RMP*flow*: A Computational Graph for Automatic Motion Policy Generation

Ching-An Cheng[1,2], Mustafa Mukadam[1,2], Jan Issac[1], Stan Birchfield[1], Dieter Fox[1,3], Byron Boots[1,2], and Nathan Ratliff[1]

[1] NVIDIA, Seattle Robotics Lab, Seattle, WA, USA
[2] Georgia Institute of Technology, Robot Learning Lab, Atlanta, GA, USA
[3] University of Washington, Robotics and State Estimation Lab, Seattle, WA, USA

Abstract. We develop a novel policy synthesis algorithm, RMPflow, based on geometrically consistent transformations of Riemannian Motion Policies (RMPs). RMPs are a class of reactive motion policies designed to parameterize non-Euclidean behaviors as dynamical systems in intrinsically nonlinear task spaces. Given a set of RMPs designed for individual tasks, RMPflow can consistently combine these local policies to generate an expressive global policy, while simultaneously exploiting sparse structure for computational efficiency. We study the geometric properties of RMPflow and provide sufficient conditions for stability. Finally, we experimentally demonstrate that accounting for the geometry of task policies can simplify classically difficult problems, such as planning through clutter on high-DOF manipulation systems.

Keywords: Motion and Path Planning, Collision Avoidance, Dynamics

1 Introduction

In this work, we develop a new motion generation and control framework that enables globally stable controller design within *intrinsically*[4] non-Euclidean spaces. Non-Euclidean geometries are not often modeled explicitly in robotics, but are nonetheless common in the natural world. One important example is the apparent non-Euclidean behavior of obstacle avoidance. Obstacles become holes in this setting. As a result, straight lines are no longer a reasonable definition of shortest distance—geodesics must, therefore, naturally flow around them. This behavior implies a form of non-Euclidean geometry: the space is naturally curved by the presence of obstacles.

The planning literature has made substantial progress in modeling non-Euclidean task-space behaviors, but at the expense of efficiency and reactivity. Starting with early differential geometric models of obstacle avoidance [1] and building toward modern planning algorithms and optimization techniques [2–9], these techniques can calculate highly nonlinear trajectories. However, they are often computationally intensive, sensitive to noise, and unresponsive to perturbation. In addition, the internal nonlinearities of robots due to kinematic constraints are sometimes simplified in the optimization.

[4] An intrinsically non-Euclidean space is one which is defined by a non-constant Riemannian metric with non-trivial curvature.

At the same time, a separate thread of literature, emphasizing fast reactive control over computationally expensive planning, developed efficient closed-loop control techniques such as Operational Space Control (OSC) [10]. But while these techniques account for internal geometries from the robot's kinematic structure, they assume simple Euclidean geometry in task spaces [11, 12], failing to provide a complete treatment of the external geometries. As a result, obstacle avoidance, e.g., has to rely on *extrinsic* potential functions, leading to undesirable deacceleartion behavior when the robot is close to the obstacle. If the non-Euclidean geometry can be *intrinsically* considered, then fast obstacle avoidance motion would naturally arise as traveling along the induced geodesic. The need for a holistic solution to motion generation and control has motivated a number of recent system architectures tightly integrating planning and control [13, 14].

We develop a new approach to synthesizing control policies that can accommodate and leverage the modeling capacity of intrinsically non-Euclidean robotics tasks. Taking inspiration from Geometric Control Theory [15],[5] we design a novel recursive algorithm, RMPflow, based on a recently proposed mathematical object for representing nonlinear policies known as the Riemannian Motion Policy (RMP) [16]. This algorithm enables the geometrically consistent fusion of many component policies defined across non-Euclidean task spaces that are related through a tree structure. We show that RMPflow, which generates behavior by calculating how the robot should accelerate, mimics the Recursive Newton-Euler algorithm [17] in structure, but generalizes it beyond rigid-body systems to a broader class of highly-nonlinear transformations and spaces.

In contrast to existing frameworks, our framework naturally models non-Euclidean task spaces with Riemannian metrics that are not only configuration dependent, but also *velocity* dependent. This allows RMPflow to consider, e.g., the *direction* a robot travels to define the importance weights in combing policies. For example, an obstacle, despite being close to the robot, can usually be ignored if robot is heading away from it. This new class of policies leads to an extension of Geometric Control Theory, building on a new class of non-physical mechanical systems we call Geometric Dynamical Systems (GDS).

We also show that RMPflow is Lyapunov-stable and coordinate-free. In particular, when using RMPflow, robots can be viewed each as different parameterizations of the same task space, defining a precise notion of behavioral consistency between robots. Additionally, under this framework, the implicit curvature arising from non-constant Riemannian metrics (which may be roughly viewed as position-velocity dependent inertia matrices in OSC) produces nontrivial and intuitive policy contributions that are critical to guaranteeing stability and generalization across embodiments. Our experimental results illustrate how these curvature terms can be impactful in practice, generating nonlinear geodesics that result in curving or orbiting around obstacles. Finally, we demonstrate the utility of our framework with a fully reactive real-world system on multiple dual-arm manipulation problems.

[5] See Appendix A.1 for a discussion of why geometric mechanics and geometric control theory constitute a good starting point.

2 Motion Generation and Control

Motion generation and control can be formulated as the problem of transforming curves from the configuration space $\mathcal{C}$ to the task space $\mathcal{T}$. Specifically, let $\mathcal{C}$ be a d-dimensional smooth manifold. A robot's motion can be described as a curve $q : [0, \infty) \to \mathcal{C}$ such that the robot's configuration at time t is a point $q(t) \in \mathcal{C}$. Without loss of generality, suppose $\mathcal{C}$ has a global coordinate $\mathbf{q} : \mathcal{C} \to \mathbb{R}^d$, called the *generalized coordinate*; for short, we would identify the curve q with its coordinate and write $\mathbf{q}(q(t))$ as $\mathbf{q}(t) \in \mathbb{R}^d$. A typical example of the generalized coordinate is the joint angles of a d-DOF (degrees-of-freedom) robot: we denote $\mathbf{q}(t)$ as the joint angles at time t and $\dot{\mathbf{q}}(t)$, $\ddot{\mathbf{q}}(t)$ as the joint velocities and accelerations. To describe the tasks, we consider another manifold $\mathcal{T}$, the task space, which is related to the configuration space $\mathcal{C}$ through a smooth *task map* $\psi : \mathcal{C} \to \mathcal{T}$. The task space $\mathcal{T}$ can be the end-effector position/orientation [10, 18], or more generally can be a space that describes whole-body robot motion, e.g., in simultaneous tracking and collision avoidance [19, 20]. The goal of motion generation and control is to design the curve q so that the transformed curve $\psi \circ q$ exhibits desired behaviors on the task space $\mathcal{T}$.

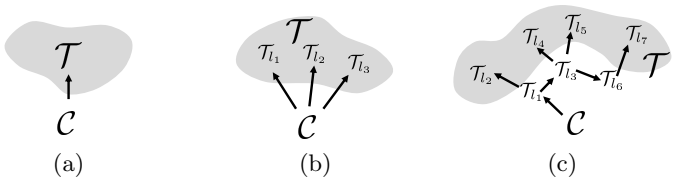
Notation For clarity, we use boldface to distinguish the coordinate-dependent representations from abstract objects; e.g. we write $q(t) \in \mathcal{C}$ and $\mathbf{q}(t) \in \mathbb{R}^d$. In addition, we will often omit the time- and input-dependency of objects unless necessary; e.g. we may write $q \in \mathcal{C}$ and $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. For derivatives, we use both symbols ∇ and ∂, with a transpose relationship: for $\mathbf{x} \in \mathbb{R}^m$ and a differential map $\mathbf{y} : \mathbb{R}^m \to \mathbb{R}^n$, we write $\nabla_{\mathbf{x}}\mathbf{y}(\mathbf{x}) = \partial_{\mathbf{x}}\mathbf{y}(\mathbf{x})^\top \in \mathbb{R}^{m\times n}$. For a matrix $\mathbf{M} \in \mathbb{R}^{m\times m}$, we denote $\mathbf{m}_i = (\mathbf{M})_i$ as its ith column and $M_{ij} = (\mathbf{M})_{ij}$ as its (i, j) element. To compose a matrix, we use $(\cdot)\vdots$ for vertical (or matrix) concatenation and $[\cdot]\vdots$ for horizontal concatenation. For example, we write $\mathbf{M} = [\mathbf{m}_i]_{i=1}^m = (M_{ij})_{i,j=1}^m$ and $\mathbf{M}^\top = (\mathbf{m}_i^\top)_{i=1}^m = (M_{ji})_{i,j=1}^m$. We use $\mathbb{R}_+^{m\times m}$ and $\mathbb{R}_{++}^{m\times m}$ to denote the symmetric, positive semi-definite/definite matrices, respectively.

