

Geometric Fabrics for the Acceleration-based Design of Robotic Motion

Mandy Xie^{*†}, Karl Van Wyk*, Anqi Li^{*‡} and Muhammad Asif Rana[†],
 Qian Wan*, Dieter Fox^{*‡}, Byron Boots^{*‡}, Nathan D. Ratliff^{*}
 *NVIDIA ({nratliff,kvanwyk}@nvidia.com); [†]Georgia Tech; [‡]University of Washington

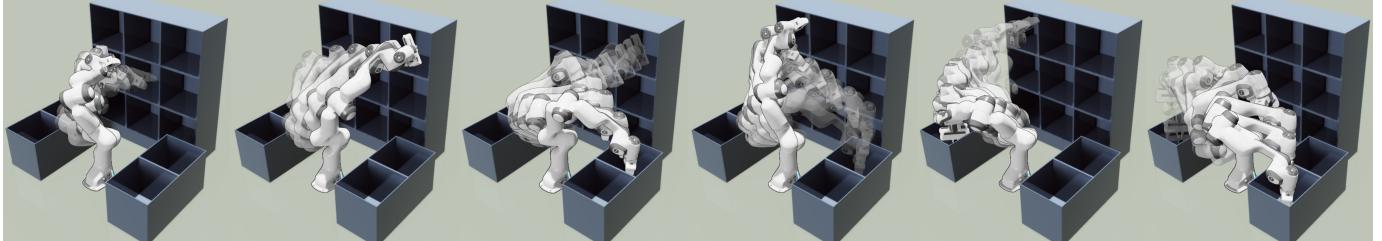


Fig. 1. Robot seamlessly navigating cubbies using a geometric fabric that enables obstacle and joint limit avoidance, redundancy resolution, and goal reaching.

Abstract—This paper describes the pragmatic design and construction of geometric fabrics for shaping a robot’s task-independent nominal behavior, capturing behavioral components such as obstacle avoidance, joint limit avoidance, redundancy resolution, global navigation heuristics, etc. Geometric fabrics constitute the most concrete incarnation of a new mathematical formulation for reactive behavior called optimization fabrics. Fabrics generalize recent work on Riemannian Motion Policies (RMPs); they add provable stability guarantees and improve design consistency while promoting the intuitive acceleration-based principles of modular design that make RMPs successful. We describe a suite of mathematical modeling tools that practitioners can employ in practice and demonstrate both how to mitigate system complexity by constructing behaviors layer-wise and how to employ these tools to design robust, strongly-generalizing, policies that solve practical problems one would expect to find in industry applications. Our system exhibits intelligent global navigation behaviors expressed entirely as provably stable fabrics with zero planning or state machine governance.

I. INTRODUCTION

Motion generation is one of the fundamental problems of robotics. Fast, reactive motion is essential for most modern tasks, especially in highly-dynamic and uncertain collaborative environments. We describe here a set of tools built from *geometric fabrics*, derived from our recent theory of *optimization fabrics* (see Section III of supplemental paper [16]), for the direct construction of *stable* robotic behavior in modular parts.¹ Fabrics define a nominal behavior independent of a specific task, capturing cross-task commonalities like joint-limit avoidance, obstacle avoidance, and redundancy resolution, and implement a task as an optimization problem across the fabric. The fabric defines behavior by shaping the optimization path.

For instance, the goal of a given task may be to reach a target point with the robot’s end-effector. En route, the system should avoid obstacles and joint limits, resolve redundancy intelligently, implement global navigation heuristics, and may

¹The term *fabric* is used analogously to the term *fabric of spacetime* from theoretical physics, but formalizes the idea as a second-order nonlinear differential equation characterizing an *unbiased* nominal behavior.

even shape the end-effector path to approach the target from a specific direction. The task is the optimization problem characterizing the end-effector’s target as its local minimum; the fabric captures everything else about the behavior we want the robot to exhibit as it optimizes that objective.

The theory of fabrics was motivated by the empirical success of Riemannian Motion Policies (RMPs) [15, 2] which have been shown to demonstrate flexible robust performance on real-world reactive and adaptive tasks. RMPs are intuitive. Designers can build behaviors as acceleration policies on different spaces and provide velocity dependent weight matrices defining how they should combine as metric weighted averages. Unfortunately, they have no theoretical guarantees of stability and empirically some care is needed to tune them well so the contributing RMP terms don’t conflict with one another. Fabrics, on the other hand, are fundamentally *unbiased* in a rigorous sense (see Section III of supplemental paper [16]), meaning practically that the underlying fabric does not prevent the system from achieving task goals. And they fundamentally engender asymptotic stability making them an alluring framework for behavioral design.

Geometric fabrics are a special type of fabric that expresses its unbiased nominal behavior as a generalized nonlinear geometry in the robot’s configuration space; they constitute the most concrete incarnation of optimization fabric and capture many of intuitive properties that make RMPs so powerful, such as acceleration-based policy design and independent priority metric specification. Similar to RMPs, geometric fabrics can be conveniently constructed in parts distributed across a transform tree of relevant task spaces. Importantly, they inherit key theoretical properties from the theory of fabrics, including stability and their unbiased behavior. Additionally, due to their construction as nonlinear geometries of paths, geometric fabrics exhibit a characteristic geometric consistency which allows us both to construct them layer-wise to mitigate design complexity and to independently control execution speed by accelerate along the direction of motion without affecting the

overall the behavior. Geometric fabrics capture RMP intuition but with important gains from their theoretical foundation.

We will show that a wide range of robotic behavior can be captured purely by its geometric fabric. We detail a pragmatic collection of modeling tools derived from this framework, and present experimental results on a Franka Panda robotic manipulator fluently navigating furniture mimicking problems common in logistics or industrial settings.

A. Related work

In [5] the authors observed that even systems built on classical planning [7] or optimization [14, 10, 4] require a layer of real-time reactive control leveraging techniques like operational space control [6]. Research into Riemannian Motion Policies (RMPs) [15, 2] built on these observations and proposed a behavioral design framework, embedding more globally aware behaviors into reactive control, powerful enough to develop strongly-generalizing systems² often circumventing standard planning architectures entirely.

Optimization fabrics (see Section III of supplemental paper [16]) are the culmination of that line of work into a comprehensive mathematical theory of behavioral design with rigorous stability guarantees, and geometric fabrics are their concrete incarnation. Earlier systems orchestrated RMPs in system applications using complex state machines to skirt the difficulty of designing nonlinear policies directly. These limitations motivated work on learning highly nonlinear RMPs from demonstration [13, 11, 9], but it proved challenging to integrate policy learning with existing RMP systems.

A fundamental limitation of many techniques, such as operational space control [6, 12], geometric control [1], and geometric dynamical systems [2], which we now understand as Lagrangian fabrics within the theory of fabrics, is that all of those systems *either* have fundamentally limited expressivity *or* they must express behavior through objective potentials (excludes velocity dependence) which creates conflicting objectives. This observation suggests that the challenges of incorporating learned RMPs stemmed from using the subclass of classical mechanical systems (a form of Lagrangian fabric). This paper studies the broader class of *geometric* fabric which is provably more flexible, exhibits geometrical consistency (speed-independence), and inherits rigorous stability guarantees from the theory of fabrics.

II. PRELIMINARIES

Geometric fabrics build on the theory of spectral semi-sprays (specs), which generalize the idea of modular second-order differential equations first derived and used as Riemannian Motion Policies (RMPs) in [15, 2]. Let \mathcal{C} be the d -dimensional configuration space of the robot. Throughout this paper, we will use vector-notation describing elements of a space in coordinates. Mapped task spaces $\mathbf{x} = \phi(\mathbf{q})$

²Strongly-generalizing system are systems designed and tested on a collection of validation examples that then perform robustly on an entire distribution of problems.

are defined in coordinates³ denoting $\mathbf{q} \in \mathcal{C} \subset \mathbb{R}^d$ and $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ with Jacobian matrix $\mathbf{J} = \partial_{\mathbf{x}}\phi$, used in the relations $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ and $\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$.

Natural-form specs $(M, f)_{\mathcal{X}}$ represent equations of the form $M(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + f(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{0}$, and their algebra derives from how these equations sum and transform under $\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$. *Canonical-form* specs $(M, h)_{\mathcal{X}}^{\mathcal{C}}$ express standard acceleration-form equation $\ddot{\mathbf{x}} + h(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{0}$ where $h = M^{-1}f$. For robotics applications we find it useful to additionally introduce a *policy-form* spec $[M, \pi]_{\mathcal{X}}$ to denote the *solved acceleration policy* expression $\ddot{\mathbf{x}} = -h(\mathbf{x}, \dot{\mathbf{x}}) = \pi(\mathbf{x}, \dot{\mathbf{x}})$.

