2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Geometric Fabrics: a Safe Guiding Medium for Policy Learning

Karl Van Wyk[1], Ankur Handa[1], Viktor Makoviychuk[1], Yijie Guo[1], Arthur Allshire[1,2] and Nathan D. Ratliff[1]

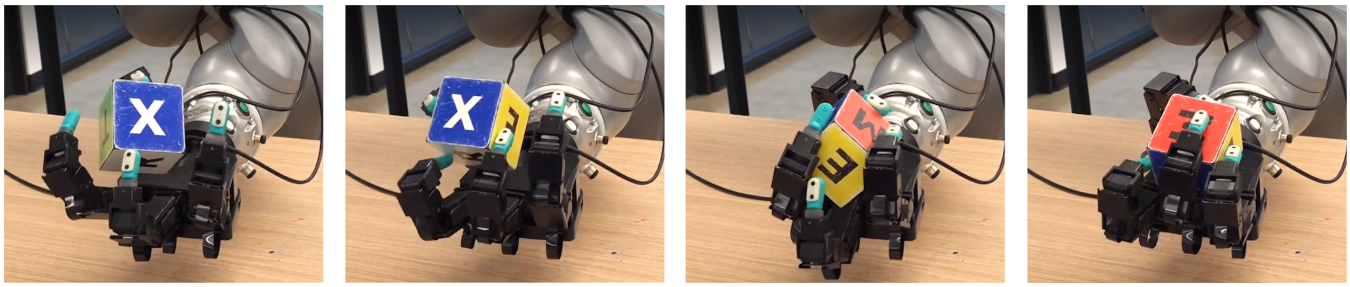

Fig. 1. Reinforcement learning over a geometric fabric layer yields safe, high-performance manipulation behavior for a highly-actuated hand. The learned behavior switches between two-, three-, and four-fingered grasps during prehensile manipulation. Videos at `https://dextreme.org/fgp.html`.

***Abstract*— Robotics policies are always subjected to complex, second order dynamics that entangle their actions with resulting states. In reinforcement learning (RL) contexts, policies have the burden of deciphering these complicated interactions over massive amounts of experience and complex reward functions to learn how to accomplish tasks. Moreover, policies typically issue actions directly to controllers like Operational Space Control (OSC) or joint PD control, which induces straightline motion towards these action targets in task or joint space. However, straightline motion in these spaces for the most part do not capture the rich, nonlinear behavior our robots need to exhibit, shifting the burden of discovering these behaviors more completely to the agent. Unlike these simpler controllers, geometric fabrics capture a much richer and desirable set of behaviors via artificial, second order dynamics grounded in nonlinear geometry. These artificial dynamics shift the uncontrolled dynamics of a robot via an appropriate control law to form *behavioral dynamics*. Behavioral dynamics unlock a new action space and safe, guiding behavior over which RL policies are trained. Behavioral dynamics enable bang-bang-like RL policy actions that are still safe for real robots, simplify reward engineering, and help sequence real-world, high-performance policies. We describe the framework more generally and create a specific instantiation for the problem of dexterous, in-hand reorientation of a cube by a highly actuated robot hand.**

I. Introduction

Imbuing robots with high-performance, real-world manipulation skills is critical for their societal relevance. However, such capabilities remain largely elusive, especially for robots with a large number of actuators and high-frequency joint control. Historically, analytically derived controllers have shown successful demonstrations in enabling robots to reach through their environments [1], [2], manipulate objects [3], enable end-effector control [4], and perform insertion [5]. Although the behavior of these controllers are well understood along with interpretable parameters, their applicability is limited to our ability to design them which are always accompanied by constraining assumptions.

Recently, there have been an escalating number of advancements in robots learning their manipulation skills. Striking exhibitions include high degree-of-actuation (DoA) robots manipulating objects [6], [7], [8], [9] and collaborative arms performing tight-tolerance insertions [10]. Furthermore, these skills were trained entirely in simulation via online RL methods and demonstrate real-world robot manipulation skills that are often not achievable by any other means. Similar findings have been reported in legged robots [11], [12] as well as character animation [13], [14].

Reinforcement learning grounded in simulation is a very compelling approach in that it can train value functions and policies across an enormous amount of diverse experience. For instance, decades or centuries of simulated experience can be achieved in only a few real-world days [8]. However, reinforcement learning does come with challenges like reward engineering, optimization efficacy, and resulting policies are often not safe for real-world deployment. Despite action regularization, RL policies still tend to bang-bang actions which are damaging to actuator drives of robots. To combat these destructive action signals, low-pass filters are often applied to reject high-frequency content in the actions [8], [12], [6], [7], [9], but this still does not handle motor constraints explicitly, can result in sluggish policy behavior, and still admit high-frequency content in actions.