2.1 Motion Policies and the Geometry of Motion

We model motion as a second-order differential equation[6] of $\ddot{\mathbf{q}} = \pi(\mathbf{q}, \dot{\mathbf{q}})$, where we call π a *motion policy* and $(\mathbf{q}, \dot{\mathbf{q}})$ the *state*. In contrast to an open-loop trajectory, which forms the basis of many motion planners, a motion policy expresses the entire continuous collection of its integral trajectories and therefore is robust to perturbations. Motion policies can model many adaptive behaviors, such as reactive obstacle avoidance [21, 13] or responses driven by planned Q-functions [22], and their second-order formulation enables rich behavior that cannot be realized by the velocity-based approach [23].

The geometry of motion has been considered by many planning and control algorithms. Geometrical modeling of task spaces is used in topological motion planning [3], and motion optimization has leveraged Hessian to exploit the natural geometry of costs [24, 5, 25, 26]. Ratliff et al. [2], e.g., use the workspace ge-

[6] We assume the system has been feedback linearized. A torque-based setup can be similarly derived by setting the robot inertia matrix as the intrinsic metric on $\mathcal{C}$ [11].



(a) (b) (c)

Fig. 1: Tree-structured task maps

ometry inside a Gauss-Newton optimizer and generate natural obstacle-avoiding reaching motion through traveling along geodesics of curved spaces.

Geometry-aware motion policies were also developed in parallel in controls. OSC is the best example [10]. Unlike the planning approaches, OSC focuses on the internal geometry of the robot and considers only simple task-space geometry. It reshapes the workspace dynamics into a simple spring-mass-damper system with a constant inertia matrix, enforcing a form of Euclidean geometry in the task space. Variants of OSC have been proposed to consider different metrics [27, 11, 20], task hierarchies [19, 28], and non-stationary inputs [29].

While these algorithms have led to many advances, we argue that their isolated focus on either the internal or the external geometry limits the performance. The planning approach fails to consider reactive dynamic behavior; the control approach cannot model the effects of velocity dependent metrics, which are critical to generating sensible obstacle avoidance motions, as discussed in the introduction. While the benefits of velocity dependent metrics was recently explored using RMPs [16], a systematic understanding is still an open question.

3 Automatic Motion Policy Generation with RMPflow

RMPflow is an efficient manifold-oriented computational graph for automatic generation of motion policies. It is aimed for problems with a task space $\mathcal{T} = \{\mathcal{T}_{l_i}\}$ that is related to the configuration space $\mathcal{C}$ through a tree-structured task map ψ, where $\mathcal{T}_{l_i}$ is the ith subtask. Given user-specified motion policies $\{\pi_{l_i}\}$ on $\{\mathcal{T}_{l_i}\}$ as RMPs, RMPflow is designed to *consistently* combine these subtask policies into a global policy π on $\mathcal{C}$. To this end, RMPflow introduces 1) a data structure, called the *RMP-tree*, to describe the tree-structured task map ψ and the policies, and 2) a set of operators, called the *RMP-algebra*, to propagate information across the RMP-tree. To compute $\pi(\mathbf{q}(t), \dot{\mathbf{q}}(t))$ at time t, RMPflow operates in two steps: it first performs a *forward pass* to propagate the state from the root node (i.e. $\mathcal{C}$) to the leaf nodes (i.e. $\{\mathcal{T}_{l_i}\}$); then it performs a *backward pass* to propagate the RMPs from the leaf nodes to the root node. These two steps are realized by recursive use of RMP-algebra, exploiting shared computation paths arising the tree structure to maximize efficiency.

3.1 Structured Task Maps

In most cases, the task-space manifold $\mathcal{T}$ is structured. In this paper, we consider the case where the task map ψ can be expressed through a tree-structured composition of transformations $\{\psi_{e_i}\}$, where ψ_{e_i} is the ith transformation. Fig. 1 illustrates some common examples. Each node denotes a manifold and each edge

denotes a transformation. This family trivially includes the unstructured task space $\mathcal{T}$ (Fig. 1a) and the product manifold $\mathcal{T} = \mathcal{T}_{l_1} \times \cdots \times \mathcal{T}_{l_K}$ (Fig. 1b), where K is the number of subtasks. A more interesting example is the kinematic tree (Fig. 1c), where, e.g., the subtask spaces on the leaf nodes can describe the tracking and obstacle avoidance tasks along a multi-DOF robot.

The main motivation of explicitly handling the structure in the task map ψ is two-fold. First, it allows RMPflow to exploit computation shared across different subtask maps. Second, it allows the user to focus on designing motion policies for each subtask individually, which is easier than directly designing a global policy for the entire task space $\mathcal{T}$. For example, $\mathcal{T}$ may describe the problem of humanoid walking, which includes staying balanced, scheduling contacts, and avoiding collisions. Directly parameterizing a policy to satisfy all these objectives can be daunting, whereas designing a policy for each subtask is more feasible.

3.2 Riemannian Motion Policies (RMPs)

Knowing the structure of the task map is not sufficient for consistently combining subtask policies: we require some geometric information about the motion policies' behaviors [16]. Toward this end, we adopt an abstract description of motion policies, called RMPs [16], for the nodes of the RMP-tree. Specifically, let $\mathcal{M}$ be an m-dimensional manifold with coordinate $\mathbf{x} \in \mathbb{R}^m$. The *canonical form* of an RMP on $\mathcal{M}$ is a pair $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$, where $\mathbf{a} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is a continuous motion policy and $\mathbf{M} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times m}_+$ is a differentiable map. Borrowing terminology from mechanics, we call $\mathbf{a}(\mathbf{x}, \dot{\mathbf{x}})$ the *desired acceleration* and $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ the *inertia matrix* at $(\mathbf{x}, \dot{\mathbf{x}})$, respectively.[7] $\mathbf{M}$ defines the directional importance of $\mathbf{a}$ when it is combined with other motion policies. Later in Section 4, we will show that $\mathbf{M}$ is closely related to Riemannian metric, which describes how the space is stretched along the curve generated by $\mathbf{a}$; when $\mathbf{M}$ depends on the state, the space becomes *non-Euclidean*. We additionally introduce a new RMP form, called the *natural form*. Given an RMP in its canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$, the natural form is a pair $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, where $\mathbf{f} = \mathbf{M}\mathbf{a}$ is the *desired force* map. While the transformation between these two forms may look trivial, their distinction will be useful later when we introduce the RMP-algebra.

