Integration of Riemannian Motion Policy with Whole-Body Control for Collision-Free Legged Locomotion

Daniel Marew [1], Misha Lvovsky[1], Shangqun Yu[1], Shotaro Sessions[1], and Donghyun Kim[1]

***Abstract*— In this paper, we present a Riemannian Motion Policy (RMP)flow-based whole-body control framework for improved dynamic legged locomotion. RMPflow is a differential geometry-inspired algorithm for fusing multiple task-space policies (RMPs) into a configuration space policy in a geometrically consistent manner. RMP-based approaches are especially suited for designing simultaneous tracking and collision avoidance behaviors and have been successfully deployed on serial manipulators. However, one caveat of RMPflow is that it is designed with fully actuated systems in mind. In this work, we, for the first time, extend it to the domain of dynamic-legged systems, which have unforgiving under-actuation and limited control input. Thorough push recovery experiments are conducted in simulation to validate the overall framework. We show that expanding the valid stepping region with an RMP-based collision-avoidance swing leg controller improves balance robustness against external disturbances by up to 53% compared to a baseline approach using a restricted stepping region. Furthermore, a point-foot biped robot is purpose-built for experimental studies of dynamic biped locomotion. A preliminary unassisted in-place stepping experiment is conducted to show the viability of the control framework and hardware.**

I. Introduction

A. Motivation

In dynamic legged locomotion, the size of a valid stepping region is crucial as it determines the maximum CoM velocity that can be safely regulated [1]. Previous work on dynamic legged locomotion conservatively restricted the stepping region in the lateral direction so that the robot's legs do not cross one another in order to mitigate the risk of self-collision [2]–[4]. However, this restriction prevents the robots from taking aggressive but stabilizing steps, thus reducing their balance stability and robustness to external disturbances.

In this work, we aim to address this undue trade-off by deploying an RMPflow-based [5] reactive collision-avoidance swing leg controller. The controller steers the swing foot towards the planned step location while avoiding collisions between the robot's links. This approach enables full utilization of the kinematically reachable region as a valid stepping area, thereby widening the robot's stability margin and improving its robustness to external disturbances.

B. Related Works

Collision avoidance algorithms can generally be divided into two categories. The first encompasses planning-based approaches wherein long horizon collision-free trajectories are generated using either sampling-based motion planning techniques, such as Rapidly-Exploring Random Trees (RRTs) [6]–[8], or optimization-based techniques [9]–[11]. Although these methods are commonly employed in robotic manipulators, they tend to be computationally intensive, rendering them less suitable for real-time applications, especially those involving high-speed movements as found in dynamic legged locomotion.

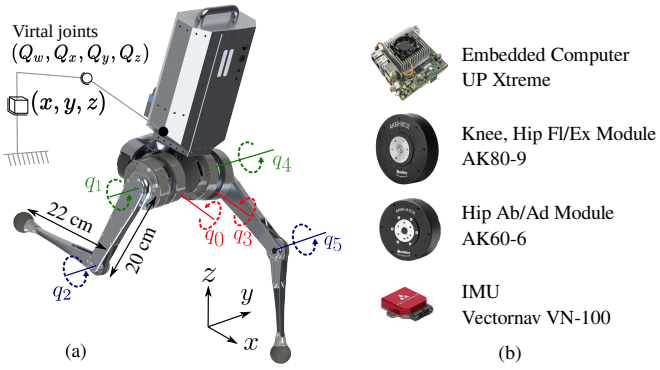

Fig. 1. **Point-foot biped robot, Pat.** (a) Small scale point-foot biped with 6 actuated degrees of freedom. (b) Off-the-shelf components used to build the robot.

The second strategy to tackle the collision avoidance problem involves the application of reactive controllers. Historically, Artificial Potential Field (APF)-based strategies have been the predominant technique for designing reactive collision avoidance behaviors. APF-based methods achieve collision avoidance by applying a virtual repulsive force to the robot to steer it away from obstacles [12], [13]. However, these techniques can pose calibration challenges, often neglect full-body dynamics, and frequently produce undesired behaviors due to local minima.

Another widely used reactive method is using a constrained optimization [14]–[16]. In this context, the recently proposed Control Barrier Functions (CBFs) are particularly noteworthy. Specifically, Khazoom et al. derived acceleration level self-collision avoidance constraints using signed distance-based CBFs and incorporated them into a whole-body control QP formulation. However, these methods do not tend to scale efficiently as the number of considered collision elements increases. Each additional collision element introduces a new constraint to the optimization problem, potentially rendering these methods impractical for fast and real-time applications. Moreover, conflicts between these constraints can sometimes render the optimization problem infeasible [16].

RMPflow [5] is a promising alternative to these approaches

Authors are with the [1] Manning College of Information and Computer Sciences at University of Massachusetts Amherst, Amherst, MA, 140 Governors Dr, Amherst, MA 01002, USA. Corresponding Author: robot.dhkim@gmail.com

because it scales well with the number of collision tasks requiring a limited amount of additional computation power, making it suitable for fast and real-time operations. Moreover, it mitigates local-minima-induced undesired behaviors of APF-based methods by specifying the behavior with an additional highly non-linear priority Riemannian metric that stretches the space in the direction of obstacles [17]. This priority metric heavily penalizes motion in the direction of obstacles while being indifferent to motion in the orthogonal direction [5]. In addition, because the priority metric can be velocity dependent, it can be designed so that the robot ignores nearby obstacles it is moving away from seamlessly [17]. This behavior is crucial for high-speed operations and cannot be achieved with APF methods that rely solely on position-dependent repulsive forces.

In the RMPflow framework [5], [17], each desired task-space behavior is specified by a local motion policy called a Riemannian motion policy (RMP), which is composed on the task space manifold. An RMP consists of an acceleration policy and an associated Riemannian metric, also known as a state-dependent inertia metric, that encodes information about the task space's geometry and the relative priority of the task. The RMPflow algorithm fuses these local RMPs into a single configuration-space ($\mathcal{C}$-space) RMP using an operator called the pullback operator.

