and $\boldsymbol{f}$ are the mass matrix, Coriolis force matrix and gravitation force vector, the actuator selection matrix, joint torques vector, the stacked contact Jocabian and reaction force vector. The subscript, f and a, indicates the floating part and actuated part respectively. The weighted sum formulation is applied, in which one QP problem is solved at each control loop. The formulation of the QP problem can be written as

$$\begin{aligned} \min_{\ddot{\boldsymbol{q}},\, \boldsymbol{f}} \quad & \|\boldsymbol{A}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}\dot{\boldsymbol{q}} - \boldsymbol{B}^{\mathrm{cmd}}\|^2_{\boldsymbol{W}} && (17) \\ \text{s.t.} \quad & \boldsymbol{M}_f\ddot{\boldsymbol{q}} - \boldsymbol{J}_f^{\mathrm{T}}\boldsymbol{f} = -\boldsymbol{H}_f \quad \text{(floating base dynamics)} \\ & \boldsymbol{P}\boldsymbol{f} \leq \boldsymbol{0} \quad \text{(friction cone)} \\ & \boldsymbol{S}_a^{-1}(\boldsymbol{M}_a\ddot{\boldsymbol{q}} + \boldsymbol{H}_a - \boldsymbol{J}_a^{\mathrm{T}}\boldsymbol{f}) \in [\boldsymbol{\tau}_{min},\, \boldsymbol{\tau}_{max}] \\ & \quad \text{(input limits)} \end{aligned}$$

where $\boldsymbol{A}$ is a stack of the Jacobian matrices for the tasks of interest, $\boldsymbol{B}^{\mathrm{cmd}}$ is a stack of the commanded accelerations and $\boldsymbol{W}$ is the weighting matrix, $\boldsymbol{P}$ denotes the linearized friction cone. Similar to [10][15] , the unilateral contact constraint is treated as a soft constraint by simply assigning a large weight on the desired zero acceleration. It is reported in [16] that this gives a better stability.

The output torque commands $\boldsymbol{\tau}$ at each control iteration is computed by

$$\boldsymbol{\tau} = \boldsymbol{S}_a^{-1}(\boldsymbol{M}_a\ddot{\boldsymbol{q}} + \boldsymbol{H}_a - \boldsymbol{J}_a^{\mathrm{T}}\mathbf{f}) \qquad (18)$$

V. SIMULATION RESULTS

This section discusses the implementation and results of several versatile jumping motion generated by our framework. We verify our planning and control pipeline through several jumping motions such as jumping onto a box with 0.4 m height and twisting jump. All the motions are showed in details in the accompanying video.

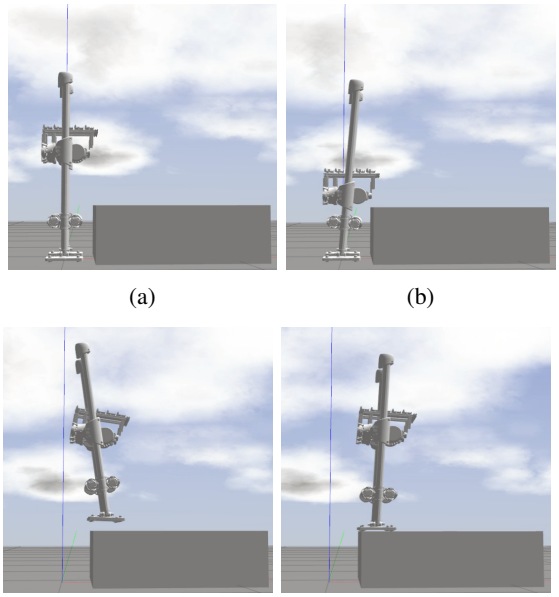


Fig. 6: Snapshots of SLIDER performing a forward jump onto a box with a height of 0.35m. (a) $\sim$ (b): pre-jump phase, (c): flight phase, (d): post-landing phase. The x, y and z axes of the world frame are labeled in red, green and blue.