In practice, we usually construct a transform tree of task spaces where the specs reside. Each directed edge of the tree represents the differentiable map taking its parent (domain) to its child (co-domain). Specs populating a transform tree collectively represent a complete second-order differential equation in parts, linking a given spec to the root via the chain of differentiable maps encountered along the unique path to the root. Denoting that composed map as $\mathbf{x} = \phi(\mathbf{q})$ as above, we can use the expressions $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ and $\ddot{\mathbf{x}} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$ relating velocities and accelerations in the task space to velocities and accelerations in the root to derive a spec algebra that defines both how specs combine on a single space and how they transform backward across edges from child to parent (see [2] for details). The tree implicitly represents a complete differential equation at the root as a sum of the parts, computed by recursive application of the spec algebra.

As in [2] second-order differential equations can easily be executed on fully actuated robotic systems using standard control techniques such as feedback linearization. For instance, a straightforward method is to use policies as trajectory generators (integrate forward integral curves) and follow those in the physical system using PID control as in [18].

III. THEORETICAL DEVELOPMENT

Here we review some of the necessary background for understanding geometric fabrics. We discuss the speed independence of *nonlinear geometries of paths* and how we can exploit that by *energizing* such geometries to conserve a type of energy called a Finsler energy which, itself, has geometric underpinnings. We then show how such energized geometries can be used to give global stability characterizations through a Hamiltonian much like in classical mechanics (the Hamiltonian can be used as a Lyapunov function).

Importantly, we note that these energized geometries can be viewed in two complimentary ways which gives intuition around how to use them in behavioral design. First, we can view it as endowing the geometry with a priority metric derived from the energy. Or second, and equivalently, since each Finsler energy has its own associated geometry, we can view it as *bending* that geometry into the shape of the desired geometry using a *zero work modification* which preserves its metric structure. This analysis shows that these energized

³The spec algebra defines covariant transforms, so the behavior is independent of curvilinear changes of coordinates [8]. For notational simplicity, we express our results a single choice of coordinates.

geometries (what we will see are *geometric fabrics* in the next section) can be used to optimize potential functions; later on, we show that potential functions can capture task objectives and therefore, optimizing potential functions are akin to solving task objectives while using the geometry to shape the behavior along the way.

A. Generalized nonlinear geometries

A generalized nonlinear geometry is an acceleration policy $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}})$ for which π has a special homogeneity property. We require that it be *positively homogeneous of degree 2* (HD2), which means that for any $\lambda \geq 0$ we have $\pi(\mathbf{x}, \lambda\dot{\mathbf{x}}) = \lambda^2\pi(\mathbf{x}, \dot{\mathbf{x}})$. One can show that the HD2 property ensures the differential equation is more than just a collection of trajectories (its integral curves); it additionally has a *path consistency* property whereby every integral curve starting from a given position \mathbf{x}_0 with velocity $\dot{\mathbf{x}}_0 = \eta\hat{\mathbf{n}}$ pointing in a given direction $\hat{\mathbf{n}}$ (here $\eta > 0$) will follow *the same path* (see discourse in supplemental [17]). In particular, any variant of the differential equation of the form $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}}) + \alpha(t, \mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$, where $\alpha \in \mathbb{R}$, will have integral curves that trace out the same paths as π . That geometric consistency property turns π into a *geometry of paths*.

B. Finsler energies

A Finsler energy $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$ is a generalization of kinetic energy from classical mechanics (the classical kinetic energy $\mathcal{K} = \frac{1}{2}\dot{\mathbf{x}}^T \mathbf{G}(\mathbf{x})\dot{\mathbf{x}}$ is a form of Finsler energy). Analogous to the classical case, the Euler-Lagrange equation applied to a Finsler energy defines an equation of motion $\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}})\ddot{\mathbf{x}} + \mathbf{f}_e(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{0}$ where $\mathbf{M}_e = \partial_{\dot{\mathbf{x}}\dot{\mathbf{x}}}^2 \mathcal{L}_e$ is the *energy (or metric) tensor* and $\mathbf{f}_e = \partial_{\dot{\mathbf{x}}\mathbf{x}} \mathcal{L}_e \dot{\mathbf{x}} - \partial_{\mathbf{x}} \mathcal{L}_e$ captures curvature terms (Coriolis and centripetal forces in classical mechanics). This equation matches the classical mechanical equations of motion when $\mathcal{L}_e = \mathcal{K}$, for which $\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{G}(\mathbf{x})$.

In geometric fabrics, the energy tensor defines the policy's *priority metric* and the curvature terms \mathbf{f}_e are used for stability. Finsler energies are Lagrangians, $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$, that satisfy

- 1) Positivity: $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}}) \geq 0$ with equality only for $\dot{\mathbf{x}} = \mathbf{0}$.
- 2) Homogeneity: $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$ is positively homogenous of degree 2 in $\dot{\mathbf{x}}$; i.e. for $\lambda \geq 0$ we have $\mathcal{L}_e(\mathbf{x}, \lambda\dot{\mathbf{x}}) = \lambda^2\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$.
- 3) Energy tensor invertibility: $\mathbf{M}_e = \partial_{\dot{\mathbf{x}}\dot{\mathbf{x}}}^2 \mathcal{L}_e$ is invertible.

The metric tensor $\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}})$ is in general a function of *velocity* as well as position, although the above homogeneity requirement enforces that \mathbf{M}_e depends only on the *directionality* of the velocity ($\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{M}_e(\mathbf{x}, \hat{\mathbf{x}})$ for $\dot{\mathbf{x}} \neq \mathbf{0}$) and not the magnitude. This enables directionally dependent priority matrices in Section VII from Finsler energies.

C. Energization

As previously discussed, any variant of the differential equation of the form $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}}) + \alpha(t, \mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$ will have integral curves that trace out the same paths as π (where π is HD2). Given a particular $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$ with associated \mathbf{M}_e and \mathbf{f}_e

terms, $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}}) + \alpha\dot{\mathbf{x}}$ will conserve \mathcal{L}_e if α is designed as (see Section IIIC of supplemental paper [16] for more details)

$$\alpha = -(\dot{\mathbf{x}}^T \mathbf{M}_e \dot{\mathbf{x}})^{-1} \dot{\mathbf{x}}^T [-\mathbf{M}_e \pi - \mathbf{f}_e]. \quad (1)$$

Moreover, $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}}) + \alpha\dot{\mathbf{x}}$ can then be rewritten as

$$\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e + \mathbf{P}_e [-\mathbf{M}_e \pi - \mathbf{f}_e] = \mathbf{0}, \quad (2)$$

where $\mathbf{P}_e = \mathbf{M}_e \mathbf{R}_{\mathbf{P}_e}$ and $\mathbf{R}_{\mathbf{P}_e} = \mathbf{M}_e^{-1} - \frac{\dot{\mathbf{x}}\dot{\mathbf{x}}^T}{\dot{\mathbf{x}}^T \mathbf{M}_e \dot{\mathbf{x}}}$. As discussed in Section IIIC of [16], $\mathbf{f}_f = \mathbf{P}_e [-\mathbf{M}_e \pi - \mathbf{f}_e]$ is a zero-work modification term, which means that $\dot{\mathbf{x}}^T \mathbf{f}_f = 0$. We rewrite (2) as $\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e + \mathbf{f}_f = \mathbf{0}$. As previously stated, this system will conserve \mathcal{L}_e , and therefore, its associated Hamiltonian, \mathcal{H}_e , will be conserved as well (i.e., $\dot{\mathcal{H}}_e = 0$).