A promising avenue forward for sequencing high-performance skills consist of mixing well-understood control frameworks and learning methods. For instance, there exists a growing number of theoretical control frameworks grounded in second-order dynamics like geometric control [15], [16], Dynamic Movement Primitives, [17], Riemannian Motion Policies [2], [1], [18], and most recently, geometric fabrics [19], [20]. Geometric fabrics generalize classical mechanics and are stable, path consistent, and expressive. We refer the reader to [19] for extended discussion on theoretical differences between geometric fabrics and the related control

[1]Karl Van Wyk, Ankur Handa, Viktor Makoviychuk, Yijie Guo, and Arthur Allshire, and Nathan Ratliff are with Nvidia, USA {kvanwyk,ahanda}@nvidia.com
[2]Arthur Allshire is with University of Toronto, CA

work. All of these methods substantially advance beyond standard joint- or task-space controllers, which typically only enable straightline motions in their targeted spaces [21], [22], [4]. Despite this, a significant portion of policy learning is applied over these simplistic controllers [10], [23], [24], [25], [26]

Built upon generalized nonlinear geometry [27], geometric fabric policies have shown to outperform RMPs [19], [28], DMPs [28], and Koopman Operator policies [29]. RMPs have also been used in RL contexts, leveraging a range of structure imposed by the RMP framework to improve policy performance on reaching tasks [30]. Most recently, an even broader classes of geometric fabrics have been theoretically derived that are easier to construct while retaining all important prior properties [20]. We elect to leverage geometric fabrics in this work because of their empirical performance advancement over prior state-of-the-art methods, their provable stability, and path consistency. Note, RMPs are incredibly broad as defined in [2], and actually encapsulate geometric fabrics as a special subclass with important properties.

Grounded in these recent advances, this work proposes a general framework that combines RL, second-order control frameworks, and physical dynamics to sequence high-performance manipulation skill. Our contributions include:

- A general framework that cascades an RL policy and *behavioral dynamics*: a stacked, second order dynamical system that mixes artificial and real dynamics, shifting the underlying dynamics more favorably.
- A quadratic program with closed-form solution that enables acceleration and jerk constraint handling by second-order systems.
- A novel geometric fabric of the latest form [20] that handles robot joint position constraints, encourages fingertip contact via geometric paths, and opens a force action space for a highly actuated robot hand.
- Instantiates the framework by combining RL, this geometric fabric, and simulation at scale to train dexterous in-hand cube re-orientation skill as described in [8], [6] leading to break-through sim2real performance.

II. Formulation

We reshape the real second-order dynamics of a robot via an artificial second-order dynamical system. These artificial dynamics are constructed from a recently uncovered family of geometric fabrics [20]. This geometric fabric will generate speed-invariant paths through space, automatically handle certain constraints, capture useful, guiding tendencies, and expose an action space. Policies can issue actions in this space that will mix with the guiding fabric, ultimately generating a combined behavior manifested by the real robot. We call this mixing of artificial and real dynamics, *behavioral dynamics*, which is constructed as follows.

A. Forcing Energized Fabrics

We leverage recent theory in fabrics and follow a provably stable subclass thereof as described by Theorem IV.1 in [20]. This fabric is the following stable second order dynamical system

$$\begin{aligned}\ddot{\mathbf{q}}_f = \widetilde{\mathbf{h}}(\mathbf{q}_f, \dot{\mathbf{q}}_f) + \alpha_{\mathcal{L}}(\mathbf{q}_f, \dot{\mathbf{q}}_f)\dot{\mathbf{q}}_f \\ - \mathbf{M}_f^{-1}(\mathbf{q}_f, \dot{\mathbf{q}}_f)\left(\partial_\psi(\mathbf{q}_f) + \mathbf{B}(\mathbf{q}_f, \dot{\mathbf{q}}_f)\dot{\mathbf{q}}_f\right) \\ - \beta(\mathbf{q}_f, \dot{\mathbf{q}}_f)\dot{\mathbf{q}}_f\end{aligned} \tag{1}$$

where $\mathbf{q}_f, \dot{\mathbf{q}}_f, \ddot{\mathbf{q}}_f \in \mathbb{R}^n$ are the position, velocity, and acceleration of the fabric with n dimensions. $\mathbf{M}_f \in \mathbb{R}^{n\times n}$ is the positive-definite system metric (mass), which captures system prioritization (dependencies dropped for brevity). $\widetilde{\mathbf{h}} \in \mathbb{R}^n$ is a fabric, which we make homogeneous of degree 2 in velocity (HD2) to produce geometric paths through space (which can be interpreted as nominal system behavior). $\alpha_{\mathcal{L}} \in \mathbb{R}$ is an energization coefficient which ensures the fabric maintain a certain energy, $\mathcal{L}$. $\partial_\psi \in \mathbb{R}^n$ is the gradient of a potential function and $\mathbf{B} \in \mathbb{R}^{n\times n}$ a positive semi-definite damping matrix, both of which additionally perturb system acceleration from the nominal fabric. These can be used to impose constraints on the system, for instance. Finally, $\beta \in \mathbb{R}^+$ is an additional damping scalar that preserves the fabric geometry and serves to stabilize the system by removing energy.