3.3 RMP-tree

The RMP-tree is the core data structure used by RMPflow. An RMP-tree is a directed tree, in which each node represents an RMP and its state, and each edge corresponds to a transformation between manifolds. The root node of the RMP-tree describes the global policy π on $\mathcal{C}$, and the leaf nodes describe the local policies $\{\pi_{l_i}\}$ on $\{\mathcal{T}_{l_i}\}$. To illustrate, let us consider a node u and its K child nodes $\{v_i\}_{i=1}^K$. Suppose u describes an RMP $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ and v_i describes an RMP $[\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i}$, where $\mathcal{N}_i = \psi_{e_i}(\mathcal{M})$ for some ψ_{e_i}. Then we write $u = ((\mathbf{x}, \dot{\mathbf{x}}), [\mathbf{f}, \mathbf{M}]^{\mathcal{M}})$ and $v_i = ((\mathbf{y}_i, \dot{\mathbf{y}}_i), [\mathbf{f}_i, \mathbf{M}_i]^{\mathcal{N}_i})$; the edge connecting u and v_i points from u to v_i along ψ_{e_i}. We will continue to use this example to illustrate how RMP-algebra propagates the information across the RMP-tree.

[7] Here we adopt a slightly different terminology from [16]. We note that $\mathbf{M}$ and $\mathbf{f}$ do not necessarily correspond to the inertia and force of a physical mechanical system.

3.4 RMP-algebra

The RMP-algebra consists of three operators (`pushforward`, `pullback`, and `resolve`) to propagate information.[8] They form the basis of the forward and backward passes for automatic policy generation, described in the next section.

1. `pushforward` is the operator to forward propagate the *state* from a parent node to its child nodes. Using the previous example, given $(\mathbf{x}, \dot{\mathbf{x}})$ from u, it computes $(\mathbf{y}_i, \dot{\mathbf{y}}_i) = (\psi_{e_i}(\mathbf{x}), \mathbf{J}_i(\mathbf{x})\dot{\mathbf{x}})$ for each child node v_i, where $\mathbf{J}_i = \partial_{\mathbf{x}}\psi_{e_i}$ is a Jacobian matrix. The name "pushforward" comes from the linear transformation of tangent vector $\dot{\mathbf{x}}$ to the image tangent vector $\dot{\mathbf{y}}_i$.
2. `pullback` is the operator to backward propagate the natural-formed RMPs from the child nodes to the parent node. It is done by setting $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ with

$$\mathbf{f} = \sum_{i=1}^{K} \mathbf{J}_i^\top (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}}) \qquad \text{and} \qquad \mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i \tag{1}$$

   The name "pullback" comes from the linear transformations of the cotangent vector (1-form) $\mathbf{f}_i - \mathbf{M}_i\dot{\mathbf{J}}_i\dot{\mathbf{x}}$ and the inertia matrix (2-form) $\mathbf{M}_i$. In summary, velocities can be pushfowarded along the direction of ψ_i, and forces and inertial matrices can be pullbacked in the opposite direction.
   To gain more intuition of `pullback`, we write `pullback` in the canonical form of RMPs. It can be shown that the canonical form $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ of the natural form $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ above is the solution to a least-squared problem:

$$\mathbf{a} = \arg\min_{\mathbf{a}'} \tfrac{1}{2} \sum_{i=1}^{K} \|\mathbf{J}_i\mathbf{a}' + \dot{\mathbf{J}}_i\dot{\mathbf{x}} - \mathbf{a}_i\|^2_{\mathbf{M}_i} \tag{2}$$

   where $\mathbf{a}_i = \mathbf{M}_i^\dagger \mathbf{f}_i$ and $\|\cdot\|^2_{\mathbf{M}_i} = \langle \cdot, \mathbf{M}_i \cdot \rangle$. Because $\ddot{\mathbf{y}}_i = \mathbf{J}_i\ddot{\mathbf{x}} + \dot{\mathbf{J}}_i\dot{\mathbf{x}}$, `pullback` attempts to find an $\mathbf{a}$ that can realize the desired accelerations $\{\mathbf{a}_i\}$ while trading off approximation errors with an importance weight defined by the inertia matrix $\mathbf{M}_i(\mathbf{y}_i, \dot{\mathbf{y}}_i)$. The use of state dependent importance weights is a distinctive feature of RMPflow. It allows RMPflow to activate different RMPs according to *both* configuration and velocity (see Section 3.6 for examples). Finally, we note that the `pullback` operator defined in this paper is slightly different from the original definition given in [16], which ignores the term $\dot{\mathbf{J}}_i\dot{\mathbf{x}}$ in (2). While ignoring $\dot{\mathbf{J}}_i\dot{\mathbf{x}}$ does not necessary destabilize the system [20], its inclusion is critical to implement consistent policy behaviors.
3. `resolve` is the last operator of RMP-algebra. It maps an RMP from its natural form to its canonical form. Given $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$, it outputs $(\mathbf{a}, \mathbf{M})^{\mathcal{M}}$ with $\mathbf{a} = \mathbf{M}^\dagger \mathbf{f}$, where $\dagger$ denotes Moore-Penrose inverse. The use of pseudo-inverse is because in general the inertia matrix is only positive semi-definite. Therefore, we also call the natural form of $[\mathbf{f}, \mathbf{M}]^{\mathcal{M}}$ the *unresolved form*, as potentially it can be realized by multiple RMPs in the canonical form.

3.5 Algorithm: Motion Policy Generation

Now we show how RMPflow uses the RMP-tree and RMP-algebra to generate a global policy π on $\mathcal{C}$. Suppose each subtask policy is provided as an RMP. First,

[8] Precisely it propagates the numerical values of RMPs and states at a particular time.

we construct an RMP-tree with the same structure as ψ, where we assign subtask RMPs as the leaf nodes and the global RMP $[\mathbf{f}_r, \mathbf{M}_r]^{\mathcal{C}}$ as the root node. With the RMP-tree specified, RMPflow can perform automatic policy generation. At every time instance, it first performs a forward pass: it recursively calls `pushforward` from the root node to the leaf nodes to update the state information in each node in the RMP-tree. Second, it performs a backward pass: it recursively calls `pullback` from the leaf nodes to the root node to back propagate the values of the RMPs in the natural form, and finally calls `resolve` at the root node to transform the global RMP $(\mathbf{f}_r, \mathbf{M}_r)^{\mathcal{C}}$ into its canonical form $(\mathbf{a}_r, \mathbf{M}_r)^{\mathcal{C}}$ for policy execution (i.e. setting $\pi(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{a}_r$).

The process of policy generation of RMPflow uses the tree structure for computational efficiency. For K subtasks, it has time complexity $O(K)$ in the worst case as opposed to $O(K \log K)$ of a naive implementation which does not exploit the tree structure. Furthermore, all computations of RMPflow are carried out using matrix-multiplications, except for the final `resolve` call, because the RMPs are expressed in the natural form in `pullback` instead of the canonical form suggested originally in [16]. This design makes RMPflow numerically stable, as only one matrix inversion $\mathbf{M}_r^{\dagger}\mathbf{f}_r$ is performed at the root node with both $\mathbf{f}_r$ and $\mathbf{M}_r$ in the span of the same Jacobian matrix due to `pullback`.

3.6 Example RMPs

We give a quick overview of some RMPs useful in practice (a complete discussion of these RMPs are postponed to Appendix D). We recall from (2) that $\mathbf{M}$ dictates the directional importance of an RMP.