One caveat of RMPflow is its implicit assumption that the $\mathcal{C}$-space acceleration policy obtained through the pullback operations is always realizable by the robot. This assumption is often not satisfied when it comes to underactuated systems like legged robots. To-date, the framework has been primarily applied to fully actuated systems such as serial manipulators [5], [17]–[20], with one notable exception [21]. Wingo et al. [21] proposed an extension of RMPflow for a class of underactuated wheeled inverted pendulum (WIP) robots. An actuated joint directly controls the floating base of the class of WIP robots considered in [21], [22]. The underactuation of this class of robots only emanates from the fact that this joint shares the same control torque as the wheelbase. This kind of underactuation lends itself to the type of dynamics separation scheme the RMPflow formulation in [21] relies on. This underactuation, however, differs from the underactuation of legged systems whose floating base can not be directly controlled [22]. Thus the RMPflow framework developed in [21] can not be directly applied to legged systems.

In this paper, we address this gap by integrating RMPflow with a traditional null space projection-based whole-body controller formulation so that tasks specified in terms of RMPs can be realized on legged systems. In our approach, the RMPs are executed in a way that is consistent with robot's contact-constrained dynamics while not compromising the tracking performance of higher priority tasks such as floating base tasks. Specifically, we formulate a constrained weighted least-squares problem inspired by the RMPflow formulations in [5] and [21] whose optimal solution, a $\mathcal{C}$-space acceleration command, will try to realize RMPs as faithfully as possible, giving priority according to the assigned metric. Moreover, the solution is by construction guaranteed not to violate contact constraints and constraints derived from higher priority tasks.

Point-foot bipeds are relatively simple legged systems, but since they are severely underactuated and highly unstable, they are extremely challenging to control, making them an ideal experimental platform for testing the limits of dynamic biped locomotion. To that end, this paper presents a new small point-foot biped named Pat designed and built to experimentally validate the proposed whole-body framework.

In summary, the key contributions of our study include the following:

1) We present a new formulation for integrating RMPflow with a null-space projection-based whole-body controller, enabling its application in legged robots.
2) Our extensive simulation experiments highlight that our proposed collision-avoidance swing-leg controller, developed based on the aforementioned formulation, significantly enhances the robustness of a point-foot biped robot against external disturbances. In addition, we apply and test high-speed self-collision avoidance on a quadruped robot, and provide a comparative analysis of our results against APF and CBF based methods.
3) We designed a low cost point-foot biped robot for the experimental study of dynamic biped locomotion. The performance of our proposed controller and the viability of the robot's hardware is validated through successful demonstrations of unassisted, in-place walking.

II. Integration of Riemannian Motion Policy and Whole-Body Control

This section details the proposed approach for integrating RMPflow with the traditional null-space projection-based whole-body controller formulation. We use the following general equations of motion of legged robots

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

where $\boldsymbol{A} \in \mathbb{R}^{n+6\times n+6}$, $\mathbf{b} \in \mathbb{R}^{n+6}$, and $\mathbf{g} \in \mathbb{R}^{n+6}$, are the generalized mass matrix, coriolis force, and gravitation forces, respectively. $\boldsymbol{S}_a \in \mathbb{R}^{n+6\times n+6}$ is the actuated joint selection matrix. $\boldsymbol{\tau} \in \mathbb{R}^{n+6}$, $\mathbf{f}_r \in \mathbb{R}^{3\cdot n_c}$, and $\boldsymbol{J}_c \in \mathbb{R}^{3\cdot n_c \times n+6}$ are joint torque, augmented reaction force and contact Jacobian, respectively. $\ddot{\mathbf{q}} \in \mathbb{R}^{6+n}$ is the configuration space acceleration where n the number of actuated degrees of freedom and n_c is the number of contact points.

A. Review of RMPflow

In this section, we briefly introduce the RMPflow computational framework first proposed in [5]. RMP composed on an m-dimensional manifold $\mathcal{M}$ is characterized, in its canonical form, by the tuple $(\mathbf{a}, \boldsymbol{M})^{\mathcal{M}}$, where $\mathbf{a} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is an acceleration policy and $\boldsymbol{M} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m\times m}$ is a state-dependent Riemannian metric. This tuple can be written in its natural form, $(\mathbf{f}, \boldsymbol{M})^{\mathcal{M}}$, by using $\mathbf{f} = \boldsymbol{M}\mathbf{a}$ where $\mathbf{f}$ can be considered as a virtual force.

RMPflow uses a tree-like data structure called RMP-tree to efficiently encode the hierarchical relationship between RMPs. In general, operational space tasks specified in terms of RMPs are placed at the leaves of the RMP tree, and the root-node RMP represents the configuration space policy. RMPflow uses three operators that consitute the RMP-algebra. These are; **pushforward**, **pullback**, and **resolve**. In the first stage of the RMPflow algorithm, the **pushforward** operator is used to propagate state (position and velocity) information from the root, $\mathcal{C}$-space, to the leaves of the RMP tree, task space. In the case of typical whole-body control formulation, this is similar to updating the position and velocity of task space based on the state of the configuration space using forward kinematics and the Jacobian map, respectively.

In the subsequent stage of RMPflow, starting at the RMP tree leaves, the pullback operator is recursively applied using (2) to obtain the combined inertia metric and virtual force of the parent RMPs and eventually that of the $\mathcal{C}$-space RMP.

$$\mathbf{f}' = \sum_{i=1}^{K} \boldsymbol{J}_i^\top \left(\mathbf{f}_i - \boldsymbol{M}_i \dot{\boldsymbol{J}}_i \dot{\mathbf{x}} \right), \ \boldsymbol{M}' = \sum_{i=1}^{K} \boldsymbol{J}_i^\top \boldsymbol{M}_i \boldsymbol{J}_i, \quad (2)$$

where $(\mathbf{f}_i, \boldsymbol{M}_i)^{\mathcal{N}_i}$ denotes the i^{th} child RMP, $\boldsymbol{J}_i$ represents the associated Jacobian matrix, $\dot{\mathbf{x}}_i$ is the task space velocity, and K is the total number of child RMPs. Lastly, **resolve** is used to transform the computed natural form RMP, $(\mathbf{f}, \boldsymbol{M})^{\mathcal{M}}$ to its canonical form $(\mathbf{a}, \boldsymbol{M})^{\mathcal{M}}$ where $\mathbf{a} = \mathbf{M}^\dagger \mathbf{f}$ and $\dagger$ denotes Moore-Penrose inverse.