B. Behavioral Dynamics

The geometric fabric governing the artificial dynamics in (1) can be compactly rewritten as

$$\mathbf{M}_f(\mathbf{q}_f, \dot{\mathbf{q}}_f)\ddot{\mathbf{q}}_f + \mathbf{f}_f(\mathbf{q}_f, \dot{\mathbf{q}}_f) = \mathbf{0} \tag{2}$$

where $\mathbf{f}_f \in \mathbb{R}^n$ is the artificial force. These dynamics are connected to the real dynamics of a robot as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \tau(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{q}_f, \dot{\mathbf{q}}_f, \ddot{\mathbf{q}}_f) \tag{3}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ are the real position, velocity, and acceleration. $\mathbf{M} \in \mathbb{R}^{n\times n}$ and $\mathbf{f} \in \mathbb{R}^n$ are the real robot mass and force (including contact, Centripetal/Coriolis, friction, and gravity forces). The torque control law, τ, connects the artificial and real dynamics together in some way. A specific instantiation of this torque law is discussed in Section II-C.

With the behavioral dynamics above, a policy, $\pi(\cdot)$, can produce a driving force on the fabric by issuing actions $\mathbf{a}$, ($\mathbf{a} \sim \pi(\cdot)$), to some function, $\mathbf{f}_\pi(\cdot)$, as

$$\mathbf{M}_f(\mathbf{q}_f, \dot{\mathbf{q}}_f)\ddot{\mathbf{q}}_f + \mathbf{f}_f(\mathbf{q}_f, \dot{\mathbf{q}}_f) + \mathbf{f}_\pi(\mathbf{a}) = \mathbf{0}, \tag{4}$$

where $\mathbf{f}_\pi(\mathbf{a})$ is interpreted as a driving force on the fabric. Consequently, the time evolution of the fabric state, $(\mathbf{q}_f, \dot{\mathbf{q}}_f)$, is a function of the fabric itself and the control forces produced by the policy. General control forces over second-order dynamical systems is a ubiquitous control input in many contexts including torque control laws for robots [31] and torque control inputs for trajectory optimization [32]. In theory, $\mathbf{f}_\pi(\mathbf{a})$ could destabilize the artificial dynamics, but in practice, sufficiently large $\mathbf{B}$ and β maintains system

stability. Moreover, energy capping methods (Proposition III.3 in [20]) can guarantee stability even with a general driving force coming from $\mathbf{f}_\pi(\mathbf{a})$. We call $\pi(\cdot)$ a Fabric-Guided Policy (FGP).

C. Torque Controller

Typically, the torque control law in (3) is a joint-level proportional-derivative (PD) controller with inverse dynamics compensation. This is *not* the only approach to connecting the two dynamical systems, but one that we most often employ. The joint PD route facilitates tracking control which ultimately means that $||\mathbf{q}_f - \mathbf{q}|| \leq \epsilon_1 \in \mathbb{R}^+$ and $||\dot{\mathbf{q}}_f - \dot{\mathbf{q}}|| \leq \epsilon_2 \in \mathbb{R}^+$. Both ϵ_1, ϵ_2 can be driven arbitrarily small based how much of the inverse dynamics are compensated and the magnitude of the PD gains (see [31] for in-depth analyses). For the particular case of a fully controllable robot arm moving in freespace, one can then see that the geometric fabric effectively replace the real dynamics since $\mathbf{q} \approx \mathbf{q}_f$ and $\dot{\mathbf{q}} \approx \dot{\mathbf{q}}_f$. More generally, the controllable robot states and the associated fabric state will closely match in free-space and separate during contact. This separation induces contact forces, which can be leveraged to perform mechanical work, e.g., object manipulation. The technique of inducing and controlling contact forces via a separation in target states and desired states is standard practice via impedance and admittance control formulations [33], [34]. This particular separation in fabric and actual state is a useful construction for a variety of reasons. First, it still follows the ubiquitous paradigm of RL policies generating joint actions for an underyling PD controller [11], [8], [6], [13], [9]. Second, it is also common in Dynamic Motion Primitive [17], RMP work [2], and tight-tolerance insertions with the PLAI scheme in [10] which positively impacted sim2real efforts.