Collision/joint limit avoidance Barrier-type RMPs are examples that use velocity dependent inertia matrices, which can express importance as a function of robot heading (a property that traditional mechanical principles fail to capture). Here we demonstrate a collision avoidance policy in the 1D distance space $x = d(\mathbf{q})$ to an obstacle. Let $g(x, \dot{x}) = w(x)u(\dot{x}) > 0$ for some functions w and u. We consider a motion policy such that $m(x, \dot{x})\ddot{x} + \frac{1}{2}\dot{x}^2\partial_x g(x, \dot{x}) = -\partial_x \Phi(x) - b\dot{x}$ and define its inertia matrix $m(x, \dot{x}) = g(x, \dot{x}) + \frac{1}{2}\dot{x}\partial_{\dot{x}} g(x, \dot{x})$, where Φ is a potential and $b > 0$ is a damper. We choose $w(x)$ to increase as x decreases (close to the obstacle), $u(\dot{x})$ to increase when $\dot{x} < 0$ (moving toward the obstacle), and $u(\dot{x})$ to be constant when $\dot{x} \geq 0$. With this choice, the RMP can be turned off in `pullback` when the robot heads away from the obstacle. This motion policy is a GDS and g is its metric (cf. Section 4.1); the terms $\frac{1}{2}\dot{x}\partial_{\dot{x}} g(x, \dot{x})$ and $\frac{1}{2}\dot{x}^2\partial_x g(x, \dot{x})$ are due to non-Euclidean geometry and produce natural repulsive behaviors.

Target attractors Designing an attractor policy is relatively straightforward. For a task space with coordinate $\mathbf{x}$, we can consider an inertia matrix $\mathbf{M}(\mathbf{x}) \succ 0$ and a motion policy such that $\ddot{\mathbf{x}} = -\nabla\widetilde{\Phi} - \beta(\mathbf{x})\dot{\mathbf{x}} - \mathbf{M}^{-1}\boldsymbol{\xi}_{\mathbf{M}}$, where $\widetilde{\Phi}(\mathbf{x}) \approx \|\mathbf{x}\|$ is a smooth attractor potential, $\beta(\mathbf{x}) \geq 0$ is a damper, and $\boldsymbol{\xi}_{\mathbf{M}}$ is a curvature term. It can be shown that this differential equation is also a GDS (see Appendix D.4).

Orientations As RMPflow directly works with manifold objects, orientation controllers become straightforward to design, independent of the choice of coordinate (cf. Section 4.4). For example, we can define RMPs on a robotic link's

surface in any preferred coordinate (e.g. in one or two axes attached to an arbitrary point) with the above described attractor to control the orientation. This follows a similar idea outlined in the Appendix of [16].

Q-functions Perhaps surprising, RMPs can be constructed using Q-functions as metrics (we invite readers to read [16] for details on how motion optimizers can be reduced to Q-functions and the corresponding RMPs). While these RMPs may not satisfy the conditions of a GDS that we later analyze, they represent a broader class of RMPs that leads to substantial benefits (e.g. escaping local minima) in practice. Also, Q-functions are closely relate to Lyapunov functions and geometric control [30]; we will further explore this direction in future work.

4 Theoretical Analysis of RMPflow

We investigate the properties of RMPflow when the child-node motion policies belong to a class of differential equations, which we call *structured geometric dynamical systems* (structured GDSs). We present the following results.

1. **Closure**: We show that the `pullback` operator retains a closure of structured GDSs. When the child-node motion policies are structured GDSs, the parent-node dynamics also belong to the same class.
2. **Stability**: Using the closure property, we provide sufficient conditions for the feedback policy of RMPflow to be stable. In particular, we cover a class of dynamics with *velocity-dependent* metrics that are new to the literature.
3. **Invariance**: As its name suggests, RMPflow is closely related to differential geometry. We show that RMPflow is intrinsically coordinate-free. This means that a set of subtask RMPs designed for one robot can be transferred to another robot while maintaining the same task-space behaviors.

Setup We assume all that manifolds and maps are sufficiently smooth. For now, we assume also that each manifold has a single chart; the coordinate-free analysis is postponed to Section 4.4. All the proofs are provided in Appendix B.

4.1 Geometric Dynamical Systems (GDSs)

We define a family of dynamics useful to specify RMPs on manifolds. Let manifold $\mathcal{M}$ be m-dimensional with chart $(\mathcal{M}, \mathbf{x})$. Let $\mathbf{G} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m\times m}$, $\mathbf{B} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_+^{m\times m}$, and $\Phi : \mathbb{R}^m \to \mathbb{R}$. The tuple $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)$ is called a *GDS* if and only if

$$\left(\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\right) \ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) = -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}, \tag{3}$$

where $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \frac{1}{2}\sum_{i=1}^{m} \dot{x}_i \partial_{\mathbf{x}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})$, $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \overset{\times}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}}(\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}})$, and $\overset{\times}{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq [\partial_{\mathbf{x}} \mathbf{g}_i(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}]_{i=1}^{m}$. We refer to $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ as the *metric* matrix, $\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})$ as the *damping* matrix, and $\Phi(\mathbf{x})$ as the *potential* function which is lower-bounded. In addition, we define $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}}) \coloneqq \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x}, \dot{\mathbf{x}})$ as the *inertia* matrix, which can be asymmetric. We say a GDS is *non-degenerate* if $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ is nonsingular. We will assume (3) is non-degenerate so that it uniquely defines a differential equation and discuss the general case in Appendix A. $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ induces

a metric of $\dot{\mathbf{x}}$, measuring its length as $\frac{1}{2}\dot{\mathbf{x}}^\top \mathbf{G}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}}$. When $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}})$ depends on $\mathbf{x}$ and $\dot{\mathbf{x}}$, it also induces the *curvature* terms $\boldsymbol{\Xi}(\mathbf{x},\dot{\mathbf{x}})$ and $\boldsymbol{\xi}(\mathbf{x},\dot{\mathbf{x}})$. In a particular case when $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, the GDSs reduce to the widely studied *simple mechanical systems* (SMSs) [15], $\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}} + \nabla_{\mathbf{x}}\Phi(\mathbf{x}) = -\mathbf{B}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}}$; in this case $\mathbf{M}(\mathbf{x}) = \mathbf{G}(\mathbf{x})$ and the Coriolis force $\mathbf{C}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}}$ is equal to $\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x},\dot{\mathbf{x}})$. The extension to velocity-dependent $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}})$ is important and non-trivial. As discussed in Section 3.6, it generalizes the dynamics of classical rigid-body systems, allowing the space to morph according to the velocity direction.

As its name suggests, GDSs possess geometric properties. Particularly, when $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}})$ is invertible, the left-hand side of (3) is related to a quantity $\mathbf{a}_{\mathbf{G}} = \ddot{\mathbf{x}} + \mathbf{G}(\mathbf{x},\dot{\mathbf{x}})^{-1}(\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x},\dot{\mathbf{x}})\ddot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{G}}(\mathbf{x},\dot{\mathbf{x}}))$, known as the *geometric acceleration* (cf. Section 4.4). In short, we can think of (3) as setting $\mathbf{a}_{\mathbf{G}}$ along the negative natural gradient $-\mathbf{G}(\mathbf{x},\dot{\mathbf{x}})^{-1}\nabla_{\mathbf{x}}\Phi(\mathbf{x})$ while imposing damping $-\mathbf{G}(\mathbf{x},\dot{\mathbf{x}})^{-1}\mathbf{B}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}}$.