B. Review of null-space projection based task prioritization

We employ the dynamically consistent null space projection technique to impose a strict hierarchy between tasks. It can be described with the following recursion rule from [23].

$$\ddot{\mathbf{q}}_i^{\text{cmd}} = \ddot{\mathbf{q}}_{i-1}^{\text{cmd}} + \overline{\boldsymbol{J}_{i|pre}^{\text{dyn}}} \left(\ddot{\mathbf{x}}_i^{\text{cmd}} - \dot{\boldsymbol{J}}_i \dot{\mathbf{q}} - \boldsymbol{J}_i \ddot{\mathbf{q}}_{i-1}^{\text{cmd}} \right), \quad (3)$$

where

$$\boldsymbol{J}_{i|pre} = \boldsymbol{J}_i \boldsymbol{N}_{i-1}, \quad (4)$$

$$\begin{aligned} \boldsymbol{N}_{i-1} &= \boldsymbol{N}_0 \boldsymbol{N}_{1|0} \cdots \boldsymbol{N}_{i-1|i-2}, \\ \boldsymbol{N}_{i|i-1} &= \boldsymbol{I} - \overline{\boldsymbol{J}_{i|i-1}^{\text{dyn}}} \boldsymbol{J}_{i|i-1}, \\ \boldsymbol{N}_0 &= \boldsymbol{I} - \overline{\boldsymbol{J}_c^{\text{dyn}}} \boldsymbol{J}_c, \end{aligned} \quad (5)$$

Here, $i \geq 1$, and

$$\ddot{\mathbf{q}}_0^{\text{cmd}} = \overline{\boldsymbol{J}_c^{\text{dyn}}} (-\boldsymbol{J}_c \dot{\mathbf{q}}). \quad (6)$$

$\ddot{\mathbf{x}}_i^{\text{cmd}}$ is the acceleration policy of i-th task defined by the PD controller.

$$\ddot{\mathbf{x}}_i^{\text{cmd}}(\mathbf{x}, \dot{\mathbf{x}}) = \ddot{\mathbf{x}}^{\text{des}} + \boldsymbol{K}_p \left(\mathbf{x}_i^{\text{des}} - \mathbf{x}_i \right) + \boldsymbol{K}_d \left(\dot{\mathbf{x}}^{\text{des}} - \dot{\mathbf{x}} \right), \quad (7)$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are position and velocity feedback gains, respectively. $\boldsymbol{J}_{i|pre}$ is the projection of the i-th task Jacobian into the null space of the prior tasks. $\boldsymbol{J}_c$ is a contact Jacobian and the dynamically consistent pseudo-inverse is denoted by an overline and superscript 'dyn' as in [24] and defined as

$$\overline{\boldsymbol{J}^{\text{dyn}}} = \boldsymbol{A}^{-1} \boldsymbol{J}^\top \boldsymbol{\Lambda}. \quad (8)$$

Here, $\boldsymbol{\Lambda}$ is the operational mass matrix given by

$$\boldsymbol{\Lambda} = \left(\boldsymbol{J} \boldsymbol{A}^{-1} \boldsymbol{J}^\top \right)^{-1}. \quad (9)$$

We would like to reiterate that the contact task is always given the highest priority (6) and all tasks including RMP-based tasks in III-C are executed in the null-space of this task to prevent contact constraint violations.

C. Integrating RMP with prioritized task execution

In this section, we will provide a modification of the pull-back operation at the root of the RMP-Tree, which computes the final $\mathcal{C}$-space acceleration command. The modification allows us to realize tasks specified in terms of RMPs in a way that does not interfere with the execution of higher priority tasks. To do so, we define the final $\mathcal{C}$-space acceleration command in the following way.

$$\ddot{\mathbf{q}}^{\text{cmd}} = \ddot{\mathbf{q}}_k^{\text{cmd}} + \boldsymbol{N}_k \ddot{\mathbf{q}}_{\text{rmp}} \quad (10)$$

where $\ddot{\mathbf{q}}_k^{\text{cmd}}$ and $\boldsymbol{N_k}$ are the acceleration command and null space projection matrix derived from the first k higher priority tasks, including the contact constraint, using successive null space projections described in II-B. $\ddot{\mathbf{q}}_{\text{rmp}}$ is any arbitrary $\mathcal{C}$-space acceleration that can be used to realize RMPs. We obtain the optimal $\mathcal{C}$-space acceleration command by solving the following constrained weighted least-squares problem.

$$\begin{aligned} &\min_{\ddot{\mathbf{q}}^{\text{cmd}}} \sum_{i=0}^{n} ||\boldsymbol{J}_i \ddot{\mathbf{q}}^{\text{cmd}} + \dot{\boldsymbol{J}}_i \dot{\mathbf{q}} - \ddot{\mathbf{x}}_i||_{\mathbf{M}_i}^2 \\ &\text{s.t.} \\ &\quad \ddot{\mathbf{q}}^{\text{cmd}} = \ddot{\mathbf{q}}_k^{\text{cmd}} + \boldsymbol{N}_k \ddot{\mathbf{q}}_{\text{rmp}} \end{aligned} \quad (11)$$

Where $(\ddot{\mathbf{x}}_i, \mathbf{M}_i)^{\mathcal{N}_i}$ is the i^{th} child RMP of the root RMP and $\boldsymbol{J}_i$ is the Jacobian that maps $\mathcal{C}$-space velocities to the i^{th} child nodes' task space. The optimal solution to (11) will try to realize all the tasks specified in the RMP-tree as faithfully as possible, giving priority according to the assigned metric $\mathbf{M}_i$. By construction it is guaranteed not to violate constraints obtained from the higher priority tasks and is consistent with the contact constrained dynamics of the robot.