D. Optimization

We can now optimize (minimize) a potential function $\psi(\mathbf{x})$ given the previous energy-conserving system with the forced and damped variant,

$$\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e + \mathbf{f}_f = -\partial_{\mathbf{x}} \psi - \mathbf{B} \dot{\mathbf{x}}, \quad (3)$$

where $\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})$ is positive definite. The total energy for this system (acting as our Lyapunov function) is $\mathcal{H}_e^\psi = \mathcal{H}_e + \psi(\mathbf{x})$. The time rate-of-change of the total energy is

$$\dot{\mathcal{H}}_e^\psi = \dot{\mathcal{H}}_e + \dot{\psi} = \dot{\mathbf{x}}^T (\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e) + \partial_{\mathbf{x}} \psi^T \dot{\mathbf{x}} \quad (4)$$

$$= \dot{\mathbf{x}}^T (\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e + \partial_{\mathbf{x}} \psi). \quad (5)$$

Rewriting our forced and damped system as $\ddot{\mathbf{x}} = -\mathbf{M}_e^{-1}(\mathbf{f}_e + \mathbf{f}_f + \partial_{\mathbf{x}} \psi + \mathbf{B} \dot{\mathbf{x}})$ and substituting yields,

$$\begin{aligned} \dot{\mathcal{H}}_e^\psi &= \dot{\mathbf{x}}^T \left(\mathbf{M}_e (-\mathbf{M}_e^{-1}(\mathbf{f}_e + \mathbf{f}_f + \partial_{\mathbf{x}} \psi + \mathbf{B} \dot{\mathbf{x}})) \right. \\ &\quad \left. + \mathbf{f}_e + \partial_{\mathbf{x}} \psi \right) \\ &= \dot{\mathbf{x}}^T (-\mathbf{f}_e - \partial_{\mathbf{x}} \psi - \mathbf{B} \dot{\mathbf{x}} + \mathbf{f}_e + \partial_{\mathbf{x}} \psi) - \dot{\mathbf{x}}^T \mathbf{f}_f \\ &= -\dot{\mathbf{x}}^T \mathbf{B} \dot{\mathbf{x}}, \end{aligned} \quad (6)$$

where all terms cancel except for the damping term. When \mathbf{B} is strictly positive definite, the rate of change is strictly negative for $\dot{\mathbf{x}} \neq \mathbf{0}$. Since $\mathcal{H}_e^\psi = \mathcal{H}_e + \psi$ is lower bounded and $\dot{\mathcal{H}}_e^\psi \leq 0$, we must have $\dot{\mathcal{H}}_e^\psi = -\dot{\mathbf{x}}^T \mathbf{B} \dot{\mathbf{x}} \rightarrow 0$ which implies $\dot{\mathbf{x}} \rightarrow \mathbf{0}$, and therefore, $\ddot{\mathbf{x}} \rightarrow \mathbf{0}$. Substituting $\dot{\mathbf{x}} = \mathbf{0}$ and $\ddot{\mathbf{x}} = \mathbf{0}$ into (3) yields $\partial_{\mathbf{x}} \psi = \mathbf{0}$, indicating that the system has come to rest at a minimum of $\psi(\mathbf{x})$.

IV. GEOMETRIC FABRICS

Above we showed that because geometric equations define collections of path, independent of speed, we can *energize* them to conserve a given measure of Finsler energy (which in turn endows them with a priority metric and can be viewed as bending the corresponding Finsler geometry into the shape of the given arbitrary geometric equation). Doing so creates a theoretical context we can use to understand their global properties: when forced by a potential function (and damped), they are guaranteed to minimize the potential over the domain.

The geometry influences the optimization path⁴ (the system’s local behavior), but its global stability is governed by the overall convergence a local minimum of the potential function.

Together, these properties allow us to specify tasks using potential functions (whose local minima describe task goals), and separately construct any number of behaviors by shaping the underlying geometry, while always maintaining convergence guarantees. We call such energized geometries *geometric fabrics*. Geometric fabrics are a form of *optimization fabric* (see Section III of supplemental paper [16]) which is the broader term for this type of differential equation designed to induce *behavior* by influencing the optimization path of a differential optimizer. Here we describe pragmatically how to effectively design the fabric to encode a desired behavior.

A *forced geometric fabric* is a collection of *fabric terms* defined as pairs $(\mathcal{L}_e, \pi)_\mathcal{X}$ of a Finsler energy $\mathcal{L}_e(\mathbf{x}, \dot{\mathbf{x}})$ and an acceleration policy $\ddot{\mathbf{x}} = \pi(\mathbf{x}, \dot{\mathbf{x}})$. Geometric terms define the fabric while forcing terms define the objective. A *geometric term* is a term $(\mathcal{L}_e, \pi_2)_\mathcal{X}$ for which π_2 is an HD2 geometry. A *forcing term* is a term $(\mathcal{L}_e, -\mathbf{M}_e^{-1} \partial_{\mathbf{x}} \psi)_\mathcal{X}$ which derives its policy from a potential function. As shown in Section X, we can optimize (minimize) this potential function when damping the system. Fabric terms can be added to spaces of a transform tree [2] for the modular design of composite behaviors.

Each fabric term defines a triple $(\mathbf{M}_e, \mathbf{f}_e, \pi)_\mathcal{X}$, where $\mathbf{M}_e \ddot{\mathbf{x}} + \mathbf{f}_e = \mathbf{0}$ derives from the Euler-Lagrange equation applied to \mathcal{L}_e , which can be viewed as two specs, a policy spec $[\mathbf{M}_e, \pi]_\mathcal{X}$ and a natural energy spec $(\mathbf{M}_e, \mathbf{f}_e)_\mathcal{X}$. Geometric fabric summation and pullback is, accordingly, defined in terms of the algebra of these two constituent specs.

Geometric fabrics are *unbiased* and thereby never prevent a system from reaching a local minimum of the objective. The objective, therefore, encodes concrete task goals independent of the fabric’s behavior. Additionally, geometric policies are geometrically consistent speed-invariant geometry of paths, which both simplifies the intuition on how they sum and enables behavior-invariant execution speed control.

The design of a geometric fabric follows the intuition of designing RMPs [15]. Policy specs $[\mathbf{M}_e, \pi_2]_\mathcal{X}$ model both a desired behavior π_2 and a priority matrix on that behavior \mathbf{M}_e defining how it combines with other policies as a metric-weighted average of parts. The spectrum of \mathbf{M}_e can assign different weights to different directions in the space, and both $\pi_2(\mathbf{x}, \dot{\mathbf{x}})$ and $\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}})$ have the flexibility of depending on both position \mathbf{x} and velocity $\dot{\mathbf{x}}$. Since \mathbf{M}_e is HD0 (see Section III-B), geometric terms remain geometric under summation and pullback (e.g. the policy resulting from a metric-weighted average of geometries is itself a geometry). The energy spec $(\mathbf{M}_e, \mathbf{f}_e)_\mathcal{X}$ of each fabric term is used only to guarantee stability during execution (see Section VI). Practitioners can, therefore, simply focus on designing the behavior policy specs $[\mathbf{M}_e, \pi_2]_\mathcal{X}$.

⁴The *optimization path* is the system trajectory generated when a fabric is forced by the negative gradient of an objective.

V. EXPLOITING GEOMETRIES FOR SPEED REGULATION

As we did in Section III-C for energization, we will again exploit the speed independence of these geometric paths, this time to accelerate those energized systems along the direction of motion to maintain a separate measure of desired speed as faithfully as possible without violating the stability constraints outlined in Section III-D (positive damping on the energized system). Fundamentally, this involves using the potential function to speed up by injecting energy into the system and using the damping term to slow down when desired by bleeding energy off. At times we can also explicitly inject additional energy into the system (what we call *boosting* the energy) as long as those energy injections are transient. For instance, we can use energy boosting to speed the system from rest quickly. By acting only along the direction of motion, the damper will not change the geometric path.

Details of our speed control methodology are given in the supplemental paper [16], but we review the framework here. Let $\pi_0(\mathbf{x}, \dot{\mathbf{x}})$ be a geometry, \mathbf{M}_e be its metric induced through energization via Finsler energy \mathcal{L}_e , and $\psi(\mathbf{x})$ be a forcing potential. The speed of the forced system can be regulated using an acceleration along the direction of motion $\alpha_{\text{reg}} \dot{\mathbf{x}}$ using

$$\ddot{\mathbf{x}} = -\mathbf{M}_e^{-1} \partial_{\mathbf{x}} \psi(\mathbf{x}) + \pi_0(\mathbf{x}, \dot{\mathbf{x}}) + \alpha_{\text{reg}} \dot{\mathbf{x}}. \quad (7)$$

By Theorem III.18 of [16], as long as $\alpha_{\text{reg}} < \alpha_{\mathcal{L}_e}$ where $\alpha_{\mathcal{L}_e}$ is the energization coefficient (see Equation 1), the system is equivalent to the original energized system with strictly positive damping coefficient. This result can be derived simply by using a velocity aligned damper $-\beta \dot{\mathbf{x}}$ on the energized system and rearranging the terms by collecting the energization term and velocity damper together (both of which are accelerations along the direction of motion). The above constraint then derives from the original conditions required for stability.