4.2 Closure

Earlier, we mentioned that by tracking the geometry in `pullback` in (1), the task properties can be preserved. Here, we formalize the consistency of RMPflow as a closure of differential equations, named structured GDSs. Structured GDSs augment GDSs with information on how the metric matrix factorizes. Suppose $\mathbf{G}$ has a structure $\mathcal{S}$ that factorizes $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{J}(\mathbf{x})^\top \mathbf{H}(\mathbf{y},\dot{\mathbf{y}})\mathbf{J}(\mathbf{x})$, where $\mathbf{y} : \mathbf{x} \mapsto \mathbf{y}(\mathbf{x}) \in \mathbb{R}^n$ and $\mathbf{H} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+^{n\times n}$, and $\mathbf{J}(\mathbf{x}) = \partial_{\mathbf{x}}\mathbf{y}$. We say the tuple $(\mathcal{M}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ is a *structured GDS* if and only if

$$(\mathbf{G}(\mathbf{x},\dot{\mathbf{x}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{x},\dot{\mathbf{x}}))\,\ddot{\mathbf{x}} + \boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x},\dot{\mathbf{x}}) = -\nabla_{\mathbf{x}}\Phi(\mathbf{x}) - \mathbf{B}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}} \tag{4}$$

where $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{x},\dot{\mathbf{x}}) \coloneqq \mathbf{J}(\mathbf{x})^\top(\boldsymbol{\xi}_{\mathbf{H}}(\mathbf{y},\dot{\mathbf{y}}) + (\mathbf{H}(\mathbf{y},\dot{\mathbf{y}}) + \boldsymbol{\Xi}_{\mathbf{H}}(\mathbf{y},\dot{\mathbf{y}}))\dot{\mathbf{J}}(\mathbf{x},\dot{\mathbf{x}})\dot{\mathbf{x}})$. Note the metric and factorization *in combination* defines $\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}$. As a special case, GDSs are structured GDSs with a *trivial* structure (i.e. $\mathbf{y} = \mathbf{x}$). Also, structured GDSs reduce to GDSs (i.e. the structure offers no extra information) if $\mathbf{G}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$, or if $n, m = 1$ (cf. Appendix B.1). Given two structures, we say $\mathcal{S}_a$ *preserves* $\mathcal{S}_b$ if $\mathcal{S}_a$ has the factorization (of $\mathbf{H}$) made by $\mathcal{S}_b$. In Section 4.4, we will show that structured GDSs are related to a geometric object, pullback connection, which turns out to be the coordinate-free version of `pullback`.

To show the closure property, we consider a parent node on $\mathcal{M}$ with K child nodes on $\{\mathcal{N}_i\}_{i=1}^K$. We note that $\mathbf{G}_i$ and $\mathbf{B}_i$ can be functions of both $\mathbf{y}_i$ and $\dot{\mathbf{y}}_i$.

Theorem 1. *Let the ith child node follow $(\mathcal{N}_i, \mathbf{G}_i, \mathbf{B}_i, \Phi_i)_{\mathcal{S}_i}$ and have coordinate $\mathbf{y}_i$. Let $\mathbf{f}_i = -\boldsymbol{\eta}_{\mathbf{G}_i;\mathcal{S}_i} - \nabla_{\mathbf{y}_i}\Phi_i - \mathbf{B}_i\dot{\mathbf{y}}_i$ and $\mathbf{M}_i = \mathbf{G}_i + \boldsymbol{\Xi}_{\mathbf{G}_i}$. If $[\mathbf{f},\mathbf{M}]^{\mathcal{M}}$ of the parent node is given by* `pullback` *with $\{[\mathbf{f}_i,\mathbf{M}_i]^{\mathcal{N}_i}\}_{i=1}^K$ and $\mathbf{M}$ is non-singular, the parent node follows $(\mathcal{M},\mathbf{G},\mathbf{B},\Phi)_{\mathcal{S}}$, where $\mathbf{G} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{G}_i\mathbf{J}_i$, $\mathbf{B} = \sum_{i=1}^K \mathbf{J}_i^\top \mathbf{B}_i\mathbf{J}_i$, $\Phi = \sum_{i=1}^K \Phi_i \circ \mathbf{y}_i$, $\mathcal{S}$ preserves $\mathcal{S}_i$, and $\mathbf{J}_i = \partial_{\mathbf{x}}\mathbf{y}_i$. Particularly, if $\mathbf{G}_i$ is velocity-free and the child nodes are GDSs, the parent node follows $(\mathcal{M},\mathbf{G},\mathbf{B},\Phi)$.*

Theorem 1 shows structured GDSs are closed under `pullback`. It means that the differential equation of a structured GDS with a tree-structured task map can be computed by recursively applying `pullback` from the leaves to the root.

Corollary 1. *If all leaf nodes follow GDSs and $\mathbf{M}_r$ at the root node is nonsingular, then the root node follows $(\mathcal{C},\mathbf{G},\mathbf{B},\Phi)_{\mathcal{S}}$ as recursively defined by Theorem 1.*

4.3 Stability

By the closure property above, we analyze the stability of RMPflow when the leaf nodes are (structured) GDSs. For compactness, we will abuse the notation to write $\mathbf{M} = \mathbf{M}_r$. Suppose $\mathbf{M}$ is nonsingular and let $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ be the resultant structured GDS at the root node. We consider a Lyapunov candidate $V(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^\top \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \Phi(\mathbf{q})$ and derive its rate using properties of structured GDSs.

Proposition 1. *For* $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$, $\dot{\mathbf{V}}(\mathbf{q}, \dot{\mathbf{q}}) = -\dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$.

Proposition 1 directly implies the stability of structured GDSs by invoking LaSalle's invariance principle [31]. Here we summarize the result without proof.

Corollary 2. *For* $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$, *if* $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$, *the system converges to a forward invariant set* $\mathcal{C}_\infty \coloneqq \{(\mathbf{q}, \dot{\mathbf{q}}) : \nabla_{\mathbf{q}}\Phi(\mathbf{q}) = 0, \dot{\mathbf{q}} = 0\}$.

To show the stability of RMPflow, we need to further check when the assumptions in Corollary 2 hold. The condition $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$ is easy to satisfy: by Theorem 1, $\mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succeq 0$; to strictly ensure definiteness, we can copy $\mathcal{C}$ into an additional child node with a (small) positive-definite damping matrix. The condition on $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$ can be satisfied similarly. In addition, we need to verify the assumption that $\mathbf{M}$ is nonsingular. Here we provide a sufficient condition. When satisfied, it implies the global stability of RMPflow.

Theorem 2. *Suppose every leaf node is a GDS with a metric matrix in the form* $\mathbf{R}(\mathbf{x}) + \mathbf{L}(\mathbf{x})^\top \mathbf{D}(\mathbf{x}, \dot{\mathbf{x}})\mathbf{L}(\mathbf{x})$ *for differentiable functions* $\mathbf{R}$, $\mathbf{L}$, *and* $\mathbf{D}$ *satisfying* $\mathbf{R}(\mathbf{x}) \succeq 0$, $\mathbf{D}(\mathbf{x}, \dot{\mathbf{x}}) = \operatorname{diag}((d_i(\mathbf{x}, \dot{y}_i))_{i=1}^n) \succeq 0$, *and* $\dot{y}_i \partial_{\dot{y}_i} d_i(\mathbf{x}, \dot{y}_i) \geq 0$, *where* $\mathbf{x}$ *is the coordinate of the leaf-node manifold and* $\dot{\mathbf{y}} = \mathbf{L}\dot{\mathbf{x}} \in \mathbb{R}^n$. *It holds* $\boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) \succeq 0$. *If further* $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}}) \succ 0$, *then* $\mathbf{M} \in \mathbb{R}^{d\times d}_{++}$, *and the global RMP generated by RMPflow converges to the forward invariant set* $\mathcal{C}_\infty$ *in Corollary 2.*

A particular condition in Theorem 2 is when all the leaf nodes with velocity dependent metric are 1D. Suppose $x \in \mathbb{R}$ is its coordinate and $g(x, \dot{x})$ is its metric matrix. The sufficient condition essentially boils down to $g(x, \dot{x}) \geq 0$ and $\dot{x}\partial_{\dot{x}} g(x, \dot{x}) \geq 0$. This means that, given any $x \in \mathbb{R}$, $g(x, 0) = 0$, $g(x, \dot{x})$ is non-decreasing when $\dot{x} > 0$, and non-increasing when $\dot{x} < 0$. This condition is satisfied by the collision avoidance policy in Section 3.6.

4.4 Invariance

We now discuss the coordinate-free geometric properties of $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ generated by RMPflow. Due to space constraint, we only summarize the results (please see Appendix B.4 and, e.g., [32]). Here we assume that $\mathbf{G}$ is positive-definite.