III. Application to Multi-Fingered Cube Reorientation

We apply the preceding framework to the problem of in-hand cube orientation by a 16-actuator, 4-finger Allegro Hand v4 with hardware setup and vision-based cube pose estimation exactly as detailed in [8] (except [8] used the Allegro Hand v3). This particular problem is quite challenging for many reasons including: 1) hybrid dynamics, 2) complexity of the hand given its geometry, high actuator count, and unmodeled dynamics, and 3) state estimation of the cube. Of course, previous works have surmounted these complexities and have shown strong real-world performance given policies that were trained purely in a physical simulator. We compliment these preceding works by studying the application and effect of our new control framework in this setting and discover important takeaways.

A. Fabrics Design and Policy Action Space

To realize the architecture covered in Section II, we design a forced energized geometric fabric for the Allegro hand following the general form in (1). The components of this fabric are described as follows. For more information on how these components are combined into (1), refer to [2], [19], [20].

1) Attraction: We animate two behavioral elements in the fabric based on manipulation insights. First, encouraging fingertip contact with the cube facilitates its controllability (like strategies in [5]). Second, maintaining inwardly curled fingers around the cube provides caging effects. We imbue the fabric with these behaviors by constructing geometric attractors in two different spaces, $\mathbf{x}$, as $(\mathbf{M}(\mathbf{x}), \ddot{\mathbf{x}}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{x}_g)$, where $\mathbf{x} \in \mathbb{R}^{12}$ for the concatenated fingertip space and $\mathbf{x} \in \mathbb{R}^{16}$ for the configuration space. $\mathbf{M}(\mathbf{x}) = m\mathbf{I}$ is a constant isotropic mass, where $m \in \mathbb{R}^+$. $\ddot{\mathbf{x}} = -k_a \|\dot{\mathbf{x}}\|^2 \tanh(\alpha_a \|\mathbf{x} - \mathbf{x}_g\|) \frac{\mathbf{x}-\mathbf{x}_g}{\|\mathbf{x}-\mathbf{x}_g\|}$, where $k_a \in \mathbb{R}^+$ is a constant attraction gain, α_a is a constant sharpness parameter, and $\mathbf{x}_g$ is a target state in this space. To engender fingertip contact, we set $\mathbf{x}_g = [\mathbf{x}_c^T, \mathbf{x}_c^T, \mathbf{x}_c^T, \mathbf{x}_c^T]^T$ ($\mathbf{x}_c \in \mathbb{R}^3$ is the 3D center of the cube) in the fingertip space. To evoke inwardly curling fingers, we set $\mathbf{x}_g$ to some fixed, curled position in the configuration space. These components partially construct the system metric, $\mathbf{M}_f$, and fully construct the geometric fabric, $\widetilde{\mathbf{h}}$, in (1).

2) Repulsion: To ensure the fabric state respects the upper and lower joint limits of the robot hand ($\overline{\mathbf{q}}, \underline{\mathbf{q}}$), we introduce repulsion forcing fabric terms in an upper joint limit task space, $\mathbf{x} = \overline{\mathbf{q}} - \mathbf{q}$, and lower joint limit task space, $\mathbf{x} = \mathbf{q} - \underline{\mathbf{q}}$. The metric for the fabric term in these spaces is $\mathbf{M}(\mathbf{x}) = \text{diag}\left(\max(-\text{sgn}(\dot{\mathbf{x}}), 0) \frac{k_b}{\mathbf{x}}\right)$, where $k_b \in \mathbb{R}^+$ is a constant gain. Effectively, this is a barrier metric for which a diagonal element $\mathbf{M}_{ii} \to \infty$ as $\mathbf{x}_i \to 0$. The paired acceleration is $\ddot{\mathbf{x}} = \mathbf{g} - b\dot{\mathbf{x}}$, where $\mathbf{g} \in \mathbb{R}^{n+}$ (n is the dimensionality of the configuration space) is constant and $b \in \mathbb{R}^+$ is a constant damping gain. Effectively, the acceleration is positive with damping. This component partially constructs $\mathbf{M}_f$ in (1), and fully constructs ∂_ψ and $\mathbf{B}$ in (1).

3) Energization: The energization coefficient $\alpha_\mathcal{L}$ in (1) is calculated as detailed in Theorem IV.5 in [19] from a configuration-space energy, $\mathcal{L} = \frac{1}{2}\dot{\mathbf{q}}_f^T \dot{\mathbf{q}}_f$, and the fabric $\widetilde{\mathbf{h}}$. This ensures that the geometric fabric itself is energy stable.