VI. EXECUTION AND ALGORITHMS

Once the forced geometric fabric is designed, one can transform it by accelerating and decelerating along the direction of motion using the methodology outlined in Section V (see also Section VII-C) to maintain a given measured of *execution energy* (e.g. speed of the end-effector or joint speed through the configuration space). The geometric consistency of the fabric ensures the behavior remains consistent despite these speed modulations. Many numerical integrators are appropriate for integrating the final differential equation. We find Euler (1ms time step) or fourth order Runge-Kutta (10ms time step) exhibit a good trade-off between speed and accuracy.

In full, we design a system in three parts (using a transform tree [2] of task spaces): (1) design the underlying behavioral fabric, (2) add a driving potential to define task goals, (3) design an execution energy for speed control.

Design of a forced geometric fabric:

- (1) Construct a transform tree.
- (2) Populate its nodes with fabric terms.
- (3) Add execution energy specs to the tree as needed to describe the execution energy we want to control.

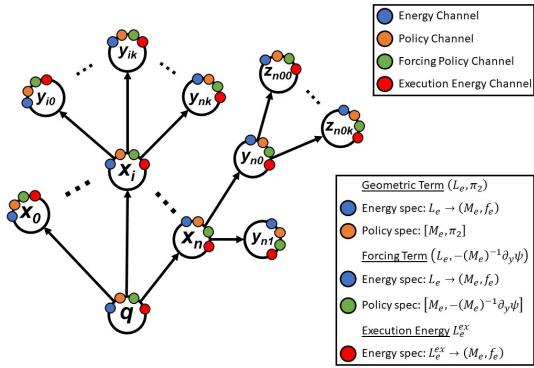


Fig. 2. Forced geometric fabric based on a transform tree of task spaces with four channels pass energies, policies, forcing policies, and execution energies.

Execution of the forced fabric with speed control:

- (1) Forward pass: Populate the nodes with the current state from the root to the leaves.
- (2) Backward pass: Evaluate the specs and pull them to root in separate channels, an *energy channel* for the geometric terms' energy specs, a *policy channel* for the geometric terms' policy specs, an *execution energy channel* for the execution energy specs, and a *forcing policy channel* for the forcing terms' policy specs. Add the forcing term's energy specs to the energy channel.
- (3) Use the four channels' root results to calculate the final desired acceleration using speed control.

Fig. 2 shows a transform tree of different spaces with four differently colored channels pass energies, policies, forcing policies, and execution energies backwards through the tree.

VII. CONCRETE DESIGN TOOLS

Geometric fabrics follow acceleration-based design principles captured in the original canonical-form RMPs [15]. As described in Section IV, a geometric fabric is a pair $(\mathcal{L}_e, \pi_2)_\chi$ characterizing two specs, an energy spec and a geometry spec. The energy spec captures stability information, while the geometry spec captures behavior. Behavioral design focuses on constructing the latter, using the class of HD2 geometries to model π_2 and deriving \mathbf{M}_e as the energy tensor from \mathcal{L}_e .

When geometric fabrics are summed $\sum_i (\mathcal{L}_e^{(i)}, \pi_2^{(i)})_\chi$, the combined fabric's geometry spec (capturing its behavior) $\sum_i (\mathbf{M}_e^{(i)}, \pi_2^{(i)})_\chi = (\widetilde{\mathbf{M}}_e, \widetilde{\pi}_2)$ is a metric-weighted average of the contributing geometries $\widetilde{\pi}_2 = (\sum_i \mathbf{M}_e^{(i)})^{-1} \sum_i \mathbf{M}_e^{(i)} \pi_2^{(i)}$, prioritized by the total metric $\widetilde{\mathbf{M}}_e = \sum_i \mathbf{M}_e^{(i)}$. When populating a transform tree, this intuitive combination rule is applied recursively at each node. Designers need only focus on intuitively creating modular acceleration policies (as HD2 geometries) in the different spaces and prioritizing them with metric tensors (from Finsler energies).

A. Construction of HD2 Geometries

An HD2 geometry is a differential equation $\ddot{\mathbf{x}} + \mathbf{h}_2(\mathbf{x}, \dot{\mathbf{x}}) = 0$ where \mathbf{h}_2 is HD2 (see Section III-A), which we usually denote in *policy form* $\ddot{\mathbf{x}} = -\mathbf{h}_2(\mathbf{x}, \dot{\mathbf{x}}) = \pi_2(\mathbf{x}, \dot{\mathbf{x}})$. Constructing an HD2 geometry is straightforward given the following rules of homogeneous functions: (1) a sum of HD2

functions is HD2; (2) multiplying homogeneous functions adds their degrees (denoting an HD k function as f_k , examples are $f_2 f_0 = f_2$, $f_1 f_1 = f_2$, etc.). For instance, a simple way to design an HD2 geometry is to choose an HD0 policy $\pi_0(\mathbf{x})$ that depends only on position and form $\pi_2(\mathbf{x}, \dot{\mathbf{x}}) = \|\dot{\mathbf{x}}\|^2 \pi_0(\mathbf{x})$ by scaling it by $\|\dot{\mathbf{x}}\|^2$. $\pi_0(\mathbf{x})$ can be chosen as the negative gradient of a potential $\pi_0(\mathbf{x}) = -\partial_{\mathbf{x}} \psi(\mathbf{x})$.

B. Acceleration-based Potentials

It is often most intuitive to design a geometric fabrics' forcing potential as a forcing *spec* $\mathcal{F} = [\mathbf{M}_f, \pi_f]_\chi$ in *policy form* so it's treated intuitively as another acceleration policy averaged into the final metric weighted average. This policy π_f must implicitly express a *forcing potential* $\psi_f(\mathbf{x})$ whose negative gradient is given by $-\partial_{\mathbf{x}} \psi_f(\mathbf{x}) = \mathbf{M}_f \pi_f$. When designing \mathcal{F} , \mathbf{M}_f and π_f must remain theoretically compatible in that sense. Following Appendix D.4 in [3], we advocate choosing $\pi_f = -\nabla_{\mathbf{x}} \psi_{\text{acc}}(\mathbf{x})$ where ψ_{acc} is a potential function that is spherically symmetric around its global minimum expressing the acceleration policy directly as its negative gradient. Any metric $\mathbf{M}_f(\mathbf{x})$ is theoretically compatible if it is also spherically symmetric around the same global minimum point. Note that position-only metrics are Riemannian and derive from Finsler energies of the form $\mathcal{L}_e = \frac{1}{2} \dot{\mathbf{x}}^T \mathbf{M}_f(\mathbf{x}) \dot{\mathbf{x}}$.

C. Speed control

We follow the speed control methodology outlined in Section V. Specifically, that entails using $\alpha_{\text{reg}} = \alpha_{\text{ex}}^\eta - \beta_{\text{reg}}(\mathbf{x}, \dot{\mathbf{x}}) + \alpha_{\text{boost}}$ in $\ddot{\mathbf{x}} = -\mathbf{M}_e^{-1} \partial_{\mathbf{x}} \psi(\mathbf{x}) + \pi_0(\mathbf{x}, \dot{\mathbf{x}}) + \alpha_{\text{reg}} \dot{\mathbf{x}}$ with $\beta_{\text{reg}} = s_\beta(\mathbf{x})B + \underline{B} + \max\{0, \alpha_{\text{ex}}^\eta - \alpha_{\mathcal{L}_e}\} \underline{B} > 0$ is a (small) baseline damping, the $B > \underline{B}$ is a larger damping coefficient for succinct convergence. The switch $s_\beta(\mathbf{x})$ turns on close to the target:

$$s_\beta(\mathbf{x}) = \frac{1}{2} \left(\tanh(-\alpha_\beta(\|\mathbf{x}\| - r)) + 1 \right) \quad (8)$$

where $\alpha_\beta \in \mathbb{R}^+$ is a gain defining the switching rate, and $r \in \mathbb{R}^+$ is the radius where the switch is half-way engaged. Denoting the desired execution energy as $\mathcal{L}_e^{\text{ex,d}}$, we use the following policy for η

$$\eta = \frac{1}{2} \left(\tanh(-\alpha_\eta(\mathcal{L}_e^{\text{ex}} - \mathcal{L}_e^{\text{ex,d}}) - \alpha_{\text{shift}}) + 1 \right) \quad (9)$$