We first present the coordinate-free version of GDSs (i.e. the structure is trivial) by using a geometric object called *affine connection*, which defines how tangent spaces on a manifold are related. Let $T\mathcal{C}$ denote the tangent bundle of $\mathcal{C}$, which is a natural manifold to describe the state space. We first show that a GDS on $\mathcal{C}$ can be written in terms of a unique, asymmetric affine connection ${}^G\nabla$ that is compatible with a Riemannian metric G (defined by $\mathbf{G}$) on $T\mathcal{C}$. It is important to note that G is defined on $T\mathcal{C}$ *not* the original manifold $\mathcal{C}$. As the metric matrix in a GDS can be velocity dependent, we need a larger manifold.

Theorem 3. *Let G be a Riemannian metric on $T\mathcal{C}$ such that, for $s = (q, v) \in T\mathcal{C}$, $G(s) = G^v_{ij}(s)dq^i \otimes dq^j + G^a_{ij}dv^i \otimes dv^j$, where $G^v_{ij}(s)$ and G^a_{ij} are symmetric and positive-definite, and $G^v_{ij}(\cdot)$ is differentiable. Then there is a unique affine connection ${}^G\nabla$ that is compatible with G and satisfies, $\Gamma^k_{i,j} = \Gamma^k_{ji}$, $\Gamma^k_{i,j+d} = 0$, and $\Gamma^k_{i+d,j+d} = \Gamma^k_{j+d,i+d}$, for $i, j = 1, \ldots, d$ and $k = 1, \ldots, 2d$. In coordinates, if $G^v_{ij}(\dot{q})$ is identified as $\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})$, then $\mathrm{pr}_3({}^G\nabla_{\dot{q}}\ddot{q})$ can be written as $\mathbf{a}_{\mathbf{G}} \coloneqq \ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}})$, where $\mathrm{pr}_3 : (\mathbf{q}, \mathbf{v}, \mathbf{u}, \mathbf{a}) \mapsto \mathbf{u}$ is a projection.*

We call $\mathrm{pr}_3({}^G\nabla_{\dot{q}}\dot{q})$ the *geometric acceleration* of $q(t)$ with respect to ${}^G\nabla$. It is a coordinate-free object, because pr_3 is defined independent of the choice of chart of $\mathcal{C}$. By Theorem 3, it is clear that a GDS can be written abstractly as $\mathrm{pr}_3({}^G\nabla_{\ddot{q}}\ddot{q}) = (\mathrm{pr}_3 \circ G^\sharp \circ F)(s)$, where $F : s \mapsto -d\Phi(s) - B(s)$ defines the covectors due to the potential function and damping, and $G^\sharp : T^*T\mathcal{C} \to TT\mathcal{C}$ denotes the inverse of G. In coordinates, it reads as $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}}) = -\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\nabla_{\mathbf{q}}\Phi(\mathbf{q}) + \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}})$, which is exactly (3).

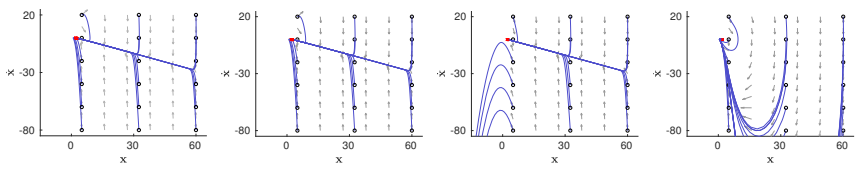
Next we present a coordinate-free representation of RMPflow.

Theorem 4. *Suppose $\mathcal{C}$ is related to K leaf-node task spaces by maps $\{\psi_i : \mathcal{C} \to \mathcal{T}_i\}_{i=1}^K$ and the ith task space $\mathcal{T}_i$ has an affine connection ${}^{G_i}\nabla$ on $T\mathcal{T}_i$, as defined in Theorem 3, and a covector function F_i defined by some potential and damping as described above. Let ${}^G\bar{\nabla} = \sum_{i=1}^K T\psi_i^{*G_i}\nabla$ be the pullback connection, $G = \sum_{i=1}^K T\psi_i^* G_i$ be the pullback metric, and $F = \sum_{i=1}^K T\psi_i^* F_i$ be the pullback covector, where $T\psi_i^* : T^*T\mathcal{T}_i \to T^*T\mathcal{C}$. Then ${}^G\bar{\nabla}$ is compatible with G, and $\mathrm{pr}_3({}^G\bar{\nabla}_{\ddot{q}}\ddot{q}) = (\mathrm{pr}_3 \circ G^\sharp \circ F)(s)$ can be written as $\ddot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\boldsymbol{\eta}_{\mathbf{G};\mathcal{S}}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\Xi}_{\mathbf{G}}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}}) = -\mathbf{G}(\mathbf{q}, \dot{\mathbf{q}})^{-1}(\nabla_{\mathbf{q}}\Phi(\mathbf{q}) + \mathbf{B}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}})$. In particular, if G is velocity-independent, then ${}^G\bar{\nabla} = {}^G\nabla$.*

Theorem 4 says that the structured GDS $(\mathcal{C}, \mathbf{G}, \mathbf{B}, \Phi)_{\mathcal{S}}$ can be written abstractly, without coordinates, using the pullback of task-space covectors, metrics, and asymmetric affine connections (that are defined in Theorem 3). In other words, the recursive calls of `pullback` in the backward pass of RMPflow is indeed performing "pullback" of geometric objects. Theorem 4 also shows, when G is velocity-independent, the pullback of connection and the pullback of metric commutes. In this case, ${}^G\bar{\nabla} = {}^G\nabla$, which is equivalent to the Levi-Civita connection of G. The loss of commutativity in general is due to the asymmetric definition of the connection in Theorem 3, which however is necessary to derive a control law of acceleration, without further referring to higher-order time derivatives.

4.5 Related Approaches

While here we focus on the special case of RMPflow with GDSs, this family already covers a wide range of reactive policies commonly used in practice. For example, when the task metric is Euclidean (i.e. constant), RMPflow recovers OSC (and its variants) [10, 19, 11, 12, 20]. When the task metric is only configuration dependent, RMPflow can be viewed as performing energy shaping to combine multiple SMSs in geometric control [15]. Further, RMPflow allows using velocity dependent metrics, generating behaviors all those previous rigid mechanics-based approaches fail to model. We also note that RMPflow can be



(a) (b) (c) (d)

Fig. 2: Phase portraits (gray) and integral curves (blue; from black circles to red crosses) of 1D example. (a) Desired behavior. (b) With curvature terms. (c) Without curvature terms. (d) Without curvature terms but with nonlinear damping.

easily modified to incorporate exogenous time-varying inputs (e.g. forces to realize impedance control [18] or learned perturbations as in DMPs [29]). In computation, the structure of RMPflow in natural-formed RMPs resembles the classical Recursive Newton-Euler algorithm [17, 33] (see Appendix C). Alternatively, the canonical form of RMPflow in (2) resembles Gauss' Principle [11, 12], but with a curvature correction $\boldsymbol{\Xi}_{\mathbf{G}}$ on the inertia matrix (suggested by Theorem 1) to account for velocity dependent metrics. Thus, we can view RMPflow as a natural generalization of these approaches to a broader class of non-Euclidean behaviors.

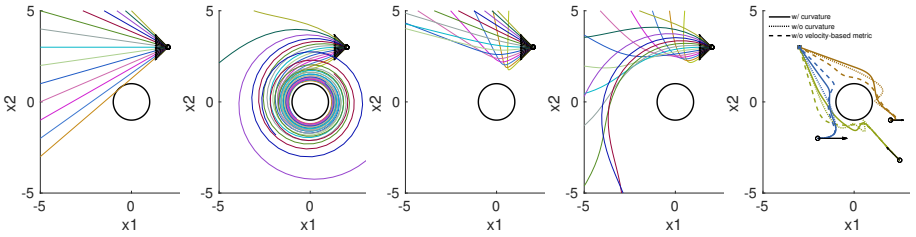
5 Experiments

We perform controlled experiments to study the curvature effects of nonlinear metrics, which is important for stability and collision avoidance. We then perform several full-body experiments (video: https://youtu.be/aFJMxfWV760) to demonstrate the capabilities of RMPflow on high-DOF manipulation problems in clutter, and implement an integrated vision-and-motion system on two physical robots.