4) Geometrically-Consistent Damping: The final damping term β in (1) is set to a constant smaller value ($\beta = 2.5$) during training to facilitate exploration. During deployment, a variety of β levels are tested (see Table II, revealing various levels of sim2real performance). Importantly, this damping term influences the speed of resulting movements in a geometrically consistent manner (see [27] for more discussion), with higher values incurring slower motion. We found it important to slow the motions down for stronger sim2real transfers as detailed in Section III-E.

5) Acceleration and Jerk Handling: Robot controllers like the torque law in (3) typically require that $\mathbf{q}_f$ is sufficiently smooth to protect the actuators, resulting in acceleration and jerk constraints. We can easily accomodate such constraints by formulating the following quadratic program

$$L = \frac{1}{2}(\ddot{\mathbf{q}}_f - \ddot{\mathbf{q}})^T \mathbf{M}_f (\ddot{\mathbf{q}}_f - \ddot{\mathbf{q}}) + \frac{\alpha}{2} \ddot{\mathbf{q}}_f^T \mathbf{M}_f \ddot{\mathbf{q}}_f \qquad (5)$$

where $\alpha \in \mathbb{R}^+$ effectively regularizes $||\ddot{\mathbf{q}}|| \to 0$ while considering $\mathbf{M}_f$. The closed-form solution is

$$(\mathbf{M}_f + \alpha\mathbf{I})\ddot{\mathbf{q}}_f + \mathbf{f}_f = \mathbf{0} \tag{6}$$

where $\mathbf{f}_f = -\mathbf{M}_f\ddot{\mathbf{q}}$. Solving for $\ddot{\mathbf{q}}_f$ produces $\ddot{\mathbf{q}}_f = -(\mathbf{M}_f + \alpha\mathbf{I})^{-1}\mathbf{f}_f$ and we can see that as $\alpha \to \infty$, $||\ddot{\mathbf{q}}_f|| \to 0$. This means that accelerations can be made arbitrarily small on-demand and we can drive them under the acceleration limits. That is, we can solve for a single α such that $|\ddot{\mathbf{q}}_{f,i}| \leq \overline{\ddot{\mathbf{q}}}_i \, \forall \, i$, where $\overline{\ddot{\mathbf{q}}}_i$ is the i^{th} joint acceleration limit.

We can easily extend the above to accommodate joint jerk limits as well via the following time-discretized jerk model (superscripts indicate time index)

$$\dddot{\mathbf{q}}_f^t = \frac{\ddot{\mathbf{q}}_f^{t+1} - \ddot{\mathbf{q}}_f^t}{\Delta t} \tag{7}$$

The largest possible jerk will occur when the next acceleration is the maximum acceleration and the previous acceleration is the minimum acceleration (assuming $\ddot{\mathbf{q}}_f^t = -\overline{\ddot{\mathbf{q}}}_f^t$), or

$$\overline{\dddot{\mathbf{q}}}_f^t = \frac{2\overline{\ddot{\mathbf{q}}}}{\Delta t} \tag{8}$$

Therefore, if jerk must not exceed some limit, $\overline{\dddot{\mathbf{q}}}$, then

$$\frac{2\overline{\ddot{\mathbf{q}}}}{\Delta t} \leq = \overline{\dddot{\mathbf{q}}}. \tag{9}$$

Therefore, we can calculate a single acceleration limit

$$\overline{\ddot{\mathbf{q}}} = \min\left(\overline{\ddot{\mathbf{q}}}, \frac{\Delta t\overline{\dddot{\mathbf{q}}}}{2\overline{\ddot{\mathbf{q}}}}\right) \tag{10}$$

that respects both the original acceleration limit and jerk limit. With this new $\overline{\ddot{\mathbf{q}}}$, we can run the previously detailed scheme for ensuring that both acceleration and jerk constraints are upheld.

6) Action Space: We elect for the RL policy actions to be converted to forces in the concatenated fingertip space resulting in $\mathbf{a} \in \mathbb{R}^{12}$. This force is created by first clamping the actions between $[-1, 1]$ and then scaling them by some positive factor, $\gamma \in \mathbb{R}^+$. This conversion becomes a force in the fingertip space, which is pulled-back to the root of the fabric via this map's Jacobian, i.e., $\mathbf{f}_\pi(\mathbf{a}) = \gamma\mathbf{J}^T(\mathbf{q}_f)\text{clamp}(\mathbf{a}, -1, 1)$ in (4). Interestingly, this RL action space is radically different and results in high policy performance.