The constrained least-squares problem in (11) can be reformulated into an unconstrained least-squares problem in (12) by moving the constraint to the cost.

$$\ddot{\mathbf{q}}_{\text{rmp}}^* = \arg\min_{\ddot{\mathbf{q}}_{\text{rmp}}} \sum_{i=0}^{n} ||\boldsymbol{J}_i \boldsymbol{N}_k \ddot{\mathbf{q}}_{\text{rmp}} - \hat{\ddot{\mathbf{x}}}_i||_{\mathbf{M}_i}^2 \quad (12)$$

where $\hat{\ddot{\mathbf{x}}} = \ddot{\mathbf{x}}_i - \dot{\boldsymbol{J}}_i \dot{\mathbf{q}} - \boldsymbol{J}_i \ddot{\mathbf{q}}_k$. The analytical solution to (12) yields the modified pullback operation given by (13)-(14).

$$\mathbf{M}_{\text{rmp}} = \boldsymbol{N}_k^\top \left(\sum_{i=0}^{n} \boldsymbol{J}_i^\top \mathbf{M}_i \boldsymbol{J}_i \right) \boldsymbol{N}_k \quad (13)$$

$$\mathbf{f}_{\text{rmp}} = \boldsymbol{N}_k^\top \left(\sum_{i=0}^{n} \boldsymbol{J}_i^\top \left(\mathbf{f}_i - \mathbf{M}_i \dot{\boldsymbol{J}}_i \dot{\mathbf{q}} \right) - \boldsymbol{J}_i^\top \mathbf{M}_i \boldsymbol{J}_i \ddot{\mathbf{q}}_k \right) \quad (14)$$

where $\mathbf{M}_{\text{rmp}}$ and $\mathbf{f}_{\text{rmp}}$ are the the projected virtual inertia metric and virtual force computed through the modified pullback operations in (13) and (14) respectively. Note the term inside the bracket in (13) and the first term in (14) are the regular RMPflow pullback operations in (2). Thus $\mathbf{M}_{\text{rmp}}$, $\mathbf{f}_{\text{rmp}}$ can be obtained from the $\mathcal{C}$-space RMP $(\mathbf{M}, \mathbf{f})^{\mathcal{M}}$ (15).

$$\mathbf{M}_{\text{rmp}} = \boldsymbol{N}_k^\top \mathbf{M} \boldsymbol{N}_k, \;\; \mathbf{f}_{\text{rmp}} = \boldsymbol{N}_k^\top \left(\mathbf{f} - \mathbf{M} \ddot{\mathbf{q}}_k^{\text{cmd}} \right) \tag{15}$$

The final acceleration command is then obtained by

$$\ddot{\mathbf{q}}^{\text{cmd}} = \ddot{\mathbf{q}}_k^{\text{cmd}} + \boldsymbol{N}_k \mathbf{M}_{\text{rmp}}^\dagger \mathbf{f}_{\text{rmp}} \tag{16}$$

D. Whole-body control (WBC)

In this work, we execute the acceleration command from (16) using a convex MPC [25] and whole body impulsive control (MPC + WBIC) based controller proposed in [24]. In [24], an optimal reaction force profile is computed using a convex MPC based on a simplified single rigid body model of the robot. This reaction force is then tracked along side acceleration commands while considering the full-body dynamics of the robot by solving the following quadratic program (QP) (17). Once the optimal $\mathbf{f}_r$ and $\ddot{\mathbf{q}}$ are obtained by solving the QP in (17), inverse dynamics is used to compute the torque command.

$$\min_{\boldsymbol{\delta}_{\mathbf{f}_r}, \boldsymbol{\delta}_f} \quad \boldsymbol{\delta}_{\mathbf{f}_r}^\top \boldsymbol{Q}_1 \boldsymbol{\delta}_{\mathbf{f}_r} + \boldsymbol{\delta}_f^\top \boldsymbol{Q}_2 \boldsymbol{\delta}_f \tag{17}$$

s.t.

$$\boldsymbol{S}_f \left(\boldsymbol{A} \ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} \right) = \boldsymbol{S}_f \boldsymbol{J}_c^\top \mathbf{f}_r \qquad \text{(floating base dyn.)}$$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^{\text{cmd}} + \begin{bmatrix} \boldsymbol{\delta}_f \\ \mathbf{0}_{n_j} \end{bmatrix} \qquad \text{(acceleration)}$$

$$\mathbf{f}_r = \mathbf{f}_r^{\text{MPC}} + \boldsymbol{\delta}_{\mathbf{f}_r} \qquad \text{(reaction forces)}$$

$$\boldsymbol{W} \mathbf{f}_r \geq \mathbf{0}, \qquad \text{(contact force constraints)}$$

where $\mathbf{f}_r^{\text{MPC}}$ is the reaction force command computed by the MPC, and $\boldsymbol{S}_f$ is the floating base selection matrix. $\boldsymbol{J}_c$ and $\boldsymbol{W}$ are the augmented contact Jacobian and contact constraint matrix respectively. $\boldsymbol{\delta}_f$ and $\boldsymbol{\delta}_{\mathbf{f}_r}$ are relaxation variables for the floating base acceleration and reaction forces.

III. Point-foot Biped Locomotion

This section details the design of the experimental platform and the components of the proposed control framework summarized in Fig.2.

A. Experimental platform

This section details the components used in our new point-foot bipedal robot Pat depicted in Fig. 1. Pat is a 70cm tall 5.4kg 6-DoF point-foot biped designed to be used as a small-scale, low-cost experimental platform that can be quickly built and tested without requiring significant maintenance effort. Thus, it primarily employs off-the-shelf components. Pat's legs are a version of the MIT-Mini-Cheetah [26] legs modified to work with off-the-shelf actuators. We use two types of torque-controlled electric actuators from T-motor to drive the robot's six actuated degrees of freedom. These are the AK80-9 and the AK60-6 actuators [27], which consist of thin, large-diameter out-runner motors, a motor controller based on the open source Mini-Cheetah motor controller [26], and a planetary gear reducer embedded into the stator of the motor. The gear reductions for the actuator modules are 9:1 and 6:1, respectively. The hip abduction and flexion axes on each leg are driven directly by the AK60-6 and AK80-9 actuators, respectively, and each knee flexion joint is driven by an AK80-9 actuator connected to a timing belt. The timing belt affords the actuator used on the knee an additional reduction factor of $\frac{14}{9} \approx 1.5$. An onboard UP-Xtreme computer with Intel Core i3-8145UE processor running CONFIG_PREEMPT_RT patched Ubuntu 18.04 is used to control the robot. The computer sends commands and receives sensory data to and from the actuators via CAN 2.0 communication protocol at 500 hz.