5.1 Controlled Experiments

1D Example Let $\mathbf{q} \in \mathbb{R}$. We consider a barrier-type task map $\mathbf{x} = 1/\mathbf{q}$ and define a GDS in (3) with $\mathbf{G} = 1$, $\Phi(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^2$, and $\mathbf{B} = (1 + 1/\mathbf{x})$, where $\mathbf{x}_0 > 0$. Using the GDS, we can define an RMP $[-\nabla_{\mathbf{x}}\Phi - \mathbf{B}\dot{\mathbf{x}} - \boldsymbol{\xi}_{\mathbf{G}}, \mathbf{M}]^{\mathbb{R}}$, where $\mathbf{M}$ and $\boldsymbol{\xi}_{\mathbf{G}}$ are defined according to Section 4.1. We use this example to study the effects of $\dot{\mathbf{J}}\dot{\mathbf{q}}$ in `pullback` (1), where we define $\mathbf{J} = \partial_{\mathbf{q}}\mathbf{x}$. Fig. 2 compares the desired behavior (Fig. 2a) and the behaviors of correct/incorrect `pullback`. If `pullback` is performed correctly with $\dot{\mathbf{J}}\dot{\mathbf{q}}$, the behavior matches the designed one (Fig. 2b). By contrast, if $\dot{\mathbf{J}}\dot{\mathbf{q}}$ is ignored, the observed behavior becomes inconsistent and unstable (Fig. 2c). While the instability of neglecting $\dot{\mathbf{J}}\dot{\mathbf{q}}$ can be recovered with a damping $\mathbf{B} = (1 + \frac{\dot{\mathbf{x}}^2}{\mathbf{x}})$ nonlinear in $\dot{\mathbf{x}}$ (suggested in [20]), the behavior remains inconsistent (Fig. 2d).

2D Example We consider a 2D goal-reaching task with collision avoidance and study the effects of velocity dependent metrics. First, we define an RMP (a GDS as in Section 3.6) in $\mathbf{x} = d(\mathbf{q})$ (the 1D task space of the distance to the obstacle). We pick a metric $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) = w(\mathbf{x})u(\dot{\mathbf{x}})$, where $w(\mathbf{x}) = 1/\mathbf{x}^4$ increases if the particle is *close* to the obstacle and $u(\dot{\mathbf{x}}) = \epsilon + \min(0, \dot{\mathbf{x}})\dot{\mathbf{x}}$ (where $\epsilon \geq 0$), increases if it moves *towards* the obstacle. As this metric is non-constant, the GDS has curvature terms $\boldsymbol{\Xi}_{\mathbf{G}} = \frac{1}{2}\dot{\mathbf{x}}w(\mathbf{x})\partial_{\dot{\mathbf{x}}}u(\dot{\mathbf{x}})$ and $\boldsymbol{\xi}_{\mathbf{G}} = \frac{1}{2}\dot{\mathbf{x}}^2u(\dot{\mathbf{x}})\partial_{\mathbf{x}}w(\mathbf{x})$. These curvature



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

Fig. 3: 2D example; initial positions (small circle) and velocities (arrows). (a-d) Obstacle (circle) avoidance: (a) w/o curvature terms and w/o potential. (b) w/ curvature terms and w/o potential. (c) w/o curvature terms and w/ potential. (d) w/ curvature terms and w/ potential. (e) Combined obstacle avoidance and goal (square) reaching.

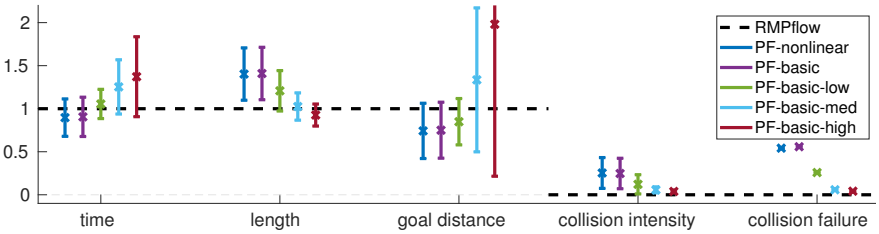
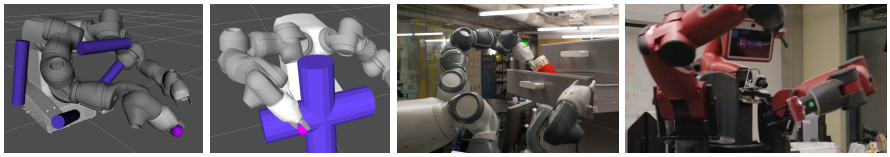


Fig. 4: Results for reaching experiments. Though some methods achieve a shorter goal distance than RMPflow in successful trials, they end up in collision in most the trials.

terms along with $\dot{\mathbf{J}}\dot{\mathbf{q}}$ produce an acceleration that lead to natural obstacle avoidance behavior, coaxing the system toward isocontours of the obstacle (Fig. 3b). On the other hand, when the curvature terms are ignored, the particle travels in straight lines with constant velocity (Fig. 3a). To define the full collision avoidance RMP, we introduce a barrier-type potential $\Phi(\mathbf{x}) = \alpha w(\mathbf{x}) \partial_{\mathbf{x}} w(\mathbf{x})$ to create extra repulsive forces, where $\alpha \geq 0$. A comparison of the curvature effects in this setting is shown in Fig. 3c and 3d (with $\alpha = 1$). Next, we use RMPflow to combine the collision avoidance RMP above (with $\alpha = 0.001$) and an attractor RMP. Let $\mathbf{q}_g$ be the goal. The attractor RMP is a GDS in the task space $\mathbf{y} = \mathbf{q} - \mathbf{q}_g$ with a metric $w(\mathbf{y})\mathbf{I}$, a damping $\eta w(\mathbf{y})\mathbf{I}$, and a potential that is zero at $\mathbf{y} = 0$, where $\eta > 0$ (see Appendix D.4). Fig. 3e shows the trajectories of the combined RMP. The combined non-constant metrics generate a behavior that transitions smoothly towards the goal while heading away from the obstacle. When the curvature terms are ignored (for both RMPs), the trajectories oscillate near the obstacle. In practice, this can result in jittery behavior on manipulators. When the metric is not velocity-based ($\mathbf{G}(\mathbf{x}) = w(\mathbf{x})$) the behavior is less efficient in breaking free from the obstacle to go toward the goal.

5.2 System Experiments

Reaching-through-clutter Experiments We compare RMPflow with OSC, (i.e. potential fields (PF) with dynamics reshaping), denoted as PF-basic, and a variant, denoted PF-nonlinear, which scales the collision-avoidance weights nonlinearly as a function of obstacle proximity. We highlight the results here; Appendix E provides additional details, and the supplementary video shows



simulated worlds real-world experiments

Fig. 5: Two of the six simulated worlds in the reaching experiments (left), and the two physical dual-arm platforms in the full system experiment (right).

footage of the trials. In both baselines, the collision-avoidance task spaces are specified by control points along the robot's body (rather than the distance space used in RMPflow) with an isotropic metric $\mathbf{G} = w(\mathbf{x})\mathbf{I}$ (here $w(\mathbf{x}) = w_o \in \mathbb{R}_+$ for PF-basic and $w(\mathbf{x}) \in [0, w_o]$ for PF-nonlinear, where w_o is the max metric size used in RMPflow). The task-space policies of both variants follow GDSs, but without the curvature terms (see Appendix E).