A. Implementation

The trajectory optimization framework is implemented in CasADi [20] with Python using the the interior-point solver IPOPT [21]. We write the whole body controller with C++ using the Pinocchio library [22] to compute full rigid body dynamics and qpOASES [23] to solve the QP problem. The whole body controller runs at 1kHz to track the desired trajectory. We carried out the experiments in the robot simulation environment Gazebo with the physics engine ODE, using the full rigid body dynamics of the real SLIDER robot. The communication between different levels

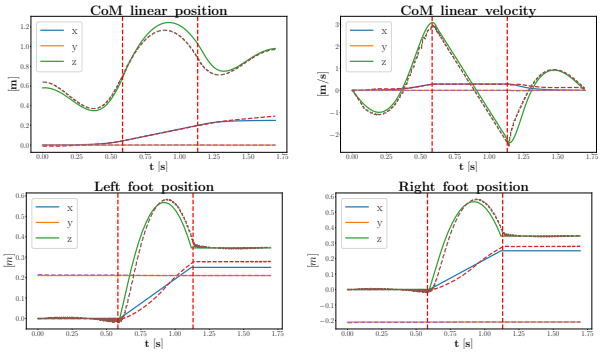
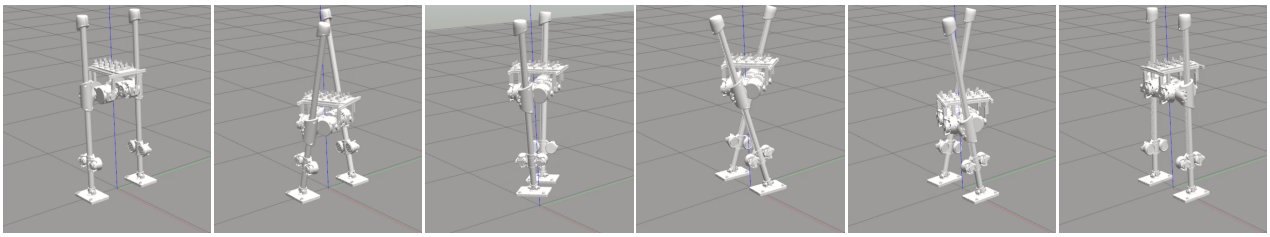


Fig. 7: The trajectories generated for jumping forward onto a box (height: 35cm). In each plot the vertical dashed line split the whole motion into the three phases: *pre-jump*, *flight* and *post-landing*. The desired motions are plotted in full line and the measured motions are plotted in dashed line.



(a) (b) (c) (d) (e) (f)

Fig. 8: Snapshots of SLIDER performing a twisting jump with a yaw orientation change of 90 degrees. From left to right: (a) $\sim$ (b): pre-jump phase, (c) $\sim$ (d): flight phase, (e) $\sim$ (f): post-landing phase. The x, y and z axes of the world frame are labeled in red, green and blue.

of the control hierarchy (see Fig. 3) is achieved through ROS.

B. Jump onto a Box

We conducted experiments on jumping onto boxes with different heights while minimizing the energy consumption. In trajectory optimization we assign 20 segments in each phase. Compared with the anthropomorphic robot design which has knees, SLIDER's straight leg has a larger range of motion, allowing SLIDER jump to the same height with a relatively smaller CoM height. Because there is no knee on the leg, the foot can go all the way up until the ankle touches the pelvis, therefore SLIDER can jump onto a high box. The maximum height of the box SLIDER can jump with current setting is 0.35m, and the maximum height of CoM throughout the entire motion is only 1.25m. Considering that SLIDER is 1.2m high, the robot can jump onto a box equal to 30% of its total height. The CoM and foot motion of this jump are shown in Fig 7.

C. Twisting Jump

To accomplish the twisting jump, the robot should be able to rotate its body around the vertical axis. However the original design with two 5 DoF legs can not achieve this since the lacking of hip yaw joints. To enable this capability, we have added two joints for the robot, one yaw joint per leg. The leg now has 6 DoF: hip roll, hip pitch, hip yaw, knee pitch, ankle roll and ankle pitch. In the trajectory optimization we simply provide a linear interpolation of the orientation as the initial guess. Also we assign a large weight to the *final state target* in the cost function to encourage the solver to find a trajectory that reaches the target pose. In the whole body controller, we assign large weights to base orientation and angular momentum tasks to ensure these two tasks have higher priority than other tasks. SLIDER can successfully perform a twisting jump with a yaw orientation change of 90 degrees, snapshots of the motion is shown in Fig 8. The generated motion plans of base orientation and angular momentum are shown in Fig 9.

