**Integration of Riemannian Motion Policy and Whole-Body Control for Dynamic Legged Locomotion**

Daniel Marew 1, Misha Lvovsky1, Shangqun Yu1, Shotaro Sessions1, and Donghyun Kim1

**Abstract**—In this paper, we present a novel 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 by applying a virtual repulsive force on the robot to push it away from obstacles. However, these techniques are hard to tune, often making it suitable for fast and real-time operations. Moreover, conflicts between these constraints may render the optimization problem infeasible in some instances.

**I. INTRODUCTION**

In dynamic legged locomotion, the size of 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.

Traditionally, artificial potential field (APF)-based approaches have been the go-to technique for designing real-time collision avoidance behaviors. APF-based methods accomplish collision avoidance behavior by applying a virtual repulsive force on the robot to push it away from obstacles [6], [7]. However, these techniques are hard to tune, often ignoring the full-body dynamics, and frequently produce undesired behaviors because of local minima. An alternative approach is to use constrained optimization [8]–[11]. However, these approaches tend not to scale as well as the number of collision elements considered increases since every added collision element introduces a constraint on the optimization problem, potentially making them impractical for fast and real-time applications. Moreover, conflicts between these constraints may render the optimization problem infeasible in some instances [11].

In this work, we, for the first time, extend it to the domain of dynamic legged locomotion, which have unforgiving under-actuation and 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.

![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.](image-url)
the task. The RMPflow algorithm fuses these local RMPs into a single configuration-space (C-space) RMP using an operator called the pullback operator.

One caveat of RMPflow is its implicit assumption that the C-space acceleration policy obtained through the pullback operation 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], [12]–[16], with one notable exception [17]. Wingo et al. [17] 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 [17], [18]. 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 [17] relies on. This underactuation, however, differs from the underactuation of legged systems whose floating base can not be directly controlled [18]. Thus the RMPflow framework developed in [17] 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 [17] whose optimal solution, a 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. Researchers have successfully demonstrated dynamic biped locomotion with passive ankle, line-foot contact, robots [2], [19], [20]. The line-foot contact on these robots constrains the yaw directional motion, which makes the balance control problem relatively easier than point-foot bipeds, whose underactuation is unforgiving. To our knowledge, there have only been two examples of unassisted 3D point-foot biped locomotion [3], [21]. Kim et al. [3] achieved up to 18 steps using a whole-body operational space control approach, and there is video evidence from the robotics company Apptronik [21], although this work appears unpublished. In this work, we demonstrate Pat successfully performing unassisted dynamic balancing for more than 40 steps, which shows its dynamic motion control capability and viability for future improvements.

In summary, our contributions are twofold: 1) a formulation for integrating RMPflow with a null-space projection-based whole-body controller. Through extensive simulation experiments, we demonstrate that a collision-avoidance swing-leg controller designed based on the proposed formulation can significantly improve a point-foot biped robot’s robustness against external disturbances. 2) a point-foot biped robot purpose-built for experimental studies of dynamic biped locomotion. We validate the proposed controller’s performance and the viability of the robot’s hardware by demonstrating 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

\[ A\ddot{q} + b + g = S_m^T \tau + J_q^T f_r, \]  

where \( A, b, \) and \( g \) are the generalized mass matrix, coriolis force, and gravitation forces, respectively. \( S_m \) is the actuated joint selection matrix. \( \tau, f_r, \) and \( J_q \) are joint torque, augmented reaction force and contact Jacobian, respectively. \( q \in \mathbb{R}^{6+n} \) is the configuration space acceleration where \( n \) the number of actuated degrees of freedom.

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 \( M \) is characterized, in its canonical form, by the tuple \( (a, M)^M \), where \( a : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m \) is an acceleration policy and \( M : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^{m \times m} \) is a state-dependent Riemannian metric. This tuple can be written in its natural form, \( (f, M)^M \), by using \( f = Ma \) where \( f \) 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 constitute 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, 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
where $(f_i, M_i)^{N_i}$ is the $i$-th child RMP, $J_i$ is a Jacobian matrix and $K$ is the number of child RMPs. Lastly, resolve is used to transform the computed natural form RMP, $(f, M)^M$ to its canonical form $(a, M)^M$ where $a = Mf$ 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 [22].

$$q_{cmd}^i = q_{cmd}^{i-1} + J_{dyn}^{i,pre} \left( \hat{x}_i - J_i q - J_i q_{cmd}^{i-1} \right), \quad (3)$$

where

$$J_{i,pre} = J_i N_i - 1,$$  \hspace{1cm} (4)

$$N_i - 1 = N_0 N_{i,0} \cdots N_{i-1,i-2},$$  \hspace{1cm} (5)

$$N_{i,i-1} = I - J_i J_i^{i,pre}. \hspace{1cm} (6)$$

Here, $i \geq 1$, and

$$q_{cmd}^0 = J_{dyn}^0 (-J_c \dot{q}). \hspace{1cm} (7)$$

$x_{cmd}^i$ is the acceleration policy of $i$-th task defined by the PD controller.

$$x_{cmd}^i(x, \dot{x}) = \dot{x}_{des} + K_p(x_{des} - x) + K_d(\ddot{x}_{des} - \ddot{x}), \hspace{1cm} (8)$$

where $K_p$ and $K_d$ are position and velocity feedback gains, respectively. $J_{i,pre}$ is the projection of the $i$-th task Jacobian into the null space of the prior tasks. $J_c$ is a contact Jacobian and the dynamically consistent pseudo-inverse is denoted by an overline and superscript ‘dyn’ as in [23] and defined as

$$\overline{J}^{dyn} = A^{-1} J^T \Lambda. \hspace{1cm} (9)$$

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

$$\Lambda = (JA^{-1}J^T)^{-1}. \hspace{1cm} (10)$$

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 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. So do we, define the final C-space acceleration command in the following way.