where $\alpha_\eta, \alpha_{\text{shift}} \in \mathbb{R}^+$ adjust the rate and offset, respectively, of the switch as an affine function of the speed (execution energy) error. Finally, α_{boost} is modeled as $\alpha_{\text{boost}} = k \eta (1 - s_\beta(\mathbf{x})) \frac{1}{\|\dot{\mathbf{x}}\| + \epsilon}$, where $k \in \mathbb{R}^+$ is a gain that directly sets the desired level of acceleration, η (from above) sets $\alpha_{\text{boost}} = 0$ when the desired speed is achieved, and $1 - s_\beta(\mathbf{x})$ sets $\alpha_{\text{boost}} = 0$ when the system is within the region of higher damping. The normalization by $\|\dot{\mathbf{x}}\| + \epsilon$ ensures that α_{boost} is directly applied along $\dot{\mathbf{x}}$ with a very small positive value for ϵ to ensure numerical stability. This overall design injects more energy into the system when $-\alpha_{\text{boost}} < \alpha_{\text{ex}}^\eta - \alpha_{\mathcal{L}_e}$. Since this injection occurs for finite time, the total system energy is still bounded. The additional switches ensure that the system is still subject to positive damping, guaranteeing convergence.

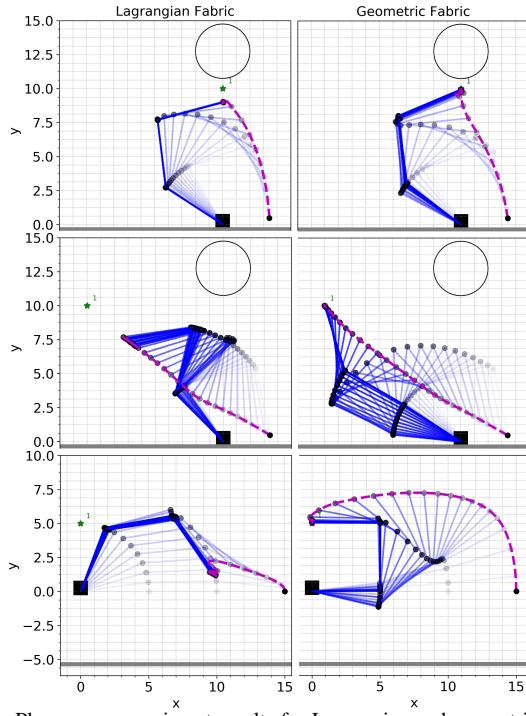


Fig. 3. Planar arm experiment results for Lagrangian and geometric fabrics, where targets are marked as green stars, and the arm's alpha transparency ranges from light at the beginning of the trajectory to dark at the end.

VIII. PLANAR MANIPULATOR EXPERIMENTS

In this section, we present a set of controlled experiments on a 3-dof planar arm to demonstrate the limitations of Lagrangian fabrics (as discussed in Section I-A) and show how geometric fabrics overcome them.

Lagrangian v.s. Geometric: Denoting Lagrangian fabric terms as $(\mathcal{L}_e, \pi_l)_\mathcal{X}$ we design Lagrangian fabric acceleration policies as $\pi_l = -\partial_{\mathbf{x}}\psi$ using acceleration-based potentials ψ . The corresponding geometric fabric terms are (\mathcal{L}_e, π_g) where $\pi_g = \|\dot{\mathbf{x}}\|^2\pi_l$, converting the position-only acceleration policies of the Lagrangian fabric into HD2 geometries for a *geometric* fabric simply by multiply by $\|\dot{\mathbf{x}}\|^2$. Lagrangian terms are actually all *forcing terms* per the categorization in Section IV. In contrast, geometric fabric terms add to the underlying fabric and are therefore inherently unbiased. Our experiments consistently show that the Lagrangian forcing terms can easily be tuned to compete with one another, preventing convergence to the task goal. In contrast, when converted to geometries, these same terms work collaboratively with one another to *influence* the optimization behavior without affecting task convergence.

All task goals are point targets for the end-effector expressed as simple acceleration-based attractor potentials (see Section VII-B). Our three experiments are to:

- 1) Reach toward a target close to an obstacle (we expect the obstacle and task terms to conflict).
- 2) Reach toward a target in an “awkward” configuration (we expect the redundancy resolution and task terms to conflict).
- 3) Reach toward a target at an “extreme” configuration (we expect the joint limit terms and task terms to conflict).

Figure 3 shows that in all cases, Lagrangian fabric (forcing) terms conflict with the task (forcing) term preventing convergence to the goal. In contrast, the analogous geometric fabrics are fundamentally unbiased and enable task convergence while still exhibiting the desired behavioral influence.

Below we describe each fabric term, focusing on describing their geometric fabric form; Lagrangian fabric terms are derived simply by dropping the velocity scaling factor $\|\dot{\mathbf{x}}\|^2$ using the relation $\pi_g = \|\dot{\mathbf{x}}\|^2\pi_l$.

End-effector Attraction: The behavior for pulling the end-effector toward a target is constructed as follows. The task map is $\mathbf{x} = \phi(\mathbf{q}) = \mathbf{q} - \mathbf{q}_d$, where $\mathbf{q}, \mathbf{q}_d \in \mathbb{R}^2$ are the current and desired particle position in Euclidean space. The metric is designed as

$$\mathbf{G}_\psi(\mathbf{x}) = (\bar{m} - m)e^{-(\alpha_m\|\mathbf{x}\|)^2}I + mI. \quad (10)$$

where \bar{m} , m are the upper and lower isotropic masses, respectively, and α_m is a constant scalar. The acceleration-based potential gradient is $\partial_{\mathbf{q}}\psi(\mathbf{x}) = M_\psi(\mathbf{x})\partial_{\mathbf{q}}\psi_1(\mathbf{x})$, where

$$\psi_1(\mathbf{x}) = k \left(\|\mathbf{x}\| + \frac{1}{\alpha_\psi} \log(1 + e^{-2\alpha_\psi\|\mathbf{x}\|}) \right) \quad (11)$$

where k and α_ψ are constant scalars. Since our task is to ultimately reach our target location, this behavior is a forcing geometric fabric term with $\pi = -\partial_{\mathbf{q}}\psi_1(\mathbf{x})$.

Circular Object Repulsion: Collision avoidance with respect to a circular object is constructed as follows. The task map is $x = \phi(\mathbf{q}) = \frac{\|\mathbf{q} - \mathbf{q}_o\|}{r} - 1$, where \mathbf{q}_o is the origin of the circle and r is its radius. The metric is defined as $G_b(x, \dot{x}) = s(\dot{x})\frac{k_b}{x^2}$, where k_b is a barrier gain and $s(\dot{x})$ is a velocity-based switching function. Specifically, $s(\dot{x}) = 1$ if $\dot{x} < 0$ and $s(\dot{x}) = 0$, otherwise. The acceleration-based potential gradient is $\partial_{\mathbf{q}}\psi_b(x) = M_b(x)\partial_{\mathbf{q}}\psi_{1,b}(x)$ with $\psi_{1,b}(x) = \frac{\alpha_b}{2x^8}$, where α_b is the barrier gain.

Limit Avoidance: To describe the joint limit avoidance policy, we use two 1D task maps, $x = \phi(q_j) = \bar{q}_j - q_j \in \mathbb{R}_+$ and $x = \phi(q_j) = q_j - \underline{q}_j \in \mathbb{R}^+$, for each joint to capture the distance between the current joint position and its lower and upper boundaries, respectively, where \bar{q}_j and \underline{q}_j are the upper and lower limits of the j^{th} joint. The 1D metric G_l is defined as $G_l(x, \dot{x}) = s(\dot{x})\frac{\lambda}{x}$, where $s(\dot{x})$ is a velocity-based switching function. Specifically, $s(\dot{x}) = 1$ if $\dot{x} < 0$ and $s(\dot{x}) = 0$, otherwise. Effectively, this removes the effect of the coordinate limit geometry once motion is orthogonal or away from the limit. The acceleration-based potential gradient, expressed as $\partial_{\mathbf{q}}\psi_l(x) = M_l(x)\partial_{\mathbf{q}}\psi_{1,l}(x)$, is designed with

$$\psi_l(x) = \frac{\alpha_1}{x^2} + \alpha_2 \log \left(e^{-\alpha_3(x-\alpha_4)} + 1 \right) \quad (12)$$

where $\alpha_1, \alpha_2, \alpha_3$, and α_4 are constant scalars.