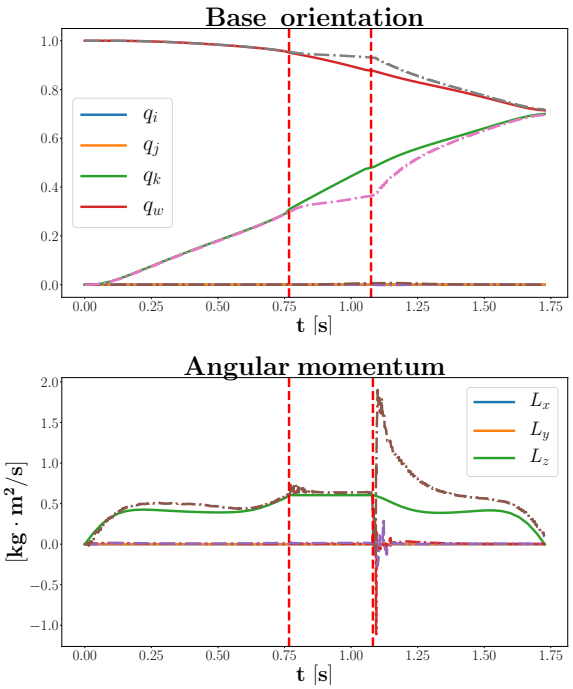


Fig. 9: Planned twisting jump motion with an orientation change of 90 degrees for the robot SLIDER. The top plot shows the trajectories of the base orientation and the bottom plot shows the trajectories of the angular momentum. In each plot the vertical dashed line split the whole motion into the three phases: *pre-jump*, *flight* and *post-landing*. The desired trajectories are plotted in solid line and the measured ones are plotted in dashed line.

VI. DISCUSSION

Real World Feasibility: Based on our measurement the slide joint is the main driving joint in jumping motions and requires the highest force/torque among all joints. In the experiment of jumping onto a box, the average value of measured force in the slide joint is 75 N and the peak value of measured force is 500 N. In the experiment of the twisting jump, the average value of measured force is 60 N and the peak value of measured force is 230 N. In the real SLIDER, each slide joint can produce a maximum force of 300 N. Therefore it is possible for real SLIDER to perform a twisting jump given yaw joints implemented. The real SLIDER cannot jump onto a box with a height of 35 cm, however it is possible for the robot to jump onto a lower box. Many bipedal robots with knees will suffer from the problem of breaking knees when performing highly dynamic jump,

which is due to the high impact generated during landing. Because of SLIDER's unique design, our robot doesn't have this problem. With a gear ratio of 1:4, the slide joint is highly backdrivable and can absorb the impact.

Limitation of the Single Rigid Body Model: The single rigid body model which we use in our planning assumes that all masses and inertia concentrated in the pelvis. Although this assumption well approximates the SLIDER robot and simplifies the planning a lot, it doesn't hold partially or completely in some cases. As shown in Fig 9, the base orientation tracking has not been tracked as well as the angular momentum in the flight phase. This is due to the neglected leg inertia during motion planning. We also observed that SLIDER cannot jump forward further than 35 cm. To jump further than 35 cm, the leg has to have a big swing and the leg inertia plays an important role in flight phase. It is interesting to consider adding leg inertia into the model during planning. It will improve the jumping performance while still keep the simplicity of the reduced order model. With this modification we can also extend our approach to bipedal robot with knees, of which the leg is heavy and the inertia of the leg cannot be neglected.

VII. CONCLUSIONS

In this paper, we generated highly dynamic jumping motions for our bipedal robot SLIDER. Due to its unique design, both planning and control of the robot are simplified. It is a perfect example to show how the performance of a robot can be optimized through the systematic consideration of both software and hardware. We believe that our proposed approach can also be extended to other bipedal robots with minor modifications. We verified our proposed control framework in simulation and showed that SLIDER can jump onto a box with 35 cm height and perform a twisting jump with a yaw orientation change of 90 degrees. Due to COVID-19 lockdown, currently we cannot perform experiments on the real robot. We are also planning to include leg inertia in the planning to improve the jumping performance.

APPENDIX

QUATERNION TO ROTATION MATRIX CONVERSION

Given a quaternion $\boldsymbol{q} = q_w + q_x\boldsymbol{i} + q_y\boldsymbol{j} + q_z\boldsymbol{k}$ which represents the orientation of the single rigid body in world frame, the equivalent rotation matrix representation can be derived as:

$$\boldsymbol{R} = \begin{bmatrix} 1-2\left(q_y^2+q_z^2\right) & 2q_xq_y-2q_wq_z & 2q_wq_y+2q_xq_z \\ 2q_xq_y+2q_wq_z & 1-2\left(q_x^2+q_z^2\right) & 2q_yq_z-2q_wq_x \\ 2q_xq_z-2q_wq_y & 2q_wq_x+2q_yq_z & 1-2\left(q_x^2+q_y^2\right) \end{bmatrix}$$

DERIVATIVE OF QUATERNION

Followed by [14], given a rigid body with quaternion $\boldsymbol{q} = q_w + q_x\boldsymbol{i} + q_y\boldsymbol{j} + q_z\boldsymbol{k}$ and with angular velocity $\boldsymbol{\omega}$, the derivative of the quaternion $\dot{\boldsymbol{q}}$ can be calculated as:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \circ \boldsymbol{\omega}$$