7) Numerical Integration: After evaluating the policy force actions $\mathbf{f}_\pi$ and fabric in (4), the resulting fabric acceleration at time t, $\ddot{\mathbf{q}}_f^t$, is forward integrated with an approximate RK2 integration scheme as in [35]. This scheme calculates the next fabric joint position and velocity, $\mathbf{q}_f^{t+1}$ and $\dot{\mathbf{q}}_f^{t+1}$, from the current fabric joint position and velocity, $\mathbf{q}_f^t$ and $\dot{\mathbf{q}}_f^t$, acceleration $\ddot{\mathbf{q}}_f^t$, and timestep Δt as

$$\mathbf{q}_f^{t+1} = \mathbf{q}_f^t + \Delta t\dot{\mathbf{q}}_f^t + \frac{1}{2}\Delta t^2\ddot{\mathbf{q}}_f^t \tag{11}$$

$$\dot{\mathbf{q}}_f^{t+1} = \dot{\mathbf{q}}_f^t + \Delta t\ddot{\mathbf{q}}_f^t \tag{12}$$

Policies issue actions at 30 Hz, while the fabric is forward integrated at 60 Hz, resulting in an integration timestep of $\Delta t = \frac{1}{60}$. The fabric position and velocities are passed as inputs to the torque law in 3, which is a PD controller with gains $k_p = 2$ and $k_d = 0.1$.

B. Reinforcement Learning Setup

We keep the exact same reward terms and their weights as in [8], but completely remove all penalties (action, action delta, and joint velocity penalties) since the fabric layer ensures smooth, safe motion that is within the hardware constraints of the robot. Thus, the fabric layer simplifies reward engineering as well. The neural architectures and sizes for both the value function and policy are the same as in [8], with the single exception that both $\mathbf{q}_f$ and $\dot{\mathbf{q}}_f$ are additionally given as inputs to both the policy and value function. We also use PPO for RL training with the same hyperparameters and automatic domain randomization (ADR) setup in [8], and trained with eight NVIDIA A40 GPUs for about 170 hours of wall-clock time.

C. Cube Disturbance Wrench

We found that policies trained with the force disturbance in [8] often resulted in macro-level behaviors that did not transfer well to the real world. For instance, fingers would full extend at times allowing the cube to roll out of the hand in the real world. To close this sim2real gap, we applied a full wrench disturbance to the cube during training which forced the policies to more frequently establish prehensile-lock on the cube to maintain controllability over the more chaotic cube dynamics. This change ultimately generated significantly better sim2real performance.

For every action step, a new disturbance wrench expressed and applied in cube-centric coordinates, C, is formed (per environment) with 10 % chance as

$${}^C\mathbf{w}_d = \begin{bmatrix} {}^C\mathbf{f}_d \\ {}^C\tau_d \end{bmatrix} = \begin{bmatrix} c_1 m {}^C\hat{\mathbf{f}}_d \\ {}^C\mathbf{r} \times {}^C\mathbf{f}_d \end{bmatrix} = \begin{bmatrix} c_1 m {}^C\hat{\mathbf{f}}_d \\ c_2^C\hat{\mathbf{r}} \times c_1 m {}^C\hat{\mathbf{f}}_d \end{bmatrix} \tag{13}$$

where $c_1 \in \mathbb{R}^+$ is an acceleration constant, $m \in \mathbb{R}^+$ is the randomized mass of the cube, and ${}^C\hat{\mathbf{f}}_d \in \mathbb{R}^3$ is randomly sampled direction of unit magnitude. Finally, $c_2 = \frac{\sqrt{3}}{2}0.065$ is the radius of a sphere centered with the cube (cube side length is 0.065 m), and ${}^C\hat{\mathbf{r}} \in \mathbb{R}^3$ is a random direction emanating from the origin of the cube. If a new disturbance wrench is not sampled by chance, then the previously sampled wrench is applied. Maintaining a consistent disturbance wrench for several time steps allows for greater influence over the cube's motion.

D. Training Performance

We train five random seeds of both FGP and DeXtreme policies on the training setup as described in [8] with a few

Policy	Seed	Time to -1.0 npd (hours)	Time to -0.5 npd (hours)	Final npd
FGP (ours)	1	26.7	113.6	-2.1e-1
	2	17.3	139.1	-3.2e-1
	3	35.6	N/A	-9.3e-1
	4	20.84	121.567	-2.2e-1
	5	26.5	131.457	-2.6e-1
DeXtreme (new)	1	16.6	38.4	-8.0e-2
	2	15.7	32.5	1.8e-3
	3	31.3	52.6	3.0e-2
	4	15.4	51.7	-8.2e-3
	5	15.9	33.2	4.7e-3

TABLE I: Entropy levels obtained by both FGP and DeXtreme policies across random seeds.

modifications. For the FGPs, we augment the inputs to the policy and value function as described in Sections III-B and also apply the new disturbance wrench as in III-C. For the Dextreme policies, we apply the new disturbance wrench (called DeXtreme (new)). As shown in Table I, DeXtreme policies train faster than FGP policies, particularly in the amount of time required to achieve -0.5 nats per dimension (npd) as defined in [8]. FGPs also produce lower entropy levels than DeXtreme policies in the allotted training time. Interestingly, neither the FGPs or DeXtreme policies are converged after 170 hours of training and higher levels of entropy could be obtained if trained for longer. Overall, it is harder to learn smoother policies with RL as it effects exploration [36]. However, high-performing FGP policies can still be trained in about 1 week.