Fig. 4 summarizes their performance. We measure time-to-goal, C-space path length (assessing economy of motion), achievable distance-to-goal (efficacy in solving the problem), collision intensity (percent time in collision *given* a collision), collision failures (percent trials with collisions). The isotropic metrics, across multiple settings, fail to match the speed and precision achieved by RMPflow. Higher-weight settings tend to have fewer collisions and better economy of motion, but at the expense of efficiency. Additionally, adding nonlinear weights as in PF-nonlinear does not seem to help. The decisive factor of RMPflow's performance is rather its non-isotropic metric, which encodes directional importance around obstacles in combing policies.

System Integration for Real-Time Reactive Motion Generation We present an integrated system for vision-driven dual arm manipulation on two robotic platforms, the ABB YuMi robot and the Rethink Baxter robot (Fig. 5) (see the supplementary video). Our system uses the real-time optimization-based tracking algorithm DART [34] to communicate with the RMP system, receiving prior information on robot configuration and sending tracking updates of world state. The system is tested in multiple real-world manipulation problems, like picking up trash in clutter, reactive manipulation of a cabinet with human perturbation, active lead-through (compliant guiding of the arms with world-aware collision controllers) and pick-and-place of objects into a drawer which the robot opens and closes. Please see Appendix F for the details of the experiments.

6 Conclusion

We propose an efficient policy synthesis framework, RMPflow, for generating policies with non-Euclidean behavior, including motion with velocity dependent metrics that are new to the literature. In design, RMPflow is implemented as a computational graph, which can geometrically consistently combine subtask policies into a global policy for the robot. In theory, we provide conditions for stability and show that RMPflow is intrinsically coordinate-free. In the experiments, we demonstrate that RMPflow can generate smooth and natural motion for various tasks, when proper subtask RMPs are specified. Future work is to

further relax the requirement on the quality of designing subtask RMPs by introducing learning components into RMPflow for additional flexibility.

References

1. Rimon, E., Koditschek, D.: The construction of analytic diffeomorphisms for exact robot navigation on star worlds. Transactions of the American Mathematical Society 327(1), 71–116 (1991)
2. Ratliff, N., Toussaint, M., Schaal, S.: Understanding the geometry of workspace obstacles in motion optimization. In: IEEE International Conference on Robotics and Automation (ICRA) (2015)
3. Ivan, V., Zarubin, D., Toussaint, M., Komura, T., Vijayakumar, S.: Topology-based representations for motion planning and generalization in dynamic environments with interactions. International Journal of Robotics Research (IJRR) 32(9-10), 1151–1163 (2013)
4. Watterson, M., Liu, S., Sun, K., Smith, T., Kumar, V.: Trajectory optimization on manifolds with applications to SO(3) and R3XS2. In: Robotics: Science and Systems (RSS) (2018)
5. Toussaint, M.: Robot trajectory optimization using approximate inference. In: ICML. pp. 1049–1056 (2009)
6. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006), available at http://planning.cs.uiuc.edu/
7. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. International Journal of Robotics Research (IJRR) 30(7), 846–894 (2011), http://arxiv.org/abs/1105.1186
8. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: IEEE International Conference on Robotics and Automation (ICRA) (2015)
9. Mukadam, M., Dong, J., Yan, X., Dellaert, F., Boots, B.: Continuous-time Gaussian process motion planning via probabilistic inference. arXiv preprint arXiv:1707.07383 (2017)
10. Khatib, O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation 3(1), 43–53 (1987)
11. Peters, J., Mistry, M., Udwadia, F.E., Nakanishi, J., Schaal, S.: A unifying framework for robot control with redundant DOFs. Autonomous Robots 1, 1–12 (2008)
12. Udwadia, F.E.: A new perspective on the tracking control of nonlinear structural and mechanical systems. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 459(2035), 1783–1800 (2003), http://rspa.royalsocietypublishing.org/content/459/2035/1783
13. Kappler, D., Meier, F., Issac, J., Mainprice, J., Garcia Cifuentes, C., Wüthrich, M., Berenz, V., Schaal, S., Ratliff, N., Bohg, J.: Real-time perception meets reactive motion generation. IEEE Robotics and Automation Letters 3(3), 1864–1871 (2018), https://arxiv.org/abs/1703.03512
14. Mukadam, M., Cheng, C.A., Yan, X., Boots, B.: Approximately optimal continuous-time motion planning and control via probabilistic inference. In: IEEE International Conference on Robotics and Automation (ICRA) (2017)
15. Bullo, F., Lewis, A.D.: Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems, vol. 49. Springer Science & Business Media (2004)

16. Ratliff, N.D., Issac, J., Kappler, D., Birchfield, S., Fox, D.: Riemannian motion policies. arXiv preprint arXiv:1801.02854 (2018)
17. Walker, M.W., Orin, D.E.: Efficient dynamic computer simulation of robotic mechanisms. Journal of Dynamic Systems, Measurement, and Control 104(3), 205–211 (1982)
18. Albu-Schaffer, A., Hirzinger, G.: Cartesian impedance control techniques for torque controlled light-weight robots. In: IEEE International Conference on Robotics and Automation (ICRA). vol. 1, pp. 657–663 (2002)
19. Sentis, L., Khatib, O.: A whole-body control framework for humanoids operating in human environments. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 2641–2648 (2006)
20. Lo, S.Y., Cheng, C.A., Huang, H.P.: Virtual impedance control for safe human-robot interaction. Journal of Intelligent & Robotic Systems 82(1), 3–19 (2016)
21. Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., Todorov, E.: An integrated system for real-time model-predictive control of humanoid robots. In: IEEE/RAS International Conference on Humanoid Robots (2013)
22. Todorov, E.: Optimal control theory. In Bayesian Brain: Probabilistic Approaches to Neural Coding pp. 269–298 (2006)
23. Liegeois, A.: Automatic supervisory control of the configuration and behaviour of multibody mechanisms. IEEE Transactions on Systems, Man and Cybernetics 7(12), 868–871 (1977)
24. Ratliff, N., Zucker, M., Bagnell, J.A.D., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning. In: IEEE International Conference on Robotics and Automation (ICRA) (2009)
25. Mukadam, M., Yan, X., Boots, B.: Gaussian process motion planning. In: IEEE Conference on Robotics and Automation (ICRA) (2016)
26. Dong, J., Mukadam, M., Dellaert, F., Boots, B.: Motion planning as probabilistic inference using Gaussian processes and factor graphs. In: Robotics: Science and Systems (RSS) (2016)
27. Nakanishi, J., Cory, R., Mistry, M., Peters, J., Schaal, S.: Operational space control: A theoretical and empirical comparison. International Journal of Robotics Research (IJRR) 6, 737–757 (2008)
28. Platt, R., Abdallah, M.E., Wampler, C.W.: Multiple-priority impedance control. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 6033–6038. Citeseer (2011)
29. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Computation 25(2), 328–373 (Feb 2013)
30. Lewis, A.D.: The geometry of the maximum principle for affine connection control systems (2000)
31. Khalil, H.K.: Noninear systems. Prentice-Hall, New Jersey 2(5), 5–1 (1996)
32. Lee, J.M., Chow, B., Chu, S.C., Glickenstein, D., Guenther, C., Isenberg, J., Ivey, T., Knopf, D., Lu, P., Luo, F., et al.: Manifolds and differential geometry. Topology 643, 658 (2009)
33. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer (2008)
34. Schmidt, T., Newcombe, R., Fox, D.: DART: Dense articulated real-time tracking with consumer depth cameras. Autonomous Robots 39(3) (2015)
35. Taylor, J.R.: Classical Mechanics. University Science Books (2005)
36. Udwadia, F.E., Kalaba, R.E.: Analytical Dynamics: A New Approach. Cambridge University Press (1996)