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \\ \dot{q}_w \end{bmatrix} = \frac{1}{2}\begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} \circ \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \\ \dot{q}_w \end{bmatrix} = \frac{1}{2}\begin{bmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \\ 0 \end{bmatrix}$$

or

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \\ \dot{q}_w \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}$$

INERTIA TENSOR

Inertia tensor in the local body frame:

$$\boldsymbol{I}_B = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$\boldsymbol{I}_B^{-1} = \begin{bmatrix} 1/I_{xx} & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 1/I_{zz} \end{bmatrix}$$

Inertia tensor in the world inertia frame:

$$\boldsymbol{I}_W = \boldsymbol{R}_B\boldsymbol{I}_B\boldsymbol{R}_B^T$$

$$\boldsymbol{I}_W^{-1} = \boldsymbol{R}_B\boldsymbol{I}_B^{-1}\boldsymbol{R}_B^T$$

where $\boldsymbol{R}_B$ is the body rotation matrix expressed in the world inertia frame.

ACKNOWLEDGMENT

Ke Wang is funded by the CSC Imperial Scholarship. Songyan Xin is supported by the EPSRC UK RAI Hub in Future AI and Robotics for Space (FAIR-SPACE, project ID: EP/R026092/1). Digby Chappell is funded by the UKRI Centre for Doctoral Training in AI for Healthcare.

REFERENCES

[1] Raibert, Marc H. Legged robots that balance. MIT press, 1986.

[2] Full, Robert J., and Daniel E. Koditschek. "Templates and anchors: neuromechanical hypotheses of legged locomotion on land." Journal of experimental biology 202.23 (1999): 3325-3332.

[3] Mordatch, Igor, Martin De Lasa, and Aaron Hertzmann. "Robust physics-based locomotion using low-dimensional planning." ACM SIGGRAPH 2010 papers. 2010. 1-8.

[4] Wensing, Patrick M., and David E. Orin. "High-speed humanoid running through control with a 3D-SLIP model." 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013.

[5] Wensing, Patrick M., and David E. Orin. "Development of high-span running long jumps for humanoids." 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014.

[6] Di Carlo, Jared, et al. "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[7] Kim, Donghyun, et al. "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control." arXiv preprint arXiv:1909.06586 (2019).

[8] Katz, Benjamin, Jared Di Carlo, and Sangbae Kim. "Mini cheetah: A platform for pushing the limits of dynamic quadruped control." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.
[9] Winkler, Alexander W., et al. "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization." IEEE Robotics and Automation Letters 3.3 (2018): 1560-1567.
[10] Apgar, Taylor, et al. "Fast Online Trajectory Optimization for the Bipedal Robot Cassie." Robotics: Science and Systems. Vol. 101. 2018.
[11] Mastalli, Carlos, et al. "Crocoddyl: An efficient and versatile framework for multi-contact optimal control." arXiv preprint arXiv:1909.04947 (2019).
[12] Koenemann, Jonas, et al. "Whole-body model-predictive control applied to the HRP-2 humanoid." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.
[13] Wang, Ke, et al. "Design and Control of SLIDER: An Ultra-lightweight, Knee-less, Low-cost Bipedal Walking Robot." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.
[14] Graf, Basile. "Quaternions and dynamics." arXiv preprint arXiv:0811.2889 (2008).
[15] Kuindersma, Scott, et al. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot." Autonomous robots 40.3 (2016): 429-455.
[16] Feng, Siyuan, et al. "Optimization-based full body control for the darpa robotics challenge." Journal of Field Robotics 32.2 (2015): 293-312.
[17] Herzog, Alexander, et al. "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid." Autonomous Robots 40.3 (2016): 473-491.
[18] Kelly, Matthew. "An introduction to trajectory optimization: How to do your own direct collocation." SIAM Review 59.4 (2017): 849-904.
[19] Orin, David E., Ambarish Goswami, and Sung-Hee Lee. "Centroidal dynamics of a humanoid robot." Autonomous robots 35.2-3 (2013): 161-176.
[20] Andersson, Joel AE, et al. "CasADi: a software framework for nonlinear optimization and optimal control." Mathematical Programming Computation 11.1 (2019): 1-36.
[21] Wächter, Andreas, and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." Mathematical programming 106.1 (2006): 25-57.
[22] Carpentier Justin, et al. "Pinocchio: fast forward and inverse dynamics for poly-articulated systems", https://stack-of-tasks.github.io/pinocchio, (2015–2021)
[23] Ferreau, Hans Joachim, et al. "qpOASES: A parametric active-set algorithm for quadratic programming." Mathematical Programming Computation 6.4 (2014): 327-363.