$$q_{cmd} = q_{cmd}^k + N_k q_{cmd}^k \hspace{1cm} (11)$$

where $q_{cmd}^k$ and $N_k$ 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 [III-B]. $q_{cmd}^k$ is any arbitrary C-space acceleration that can be used to realize RMPs. We obtain the optimal C-space acceleration command by solving the following constrained weighted least-squares problem.

$$\min \sum_{i=0}^{n} \left| J_i q_{cmd}^i + J_i q - \hat{x}_i \right|^2_{M_i}, \hspace{1cm} s.t. \hspace{1cm} q_{cmd}^i = q_{cmd} + N_k q_{cmd}^k \hspace{1cm} (12)$$

where $q_{cmd}^k$ is any arbitrary $q_{cmd}^k = \arg \min q_{cmd}^k \sum_{i=0}^{n} \left| J_i N_k q_{cmd}^k - \hat{x}_i \right|^2_{M_i} \hspace{1cm} (13)$

The analytical solution to (12) yields the modified pullback operation given by (13)-(14).

$$\begin{align*}
M_{rmp} &= N_k \left( \sum_{i=0}^{n} J_i M_i J_i \right) N_k \hspace{1cm} (13) \\
f_{rmp} &= N_k \left( \sum_{i=0}^{n} J_i (f_i - M_i J_i \dot{q}) - J_i M_i J_i \dot{q}_k \right) \hspace{1cm} (14)
\end{align*}$$

where $M_{rmp}$ and $f_{rmp}$ are the the projected virtual inertia metric and virtual force computed through the modified pullback operations in (13) and (14) respectively. The term inside the bracket in (13) and the first term in (14) are the regular RMFlow pullback operations in (2). Thus $M_{rmp}$, $f_{rmp}$ can be obtained from the C-space RMP $(M, f)^M$.

$$\begin{align*}
M_{rmp} &= N_k M N_k \hspace{1cm} f_{rmp} = N_k f - M q_{cmd}^k \hspace{1cm} (15)
\end{align*}$$

The final acceleration command is then obtained by

$$q_{cmd} = q_{cmd} + N_k M_{rmp} f_{rmp} \hspace{1cm} (16)$$

**D. Whole-body control (WBC)**

In this work, we execute the acceleration command from (16) using a convex MPC [24] and whole body impulsive control (MPC + WBIC) based controller proposed in [23]. In [23], 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 \(f_r\) and \(\ddot{q}\) are obtained by solving the QP in \((17)\), inverse dynamics is used to compute the torque command.

\[
\begin{align*}
\min_{\delta f_r, \dot{f}_r} & \quad \delta f_r^T Q_r \delta f_r + \dot{f}_r^T Q_2 \dot{f}_r \\
\text{s.t.} \quad & S_f (A\ddot{q} + b + g) = S_f J_c^T \dot{f}_r \quad \text{(floating base dyn.)} \\
& \ddot{q} = \ddot{q}_{cmd} + \begin{bmatrix} \delta f_r \\ 0_n \end{bmatrix} \quad \text{(acceleration)} \\
& \dot{f}_r = f_r^{MPC} + \delta f_r \quad \text{(reaction forces)} \\
& W f_r \geq 0, \quad \text{(contact force constraints)}
\end{align*}
\]

where \(f_r^{MPC}\) is the reaction force command computed by the MPC, and \(S_f\) is the floating base selection matrix. \(J_c\) and \(W\) are the augmented contact Jacobian and contact constraint matrix respectively. \(\delta f_r\) and \(\dot{f}_r\) 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 [25] 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 [26], which consist of thin, large-diameter out-runner motors, a motor controller based on the open source Mini-Cheetah motor controller [25], 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 \(\frac{11}{7} \approx 1.5\). 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 \((2)\). Note that the attractor RMP used in this work is not designed to meet the definition of a geometric dynamical system (GDS) required by [12] 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 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 [27] 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 [12].

\[
\ddot{x}(x, \dot{x}) = k_p \exp (-x/l_p) - k_d \frac{\sigma(\dot{x}) \dot{x}}{x/l_d + \epsilon_d} \quad \text{(18)}
\]

\[
m(x, \dot{x}) = \sigma(\dot{x}) g(x) \frac{\mu}{x/l_m + \epsilon_m} \quad \text{(19)}
\]

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

\[
g(x) = \begin{cases} 
x^2/\nu^2 - 2x/r + 1, & x \leq r \\
0, & x > r
\end{cases}
\]
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 \( \mu \) 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. \( \epsilon_m \) and \( \epsilon_d \) 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. 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 [2]. 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. Fig. 3 shows the 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 \( F_y = -50N \). 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 stepping region in both double stance (T₁ and T₂) and swing phases (T₂ and T₃). Table I shows the success rate (SR) and conditional ratios \( \eta_{pl.b} \) and \( \eta_{hp.p} \). Where \( \eta_{pl.b} \) is the ratio of the number of successes of both strategies with the number of success of the baseline and \( \eta_{hp.p} \) 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. Experimental validation

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], [23]. 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.

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.