B. Gait control

Each gait cycle has two swing phases and two brief dual support phases. These brief dual support phases are required for point-foot bipeds to perform yaw control. In addition to the dual support and swing phases, there are four transition phases that we use to facilitate a smoother contact transition [2]. In particular, during these transition phases, slowly changing upper and lower bounds are introduced in the reaction force computation in (17) to avoid discontinuous reaction force commands, which can cause jerky motion.

We use the time-to-velocity reversal (TVR) planner [2] to determine the upcoming footstep location. The planner is called once in the middle of each swing phase. In addition, TVR is used to generate future reference foot-step locations for the MPC. The swing leg is steered towards chosen footstep location following a minimum jerk trajectory.

C. Self-collision avoidance

In this work, an RMPflow-based reactive collision-avoidance swing leg controller is used to steer the swing foot towards the planned step location while avoiding collisions between the robot's links. This swing-leg controller allows the robot to fully use its kinematically reachable region as a valid stepping area, thereby improving its robustness to external disturbance. We use two types of RMPs to accomplish this swing-leg behavior: an **attractor-RMP** and a set of **collision-avoidance-RMPs**.

The **attractor-RMP** is used to move the foot towards the planned footstep location and is specified by the PD controller acceleration policy in (7) and the operational space inertia metric in (9). Note that the attractor RMP used in this work is not designed to meet the definition of a geometric dynamical system (GDS) required by [17] to guarantee RMPflow's stability in the Lyapunov-sense. But, similar to Q-Function-based-RMPs described in [5], it can be considered a part of a broader class of RMPs that do not meet this criterion but still have practical use.

Each **collision-avoidance-RMP** is tasked with avoiding collision between a pair of control and collision points that lie on the swing leg and stance leg, respectively. For computational efficiency, we approximate the geometry of

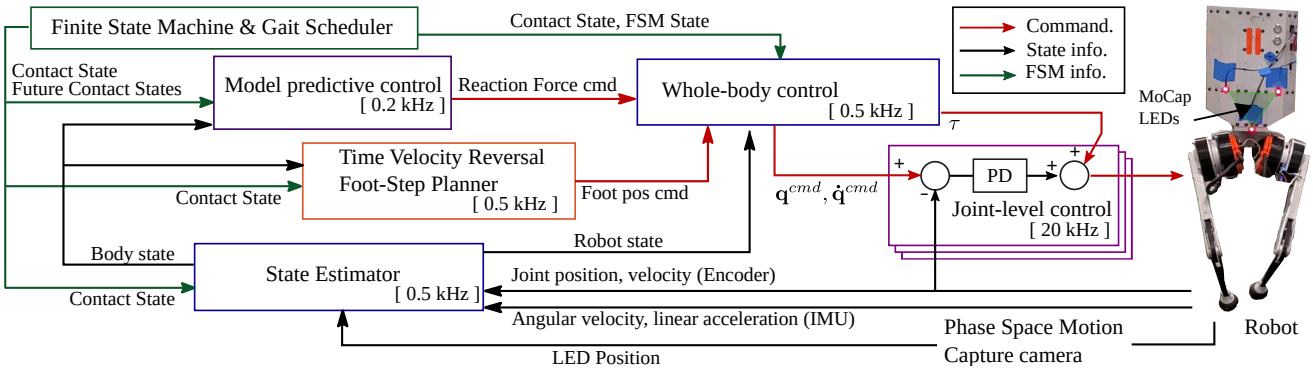

Fig. 2. **Overall Control Framework.** A time-scripted finite state machine is used to drive the system. WBC takes reference reaction force from MPC, foot trajectory from the TVR planner, and constant body posture command. Then WBC outputs joint position, velocity and toque commands which are then executed by the joint-level controllers. The body velocity is estimated by MoCap LED position data and used for foot step planning. The WBC in this figure includes the prioritized null-space projection, RMPflow and the QP computation.

each leg's links with a capsule and reduce the link-to-link collision avoidance problem to the more simplified problem of avoiding collision between the two closest points on the capsules called the witness points. Thus, only one collision RMP is required per link pair. We use the algorithm proposed in [28] to compute the position of the witness points. The collision avoidance RMPs used in this work are defined on the one-dimensional Euclidean distance space and are specified by the repulsive acceleration policy and metric pair in Eq. (18) and (19), respectively [17].

$$\ddot{x}(x,\dot{x}) = k_p \exp\left(-x/l_p\right) - k_d \frac{\sigma(\dot{x})\dot{x}}{x/l_d + \epsilon_d} \tag{18}$$

$$m(x,\dot{x}) = \sigma(\dot{x})g(x)\frac{\mu}{x/l_m + \epsilon_m} \tag{19}$$

where $\sigma(\dot{x}) = 1 - \frac{1}{1+\exp\left(-\dot{x}/v_d\right)}$ and

$$g(x) = \begin{cases} x^2/r^2 - 2x/r + 1, & x \leq r \\ 0, & x > r \end{cases} \tag{20}$$

where x is the Euclidean distance between the capsule witness points and $\dot{x}$ is its rate of change. $k_p[m/s^2]$ and $k_d[s^{-1}]$ are repulsion and damping gains. The parameter μ is used to specify the priority of the collision RMP relative to other RMPs and $r[m]$ is used to control at what distance the collision RMP is disabled. $l_p[m]$, $l_m[m]$ and $v_d[m/s]$ are scaling parameters. ϵ_m and ϵ_d are offset parameters.

D. State estimation

Together with the robot's kinematic model, joint encoder and IMU data are used to estimate the floating base and CoM states. The estimated floating base position and velocity are used for feedback control. However, the CoM velocity estimate is too noisy for foot-step planning. Thus, in this paper, similar to [2], [3], an external MoCap system is used to estimate the base velocity based on an LED attached to the robot's body which is used as an approximate estimate of the CoM velocity. We carefully chose the base-frame position so that this approximation is valid.