Default Configuration: The task map for default configuration policy is defined as $\mathbf{x} = \phi(\mathbf{q}) = \mathbf{q} - \mathbf{q}_0$, where \mathbf{q}_0 is a nominal configuration, and it usually represents the robot “at ready”. The metric \mathbf{G}_{dc} is simply an identity matrix scaled by a constant λ_{dc} , $\mathbf{G}_{dc} = \lambda_{dc}I$. The acceleration-based potential gradient is defined with Eq. 15.

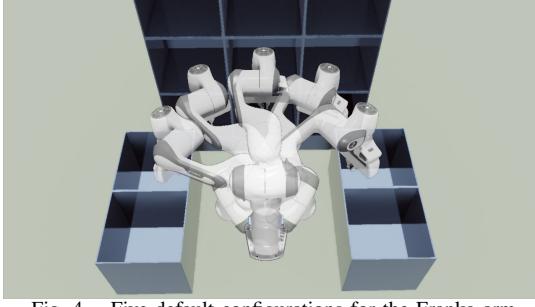


Fig. 4. Five default configurations for the Franka arm.

Experiment Results: The end-effector attraction, default configuration and joint limit avoidance are applied to all three experiments, and obstacle avoidance is applied to the first two experiments. The nominal default configuration in all three experiments is $[\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}]$, where positive sign defines counter clock-wise rotation, and vice versa. The upper and lower joint limits for each joint are set as $\pm(\pi)$ in the first two experiments and $\pm(\frac{\pi}{2} + 0.01)$ in the third experiment.

The experiment results are shown in Fig. 3. From the top row to the bottom row, it shows the results of experiments 1 to 3. As seen in the left column, no target is reached with Lagrangian fabrics due to the fighting potentials. As seen in the right column, all three targets are reached with geometric fabrics as shown in the right column.

IX. FRANKA ARM EXPERIMENTS

These experiments represent a global end-effector navigation problem common in many logistics, collaborative, and industrial settings, wherein a robot must autonomously interact with cubbies (bins). We demonstrate the effectiveness of the layer-wise construction of a geometric fabric that enables the robot to reach into and out of three sets of cubbies (see Fig. 1). There are six reachable cubbies in front and two on either side of the Franka arm. The length, width, and depth of each cubby are 0.3 m. The following discusses the fabric layers, where every new layer rests upon the previous, fixed layers, mitigating design and tuning complexity.

A. Behavioral Layers

We construct the behavior in three layers: 1) global cross-body navigation; 2) heuristic geometries for moving in and out of cubbies; 3) collision avoidance. Layers 1 and 3 are general and can be used in many contexts; Layer 2 is a common heuristic, although implemented here using task specific considerations. Each policy is an HD2 geometry of the form $\ddot{\mathbf{x}} = -\|\dot{\mathbf{x}}\|^2 \partial_{\mathbf{x}} \psi(\mathbf{x})$. Policies are weighted by $\mathbf{M}_e(\mathbf{x}, \dot{\mathbf{x}})$ from the Finsler energy, $\mathcal{L}_e = \dot{\mathbf{x}}^T \mathbf{G}(\mathbf{x}, \dot{\mathbf{x}}) \dot{\mathbf{x}}$. $\psi(\mathbf{x})$ and $\mathbf{G}(\mathbf{x}, \dot{\mathbf{x}})$ are defined as follows.

1) *Layer 1*: The first layer creates a baseline geometric fabric designed for *global*, cross-body, point-to-point end-effector navigation absent obstacles. We construct end-effector attraction, joint limit avoidance, and default configuration geometries (see Section VIII). Five separate default configurations are created to cover different regions of the robot (see Fig 4). This behavior controls robot posture and resolves manipulator redundancy.

2) *Layer 2*: This layer enables the end-effector to extract from and enter into any cubby (ignoring collision).

Cubby Extraction: This uses two task maps that are the distance between: 1) the end-effector and the *front plane* of the cubby, $y_1 = \phi_1(\mathbf{x}) = \text{sgn} * \|\mathbf{x}_f - \mathbf{x}\|$, $\text{sgn} = -1$ if the end-effector is inside of the cubby, $\text{sgn} = 1$, otherwise, where $\mathbf{x}, \mathbf{x}_f \in \mathbb{R}^3$ are the end-effector position and its orthogonal projection onto the front plane, respectively; and 2) the end-effector and a line that is centered and orthogonal to the front plane of the target cubby, $y_2 = \phi_2(\mathbf{x}) = \|\mathbf{x}_c - \mathbf{x}\|$, where $\mathbf{x}, \mathbf{x}_c \in \mathbb{R}^3$ are the end-effector position, and the closest point on the line to the end-effector, where the line is orthogonal and centered with the front plane. The priority metric is

$$G_w(\mathbf{x}) = s(y_1) \left((\bar{m} - \underline{m}) s(y_2) + \underline{m} \right) \mathbf{I}. \quad (13)$$

where $s(y_1) = 1$ if $y_1 < 0.1$ and $s(y_1) = 0$, otherwise. $s(y_2) = 0.5(\tanh(\alpha_m(y_2 - r)) + 1)$, \bar{m} , \underline{m} are the upper and lower isotropic masses, respectively, and α_m defines the rate of transition between 0 and 1, while $r = 0.15$ offsets the transition. Overall, the priority vanishes if the end-effector is either more than 0.1 m away from the front plane (outside of the cubby) or within 0.15 m of the target cubby center line. The potential function is the same as defined in Eq. 12.

Target Cubby Attraction: An additional target attraction policy is used to help pull the end-effector inside a target cubby. It is the same as the end-effector attraction used in Layer 1, but with the priority metric defined as Eq. 13. Overall, the priority remains zero until it enters a cylindrical region aligned with the target cubby. The heightening priority funnels motion into the cubby.

Way-Point Attraction: This term guides the arm to the cubby opening by attracting to a point 0.15m ahead of the front plane. The term generally matches the end-effector attraction term above, but with a different target and a switching function on the metric disabling it once within the column of the target cubby. Specifically, we make the following changes: 1) replace x_t with x_w in the task map, where $\mathbf{x}_w \in \mathbb{R}^3$ is the way point position; 2) define the switching function with the task map $y_2 = \phi_2(\mathbf{x}) = \|\mathbf{x}_c - \mathbf{x}\|$ as described in the cubby extraction policy. Note this term guides the arm but, as a geometric term, does not affect convergence to the target.

3) *Layer 3*: This layer enables cubby collision avoidance.

Cubby Collision: The task map is $y = \phi_c(\mathbf{x})$ which captures the minimum distance between a point on the arm and the cubby, where \mathbf{x} in \mathbb{R}^3 is a designated *collision* point on the robot. Denoting the closest point on the cubby as \mathbf{x}_c , we use $y = \phi_c(\mathbf{x}) = \|\mathbf{x}_c - \mathbf{x}\|$ (ignoring dependencies of \mathbf{x}_c on \mathbf{x} for simplicity). The metric is defined as a function of position, $G_b(y) = \frac{k_b}{y^2}$, where k_b is a barrier gain. The acceleration-based potential gradient, $\partial_{\mathbf{q}} \psi_b(y) = M_b(y) \partial_{\mathbf{q}} \psi_{1,b}(y)$, uses $\psi_{1,b}(y) = \frac{\alpha_b}{y^8}$, where α_b is the barrier gain.

B. Objective and Damping

The *optimization potential* is an acceleration-based design as discussed in VII-B, and it is designed for task space, $\mathbf{y} = \phi(\mathbf{x}) = \mathbf{x} - \mathbf{x}_t$, where $\mathbf{x}, \mathbf{x}_t \in \mathbb{R}^3$ are the current and target

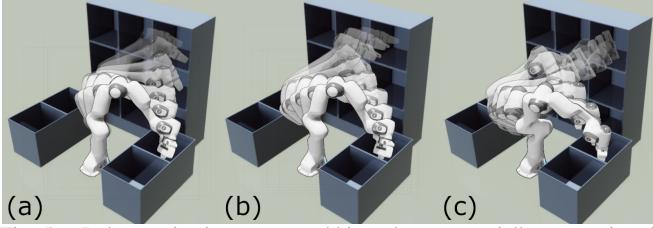


Fig. 5. Robot navigation among cubbies when sequentially composing the geometric fabric with (a) Layer 1, (b) Layer 2, and (c) Layer 3.

end-effector position in Euclidean space. The acceleration-based potential gradient, $\partial_y \psi(y) = M_\psi(y) \partial_y \psi_1(y)$, uses $\psi_1(y)$ as defined in Eq. 15, and the metric is defined as $G_\psi(y) = ((\bar{m} - m)s(\|y\|) + \underline{m})\mathbf{I}$, where $s(\|y\|) = 0.5(\tanh(\alpha_m(\|y\| - r)) + 1)$, \bar{m} , \underline{m} are the upper and lower isotropic masses, respectively, and α_m defines the rate of transition between 0 and 1, while r offsets the transition. This design allows for small attraction when far away from the target while smoothly increasing priority as getting closer to the target. *Damping* is defined as in VII-C, where the gain k in α_{boost} is defined as $k = -5|\dot{y}| - |\mathcal{L}_e^{\text{ex,d}}|$, where $|\dot{y}|$ is the current end-effector speed.