E. Real-World Performance

We analyze policy performance across three performance metrics: consecutive success (CS), rotations per minute (RPM) (see Table II), and action noise rejection. CS is an established metric for this task and counts the consecutively successful rotations with a rotational goal tolerance of 0.4 rad in the real world [6], [8]. RPM is the CS for a run divided by the duration of the run measured in minutes. Overall, CS captures reliability of the manipulation skill and RPM captures its solution speed. Finally, action noise rejection depicts the level of attenuation present in the action signals in excess of 5 Hz.

The top two highest performing policies along the CS index are the FGP with $\beta = 40$ and the DeXtreme (new) policy with an integrator in the torque law. Interestingly, the FGP policy improves with increasing damping levels up to $\beta = 40$, after which, performance degrades. We believe this performance trend is due to unmodeled dynamics effects in the simulator, latency effects, and the fact that β damps the fabric in a geometrically consistent manner (finger paths are relatively unchanged with changes in β). The DeXtreme (new) policy had two runs with over 600 CS which significantly increased the mean CS. However, the resulting median of 70 was less than that of the FGP. Since elements of the physical setup have changed since DeXtreme [8] (new hand, motors and EMA of 0.05 versus 0.1), we re-evaluated the original DeXtreme policy. Despite these changes, we found that the original DeXtreme policy performed consistently (e.g., CS mean of 29.0 vs 27.8). Ultimately, both DeXtreme (new) and FGP policies are very high performing with several runs well over 100 CS, significantly surpassing all prior results [6], [7], [8].

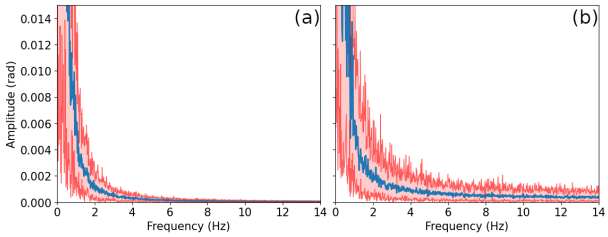

Fig. 3: Spectral content of target joint angles generated by the (a) FGP and (b) DeXtreme policies indicating greater noise attenuation in FGP control. Blue curve is mean amplitude, red curves are the minimum and maximum amplitudes.

Along the RPM index, the FGP policy significantly outperforms all DeXtreme policies. Interestingly, the RPMs across β levels do not fluctuate much and higher β leads to greater precision along this metric. From observation, we see that finger movement is indeed faster with smaller β levels (as expected), but RPMs do not improve because more manipulation errors are present. Qualitatively, the policies look more meticulous with increased β, and quantitatively, enable greater precision in solve rates and higher CS. The DeXtreme policies had much lower RPMs, likely due to the EMA filter possessing a much smaller cut-off frequency than what is set during training (EMA factor of 0.15 versus 0.05). Small EMA factors stifle exploration and learning progress during RL training, so larger values are used during training.

Finally, policies should not generate unnecessary noise in their actions as this can accelerate wear-and-tear on real robot hardware. Some suggest frequencies under 5 Hz, covering a large spectrum of robotics applications [37]). We inspect the spectral content of the target joint angles produced by the FGP at $\beta = 40$ and DeXtreme (new) policies by passing a one minute recording of these signals through the Fast-Fourier Transform (FFT) per joint (see Fig. 3). As shown, most of the action signals emit frequencies with high amplitudes below 2 Hz for both policies. However, the DeXtreme policy produces significantly higher magnitude action noise in excess of 5 Hz, indicating stronger noise rejection capabilities of the FGP. This is particularly interesting considering that the raw RL actions $\mathbf{a}$ in (4) for the fabric are *bang-bang*. Smoothness should be adopted as a critical performance index for RL policies as they become more widely adopted with significant implications on the running costs of hardware maintenance and loss of productivity for repairs.