IV. EXPERIMENTAL RESULTS AND ASSESSMENT

A. Robustness analysis in dynamics simulation

In this section, we validate the proposed locomotion controller in a kinodynamically faithful simulation of our point-foot biped robot Pat. For these simulations, we employ the multi-DoF rigid-body dynamics simulator from MIT's Biomimetic Robotics Lab, which notably incorporates actuator dynamics [29]. The controller's task is to stabilize this highly underactuated and unstable robot while avoiding collision between the robot's links, even under external disturbance. In all experiments we define the following task hierarchy in descending order of priority: contact, body orientation, body position, RMP swing leg. The MPC horizon is set to one gait period, which is 600 ms.

1) Experimental setup: In this work, we only consider the relevant body-segment collision pairs to reduce computation time. Because of Pat's morphology, the risk of collision almost exclusively exists between the two lower limbs while it is standing. Thus in all experiments, we only consider this collision pair. We approximate the geometry of the robot's lower limbs with a capsule of radius 15mm. The combined task projection, RMPflow, and whole body QP computations take less than 0.2 ms on an Intel i9-12900K CPU.

2) Push Disturbance Resistance Test: In order to test the effectiveness of the proposed controller, we run extensive push disturbance experiments. In particular, disturbance forces ranging between 10-100N that lie on the transverse plane are applied on the robot's base for 20ms at four instances—two swing phases and two double stance phases . Failure is reported if the base of the robot is below 25 cm or collision is detected between the two limbs during the subsequent 3 seconds after the disturbance is applied.

Our proposed strategy allows the robot to safely perform leg crossing movements by employing the self-collision-avoidance controller described in III-C. Thus, it takes advantage of the entire kinematically reachable region to recover from the disturbance. We benchmark the proposed strategy with a baseline conservative foot placement strategy that artificially restricts the lateral stepping region to avoid leg crossing movements used in [2], [4], [23]. Fig. 3 shows the

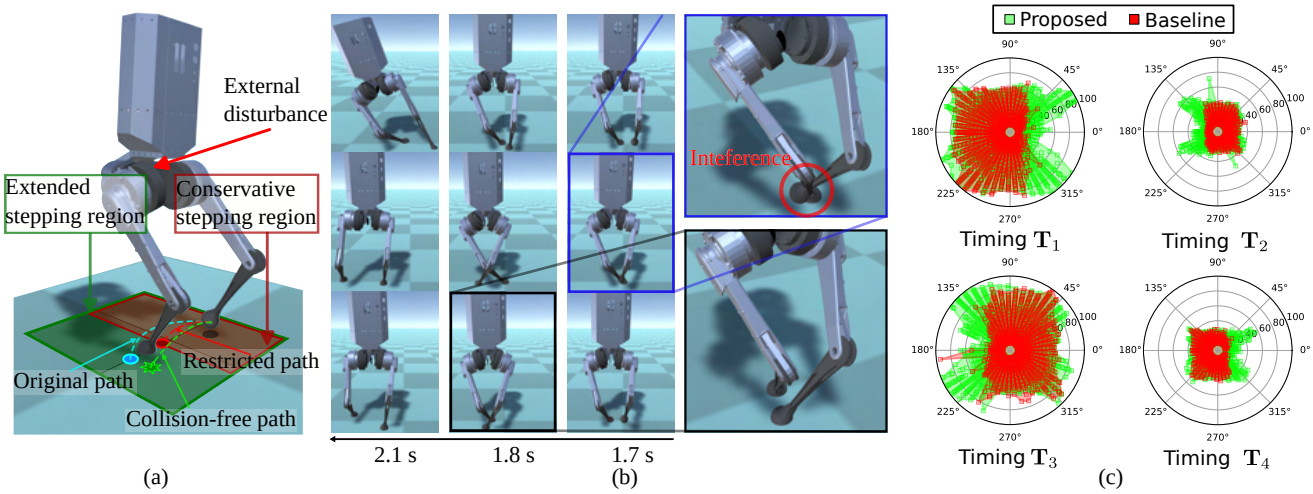

Fig. 3. **Push recovery simulation results** (a) The green and red boxes represent the valid stepping region of the proposed strategy and baseline strategies, respectively. The red and green dashed lines represent the foot trajectory under the baseline and proposed strategy in response to the external disturbance. (b) Snapshots of the robot trajectory under three scenarios. First row (baseline strategy), second row: leg crossing movement without collision avoidance, bottom row (proposed strategy): safe leg crossing movement with collision avoidance. (c) polar coordinate representation of the the applied disturbance forces. Successful outcomes are shown for the proposed (green) and baseline (red) strategies at four different disturbance timings. $\mathbf{T}_1$ and $\mathbf{T}_3$ correspond to the double stance phase and $\mathbf{T}_2$ and $\mathbf{T}_4$ correspond to the right swing and left swing phases. Note that failed trials are not represented in these plots.

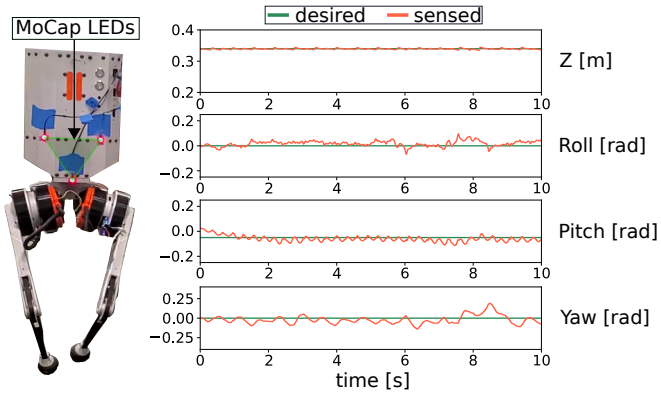

Fig. 4. **Hardware experiment result**. Floating-base reference tracking performance during unassisted in-place walking test.