C. Experiment Results

As seen in Fig. 1, the robot intelligently navigates the cubbies. This global behavior was incrementally sequenced by adding layers of complexity to the underlying geometric fabric, a technique facilitated by geometric consistency and acceleration-based design. Fig. 5 shows the evolution of the robot trajectories for each of the three behavioral layers. Layer 1 depicts a more direct route to the target, ignoring the cubbies entirely. Layer 2 improves cubby navigation. Layer 3 enables complete collision avoidance. We dynamically change the desired end-effector goal during system execution, incurring rapid changes in system behavior to solve the desired task.

To demonstrate strong generalization, we hand tuned the system (analogous to training the system) with 6.7% (6 of 90) of the available problems (navigating between a pair of cubbies), validated it on 90 problems with a 96.7% success rate, and tested it on 90 problems with a success rate of 92.2%, where the base was rotated by 30 degrees counterclockwise. Standard tuning protocol would iterate this cycle of tuning and testing until the system performed well on the entire problem distribution; this success rate is indicative of a generally fast tuning convergence. A second round of tuning easily solved the observed issues raising the success rate in both cases to 100%. We also tested the tuned policy on the 6 training problems with an array of perturbed variants, mimicking relocating the robot to the scene (such as with a mobile base). Perturbations to the base ($\Delta x, \Delta y, \Delta \theta$) include combinations of $(-0.1 \text{ m}, \pm 0.1 \text{ m}, \pm \frac{\pi}{6} \text{ rad})$ and $(-0.1 \text{ m}, \pm 0.1 \text{ m}, 0 \text{ rad})$. The system behaved well on all variants.

We additionally quantified path consistency across execution speeds. Five different execution speeds are tested for the same task with targets located in 7 different cubbies. The corresponding end-effector trajectories and speeds are shown in Fig 6. The minor differences among the paths are caused

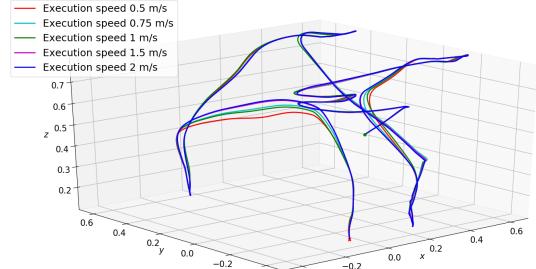


Fig. 6. End-effector trajectories for five different execution speeds.

by the optimization potential pushing the system away from the unbiased nominal path defined by the geometry. Generally speaking, slower execution speeds subjects the system to greater influence from the potential, increasing the departure from path consistency. This is evident in Fig. 6 where the fastest speeds (blue and purple curves) are nearly coincident and the slower speeds have greater changes to the paths taken. To facilitate path consistency, the optimization potential has a large gradient only when close to its minimum.

We analyze path consistency in the configuration space by calculating the average arc length integral of each path against a cost defined as a function of the difference between this path and another one as $\mathbf{L} = \left(\sum_{t=0}^T \|\dot{q}_t\| \Delta t \right)^{-1} \sum_{t=0}^T c(q_t, Q) \|\dot{q}_t\| \Delta t$, where $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^7$, and $c(q_t, Q)$ defines the minimum distance between a point at time t , namely q_t , and another path Q , and $\Delta t = 0.001 \text{ s}$. Based on this definition, we would expect the average arc length integrals for each path to be zero if the path matches each other perfectly. The average arc length integral of each path with respect to the other ones in the configuration space is shown in Table I, and from which we can observe that the arc length integral increases as the difference in the execution speed increases.

TABLE I
ARC LENGTH INTEGRALS IN THE CONFIGURATION SPACE.

Speed (m/s)	0.5	0.75	1.0	1.5	2.0
0.5	0.0	0.0473	0.0550	0.0992	0.1193
0.75	0.0473	0.0	0.0444	0.0619	0.0868
1.0	0.0550	0.0444	0.0	0.0895	0.1110
1.5	0.0992	0.0619	0.0895	0.0	0.0268
2.0	0.1193	0.0868	0.1110	0.0268	0.0

X. CONCLUSIONS

Geometric fabrics are RMPs whose acceleration policies are HD2 geometries and whose priority metrics derive from Finsler energies. They constitute compact encoding of behavior simple enough to shape by hand. They rival the broadest class of RMPs [15] in expressivity and add a geometric consistency enabling simple layer-wise design and speed regulation. We document concrete design tools and layering procedures effective in realistic settings. Interestingly, the strong generalization observed in our experiments suggests geometric fabrics may additionally represent a well-informed and flexible inductive bias for policy learning.

REFERENCES

- [1] Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer Science & Business Media, 2004.
- [2] C.-A. Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. RMPflow: A computational graph for automatic motion policy generation. In *The 13th International Workshop on the Algorithmic Foundations of Robotics*, 2018.
- [3] Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A computational graph for automatic motion policy generation. *arXiv:1811.07049*, 2018.
- [4] Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. An integrated system for real-time model-predictive control of humanoid robots. In *IEEE/RAS International Conference on Humanoid Robots*, 2013.
- [5] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, July 2018. URL <https://arxiv.org/abs/1703.03512>.
- [6] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987.
- [7] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at <http://planning.cs.uiuc.edu/>.
- [8] John M. Lee. *Introduction to Smooth Manifolds*. Springer, 2nd edition, 2012.
- [9] Anqi Li, Mustafa Mukadam, Magnus Egerstedt, and Byron Boots. Multi-objective policy generation for multi-robot systems using riemannian motion policies. *CoRR*, abs/1902.05177, 2019. URL <http://arxiv.org/abs/1902.05177>.
- [10] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time Gaussian process motion planning via probabilistic inference. *International Journal of Robotics Research (IJRR)*, 37(11):1319–1340, 2018.
- [11] Mustafa Mukadam, Ching-An Cheng, Dieter Fox, Byron Boots, and Nathan Ratliff. Riemannian motion policy fusion through learnable lyapunov function reshaping. In *Conference on Robot Learning (CoRL)*, 2019.
- [12] Jan Peters, Michael Mistry, Firdaus Udawadia, Jun Nakanishi, and Stefan Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots*, 24(1):1–12, 2008.
- [13] M Asif Rana, Anqi Li, Harish Ravichandar, Mustafa Mukadam, Sonia Chernova, Dieter Fox, Byron Boots, and Nathan Ratliff. Learning reactive motion policies in multiple task spaces from human demonstrations. In *Conference on Robot Learning (CoRL)*, 2019.
- [14] Nathan Ratliff, Marc Toussaint, and Stefan Schaal. Understanding the geometry of workspace obstacles in motion optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [15] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv:1801.02854*, 2018.
- [16] Nathan D. Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Asif Muhammad Rana. Optimization fabrics for behavioral design. *arXiv:2010.15676 [cs.RO]*, 2020. URL <https://arxiv.org/abs/2010.15676>.
- [17] Nathan D. Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Asif Muhammad Rana. Generalized nonlinear and finsler geometry for robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [18] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. Voorhies, G. Sukhatme, and S. Schaal. An autonomous manipulation system based on force control and optimization. *Autonomous Robots*, 2013.

A. CONTROLLED PARTICLE EXPERIMENTS

These experiments validate the theoretical findings and generate insight into the behavioral differences between geometric fabrics and Lagrangian fabrics, and usage of speed control. A set of 14 particles at rest to the right of a circular obstacle are attracted to a point to the left (small square). Behavior is integrated for 15 seconds using a fourth order Runge-Kutta method.