results of simulation experiment. Fig. 3(a) shows the stepping regions of the proposed and baseline strategies, which are the extended and conservative stepping regions, respectively. Fig. 3(b) depicts the snapshots of the trajectories of the robot state in reaction to a lateral disturbance force $\mathbf{F}_y = -50$N. The first row depicts the robot state trajectory when the baseline strategy is used. Because it could not take the necessary stabilizing step, the robot could not recover from the disturbance and eventually fell. In the second row, the robot is shown performing leg crossing movements to recover from the disturbance. However, as is highlighted in the figure, collision instances were recorded along the way. The last row shows the robot under the proposed strategy. Under this strategy, the robot performed the leg crossing movement safely with the help of a collision avoidance controller and was able to reject the disturbance successfully. Fig.3(c) and Table I summarize the results of 10,000 experiments. Fig. 3(c) shows that the proposed strategy can recover from a larger range of disturbance forces by utilizing its extended

	$\eta_{b\|p}$	$\eta_{p\|b}$	Baseline SR	Proposed SR	Improvement
$\mathbf{T_1}$	68.06%	93.55%	47.76%	**65.64%**	37.43%
$\mathbf{T_2}$	60.56%	93.02%	18.36%	**28.2%**	53.59%
$\mathbf{T_3}$	68.78%	90.87%	49.56%	**65.48%**	32.12%
$\mathbf{T_4}$	64.46%	91.61%	19.56%	**27.8%**	42.12%

TABLE I

SUCCESS RATIO AND RATE COMPARISON.

stepping region in both double stance ($\mathbf{T}_1$ and $\mathbf{T}_3$) and swing phases ($\mathbf{T}_2$ and $\mathbf{T}_4$). Table I shows the success rate (SR) and conditional ratios $\eta_{p|b}$ and $\eta_{b|p}$. Where $\eta_{p|b}$ is the ratio of the number of successes of both strategies with the number of success of the baseline and $\eta_{b|p}$ is the ratio of the number of successes in both strategies with the number of success with the proposed strategy. In more than 90% of cases, the proposed controller can recover from disturbances the baseline strategy recovered from, but the baseline can only recover from less than 70% of the cases the proposed strategy recovered from. Moreover, comparing their success rates shows up to 53% improvement from the baseline strategy.

B. High-speed self-collision avoidance

In order to assess the efficacy of different reactive controller-based methods under high-speed operations, we performed self-collision avoidance tests using a simulation of the MIT Mini-Cheetah quadruped robot [26]. During these experiments, the robot was inverted and tasked with tracking limb trajectories that involved numerous self-collision phases while trying to maintain a fixed body position, as illustrated in Figure 5. Each cycle of the maneuver, from (a) to (d) and back in Figure 5, lasted 0.05 seconds, and the entire sequence of motion was sustained for five seconds. We evaluated the effectiveness of three distinct approaches: a conventional Artificial Potential Field (APF) method that relies solely on repulsive force, a Control Barrier Function

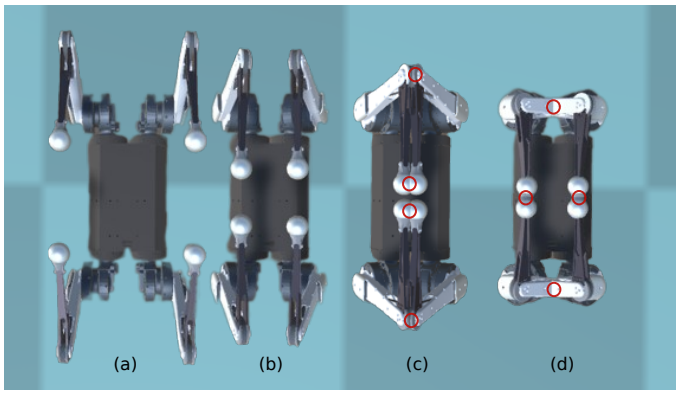

Fig. 5. High-speed self-collision avoidance test conducted with the MIT Mini-Cheetah. This figure presents a top-down view of the Mini-Cheetah robot, inverted and performing a leg-crossing maneuver. The cycle of movement, from (a) through (d) and back, is executed repetitively. Illustrated here is the scenario in which the collision avoidance feature is deactivated. Red circles highlight the specific locations of collision points.

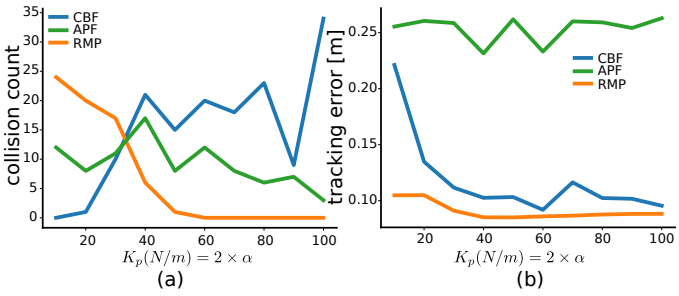

Fig. 6. Comparison of high-speed self-collision avoidance test results using APF, CBF [16], and our proposed approach. (a) illustrates the variation in the count of collision instances as we adjust the α parameter of CBF and the K_p parameter of APF and RMP. (b) presents the mean total tracking error, measured as the Euclidean distance between the desired limb trajectory and the actual trajectory throughout the entire motion period. This mean value represents the average tracking error across all four limbs.

(CBF) optimization approach as elaborated in [16], and our Riemannian Motion Policy (RMP)-based strategy. Both the APF and CBF methodologies were implemented in accordance with the procedures outlined in [16]. To ensure a balanced comparison, we administered the tests with a variety of parameter configurations. Specifically, we assessed different settings of the α parameter in the CBF-based method [16]. This parameter was subsequently doubled to obtain the value of the proportional gain K_p that was subsequently employed to compute the repulsive force in both the APF and RMP methods. As illustrated in Figure 6, the baseline APF-based approach struggles with self-collision avoidance and exhibits poor tracking performance. The CBF method, despite its proficiency at avoiding collisions with low settings of the α parameter, unfortunately, falls short in tracking performance when compared to the RMP-based method. Contrasting these, the RMP-based approach demonstrates an effective balance in maintaining a low tracking error while concurrently mitigating the risk of self-collision, outperforming the aforementioned alternative strategies.