A. Point Attraction

The behavior for pulling the end-effector towards a target location is constructed as follows. The task map is $\mathbf{x} = \phi(\mathbf{q}) = \mathbf{q} - \mathbf{q}_d$, where $\mathbf{q}, \mathbf{q}_d \in \mathbb{R}^2$ are the current and desired particle position in Euclidean space. The metric is designed as

$$\mathbf{G}_\psi(\mathbf{x}) = (\bar{m} - \underline{m})e^{-(\alpha_m \|\mathbf{x}\|)^2} I + \underline{m}I. \quad (14)$$

where $\bar{m}, \underline{m} \in \mathbb{R}^+$ are the upper and lower isotropic masses, respectively, and $\alpha_m \in \mathbb{R}^+$ controls the width of the radial basis function. For the following experiments, $\bar{m} = 2$, $\underline{m} = 0.2$, and $\alpha_m = 0.75$. The acceleration-based potential gradient, $\partial_{\mathbf{q}}\psi(\mathbf{x}) = M_\psi(\mathbf{x})\partial_{\mathbf{q}}\psi_1(\mathbf{x})$, is designed with

$$\psi_1(\mathbf{x}) = k \left(\|\mathbf{x}\| + \frac{1}{\alpha_\psi} \log(1 + e^{-2\alpha_\psi \|\mathbf{x}\|}) \right) \quad (15)$$

where $k \in \mathbb{R}^+$ controls the overall gradient strength, $\alpha_\psi \in \mathbb{R}^+$ controls the transition rate of $\partial_{\mathbf{q}}\psi_1(\mathbf{x})$ from a positive constant to 0. For these experiments, $k = 10$, $\alpha_\psi = 10$.

For Lagrangian Fabrics, the policy is created from the Lagrangian, $\mathcal{L} = \dot{\mathbf{x}}^T \mathbf{G}_\psi(\mathbf{x}) \dot{\mathbf{x}} - \psi(\mathbf{x})$. For geometric fabrics, the policy is constructed as $\ddot{\mathbf{x}} = -\partial_{\mathbf{x}}\psi_1(\mathbf{x})$ and weighted by $\mathbf{M}_\psi(\mathbf{x})$ from the Finsler energy, $\mathcal{L} = \dot{\mathbf{x}}^T \mathbf{G}_\psi(\mathbf{x}) \dot{\mathbf{x}}$.

B. Circular Object Repulsion

Collision avoidance with respect to a circular object is constructed as follows. The task map is $x = \phi(\mathbf{q}) = \frac{\|\mathbf{q} - \mathbf{q}_o\|}{r} - 1$, where \mathbf{q}_o is the origin of the circle and r is its radius. Two different metrics are designed to prioritize this behavior. The first is just a function of position, $G_b(x) = \frac{k_b}{x^2}$, while the second is a function of both position and velocity, $G_b(x, \dot{x}) = s(\dot{x}) \frac{k_b}{x^2}$. Moreover, $k_b \in \mathbb{R}^+$ is a barrier gain and $s(\dot{x})$ is a velocity-based switching function. Specifically, $s(\dot{x}) = 1$ if $\dot{x} < 0$ and $s(\dot{x}) = 0$, otherwise. For these experiments, $k_b = 20$.

The acceleration-based potential gradient, $\partial_{\mathbf{q}}\psi_b(x) = M_b(x)\partial_{\mathbf{q}}\psi_{1,b}(x)$, is designed with $\psi_{1,b}(x) = \frac{\alpha_b}{2x^8}$, where $\alpha_b \in \mathbb{R}^+$ is the barrier gain and $\alpha_b = 1$ for these experiments.

For Lagrangian fabrics, the policy is created from either $\mathcal{L} = G_b(x)\dot{x}^2 - \psi_b(x)$ or $\mathcal{L} = G_b(x, \dot{x})\dot{x}^2 - \psi_b(x)$. For geometric fabrics, the policy is $\ddot{\mathbf{x}} = -s(\dot{x})\dot{x}^2\partial_x\psi_{1,b}(x)$. This policy is weighted by $\mathbf{M}_b(x)$ or $\mathbf{M}_b(x, \dot{x})$.

C. Experiments

Speed control is used with the execution energy designed as $\mathcal{L}_{ex} = \frac{1}{v_d} \dot{\mathbf{q}}^T \dot{\mathbf{q}}$, where $v_d \in \mathbb{R}^+$ is the desired Euclidean speed. For both geometric and Lagrangian fabrics, behavior is simulated for $v_d = 2, 4$, for cases using $G_b(x)$ and $G_b(x, \dot{x})$ (see Fig. 7).

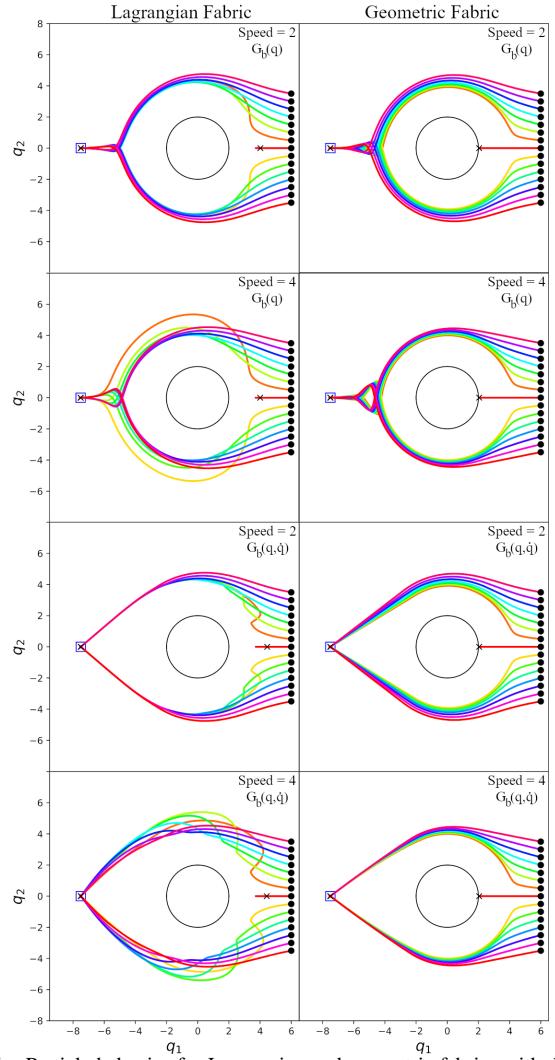


Fig. 7. Particle behavior for Lagrangian and geometric fabrics with different metric designs.

All particles successfully circumnavigate the object and reach the target position. However, the Lagrangian fabrics produce more inconsistent behavior across the speed levels. This is pronounced when using $G_b(x, \dot{x})$ since the velocity gate does not modulate the entire obstacle avoidance policy for Lagrangian fabrics. Instead, the mass of the obstacle avoidance policy vanishes, while components of its force remain. The effect of these forces are amplified when traveling at a higher velocity, producing the “launching” artifacts. In contrast, geometric fabrics produce more consistent paths across speed levels without any launching artifacts. Finally, the velocity gate facilitates straight-line motion to the desired location once the obstacle is bypassed. Without this gate the system possesses large mass, impeding motion to the desired location.

Using the same geometric fabric (with obstacle velocity gate) and optimization potential from the previous section, the system is evaluated using: 1) speed control with $v_d = 2.5$, and 2) basic damping with $\beta = 4$. The resulting time-speed traces are shown in Fig. 8. While both strategies still optimize, basic damping leads to a rather chaotic speed profile over

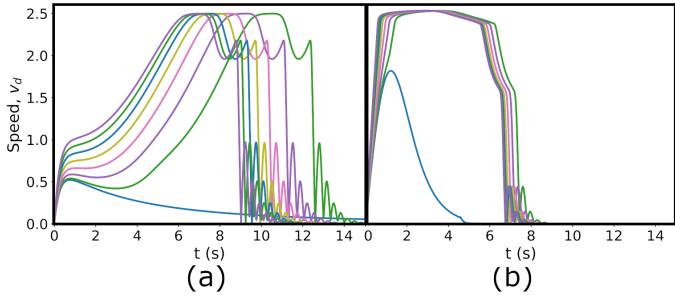


Fig. 8. Speed profiles over time when optimizing over a geometric fabric with (a) basic damping and (b) speed control.

time. In contrast, speed control achieves the desired speed in approximately 1 second and maintains it before damping to the final convergence point.