C. Experimental validation of point-foot biped

A preliminary unassisted in-place stepping test is conducted to validate the viability of the controller and the robot's hardware. In this experiment, the robot was supported by a person for the first few steps and let go afterward. The framework summarized in Fig. 2 is used for the hardware experiments. In addition to the proposed controller, kinematics-based WBC is used to improve foot placement accuracy [2], [24]. In this experiment, collision avoidance RMP is not used as the experimental platform is not yet mature enough to conduct the aggressive leg crossing movements.

Fig.4 depicts tracking performance during the stepping test and shows that the controller can follow the commanded floating base references closely. Note that, even though we only do yaw control during the 0.024 s long dual support phases, Pat can still track the commanded zero yaw. In this work, we achieved up to forty unassisted steps and a video recording of the experiments can be found in the attached supplementary material.

V. CONCLUSION AND DISCUSSION

This paper proposes a new formulation for integrating the RMPflow computational framework with a nullspace projection-based whole-body controller. Based on the proposed formulation, a collision-avoidance swing-leg controller was designed and validated in simulation. Moreover, we presented a new small-scale point-foot biped robot purpose-built for experimental studies of dynamic biped locomotion. The hardware experiment results presented in this paper show the robot's viability but also indicate room for improvement. In the future, we plan to replicate the push recovery experiments conducted in simulation on the physical robot.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 2220924. We also express our gratitude to Naver Labs and MIT Biomimetic Robotics Lab for providing the Mini-cheetah robot as a research platform for conducting dynamic motion studies on legged robots.

REFERENCES

[1] J. Pratt and R. Tedrake, *Velocity-Based Stability Margins for Fast Bipedal Walking*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 299–324. [Online]. Available: https://doi.org/10.1007/978-3-540-36119-0_14

[2] D. Kim, S. J. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 936–956, Jun. 2020. [Online]. Available: https://doi.org/10.1177/0278364920918014

[3] D. Kim, Y. Zhao, G. Thomas, B. R. Fernandez, and L. Sentis, "Stabilizing series-elastic point-foot bipeds using whole-body operational space control," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1362–1379, 2016.

[4] D. Kim, G. Thomas, and L. Sentis, "Continuous cyclic stepping on 3d point-foot biped robots via constant time to velocity reversal," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*. IEEE, Dec. 2014. [Online]. Available: https://doi.org/10.1109/icarcv.2014.7064561

[5] N. D. Ratliff, J. Issac, and D. Kappler, "Riemannian motion policies," *CoRR*, vol. abs/1801.02854, 2018. [Online]. Available: http://arxiv.org/abs/1801.02854

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[7] I. Dovgopolik, K. Artemov, S. Zabihifar, A. Semochkin, and S. Kolyubin, "Fast and memory-efficient planning in c-space: Modified bi-directional rrt* algorithm for humanoid robots," in *2021 International Conference "Nonlinearity, Information and Robotics" (NIR)*, 2021, pp. 1–6.

[8] J. Qi, Q. Yuan, C. Wang, X. Du, F. Du, and A. Ren, "Path planning and collision avoidance based on the $RRT*FN$ framework for a robotic manipulator in various scenarios," *Complex Intelligent Systems*, Jul. 2023. [Online]. Available: https://doi.org/10.1007/s40747-023-01131-2

[9] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, Jul. 2015. [Online]. Available: https://doi.org/10.1007/s10514-015-9479-3

[10] A. El Khoury, F. Lamiraux, and M. Taïx, "Optimal motion planning for humanoid robots," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3136–3141.

[11] Y. Liu, F. Zha, M. Li, W. Guo, Y. Jia, P. Wang, Y. Zang, and L. Sun, "Creating better collision-free trajectory for robot motion planning by linearly constrained quadratic programming," *Frontiers in Neurorobotics*, vol. 15, Aug. 2021. [Online]. Available: https://doi.org/10.3389/fnbot.2021.724116

[12] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 1778–1784 vol.3.

[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[14] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, Jun. 2013. [Online]. Available: https://doi.org/10.15607/rss.2013.ix.031

[15] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, Jan. 2021. [Online]. Available: https://doi.org/10.1109/lcsys.2020.3000748

[16] C. Khazoom, D. Gonzalez-Diaz, Y. Ding, and S. Kim, "Humanoid self-collision avoidance using whole-body control with control barrier functions," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 558–565.

[17] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmp flow: A computational graph for automatic motion policy generation," in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 2020, pp. 441–457.

[18] M. Mukadam, C.-A. Cheng, D. Fox, B. Boots, and N. Ratliff, "Riemannian motion policy fusion through learnable lyapunov function reshaping," in *Conference on robot learning*. PMLR, 2020, pp. 204–219.

[19] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, and N. Ratliff, "Learning reactive motion policies in multiple task spaces from human demonstrations," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 1457–1468. [Online]. Available: https://proceedings.mlr.press/v100/rana20a.html

[20] A. Li, M. Mukadam, M. Egerstedt, and B. Boots, "Multi-objective policy generation for multi-robot systems using riemannian motion policies," in *Robotics Research*, T. Asfour, E. Yoshida, J. Park, H. Christensen, and O. Khatib, Eds. Cham: Springer International Publishing, 2022, pp. 258–274.

[21] B. Wingo, C.-A. Cheng, M. Murtaza, M. Zafar, and S. Hutchinson, "Extending riemmanian motion policies to a class of underactuated wheeled-inverted-pendulum robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3967–3973.

[22] M. Zafar and H. I. Christensen, "Whole body control of a wheeled inverted pendulum humanoid," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 89–94.

[23] D. Kim, J. Lee, J. Ahn, O. Campbell, H. Hwang, and L. Sentis, "Computationally-Robust and Efficient Prioritized Whole-Body Controller with Contact Constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[24] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *CoRR*, vol. abs/1909.06586, 2019. [Online]. Available: http://arxiv.org/abs/1909.06586

[25] 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)*, 2018, pp. 1–9.

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

[27] "Tmotor actuators," https://store.tmotor.com/category.php?id=91, accessed: 2022-09-12.

[28] M. Safeea, P. Neto, and R. Bearee, "Efficient calculation of minimum distance between capsules and its use in robotics," *IEEE Access*, vol. 7, pp. 5368–5373, 2019.

[29] 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)*, 2021, pp. 1–8.