IV. Discussion

Training an RL policy over a fabric-based behavioral dynamics layer engendered new state-of-the-art performance for in-hand cube re-orientation. The FGP at $\beta = 40$ more than tripled performance along the CS metric when compared to the previous state-of-the-art DeXtreme model. Against the latest DeXtreme model, the FGP had significantly lower mean CS, but higher median CS. The FGP also had the

Policy	β	CS Mean	CS Median	RPM Mean	RPM Median	CS capped at 50
FGP	2.5	24.9 ± 11.4	20.0	8.9 ± 1.0	**9.3**	22.2 ± 7.8
	10	35.4 ± 17.1	25.0	8.3 ± 1.2	9.0	27.7 ± 9.5
	20	32.6 ± 14.0	29.5	**9.9 ± 1.3**	9.1	28.3 ± 8.7
	30	57.6 ± 22.1	67.0	9.0 ± 0.5	9.1	36.4 ± 9.7
	40	**94.1 ± 29.4**	**85.5**	**9.4 ± 0.3**	**9.4**	**47.4 ± 3.1**
	50	79.6 ± 31.1	57.5	8.7 ± 0.5	8.9	**43.0 ± 6.6**
DeXtreme* (new)	N/A	**244.6 ± 140.1**	**70.0**	7.6 ± 0.4	7.6	42.3 ± 7.7
DeXtreme* (previous, rerun) [8]	N/A	29.0 ± 18.7	15.5	6.5 ± 1.6	5.7	21.1 ± 10.2
DeXtreme* (previous) [8]	N/A	27.8 ± 19.0	14.0	N/A	N/A	23.1 ± 9.4
OpenAI [6]	N/A	15.2 ± 14.3	11.5	N/A	N/A	15.2 ± 14.3

TABLE II: Consecutive success (CS) and rotations per minute (RPM) metrics for vision-based policies across 10 runs (± indicates 95 % confidence interval following a t-distribution.) (*) Indicates integrator present in torque law during deployment.

fastest solve times with the highest and most consistent RPM. Finally, the FGP produced very little action noise, if any, with spectral amplitudes nearly zero above 5 Hz, whereas DeXtreme policies still admitted frequencies above 5 Hz despite its heavy usage of low-pass filtering. Note, we do not focus on generalizing across object geometries as in [38], [39], but instead, focus on generalizing across other aspects of system dynamics and searching for methods that push skill performance towards industrial-grade levels.

In general, the sim2real transfers among FGP and DeXtreme policies are variable despite high ADR entropy and CS among seeds in simulation. We believe domain randomization is a necessary but insufficient condition for maximizing real-world performance. In reality, different training runs evolve different macro-level behaviors. Since the reward functions under-specify the desired behavior and optimization is also subject to many local minima, policies can converge onto a continuum of different strategies for solving the problem in simulation. However, some strategies transfer more strongly, a phenomenon also expressed in [8], [40]. For example, policies that tended to cage the cube more often transferred more strongly than policies that too frequently fully extended the fingers, allowing the opportunity for the cube to roll out of the hand. Training DeXtreme policies for much longer than in [8] resulted in much higher ADR entropy and very high CS in the real world, providing evermore credence towards our infrastructure, approach, and large-scale training runs. Interestingly, none of the FPG and DeXtreme policies ever converged during RL training. Further increasing the duration of the training runs could amplify real-world performance even more as the policies continue to improve their generalization across system dynamics.

Smoother policies impede RL progression and counters RL's preference towards bang-bang control [36]. This was observed in DeXtreme, which is the reason for higher EMA factors during training. Similarly, FGP policies train more slowly than DeXtreme, which is very likely due to their even smoother action profiles (see Section III-E for details). In general, advancements need to be made to improve RL for smooth policies which could include automatically optimizing over various hyperparameters [41], employing more effective RL algorithms, investigating alternative exploration noise [42], or leveraging priors in some form [40].

Designing and tuning a geometric fabric does require deep understanding of the control method, experience, and vectorized tooling to enable training at scale. There is a rich history of practitioners succeeding at designing these second order systems [2], [19], [1], [18], [30] and the recent simplifications to geometric fabrics [20] further ease the process. Moreover, the fabric terms themselves are not overly exotic and are often very compact and interpretable equations as shown in Section III. With the right software tooling, an experienced practitioner can design and tune a fabric within a few hours, which is significantly faster than iterating over more complicated reward functions and extended RL training runs. Besides, optimizers always have the burden of decoding more complex reward functions directly, resulting in local minima and deficiencies in optimization fidelity. Instead, fabrics can directly encode behavior in closed-form, simplifying reward engineering and providing stronger guarantees on some behavioral elements. For instance, the geometric fabric herein upheld joint position, acceleration, and jerk constraints. The FGP directly benefits from the inherent constraint handling of the fabric and does not require additional low-pass filtering, enabling fast and safe motion. Additionally, the fabric generated fingertip paths towards the cube, guiding the RL policy to more strongly leverage the fingertips during manipulation. The geometric nature of this tendency also enabled deploying the fabric at different levels of β, allowing its optimization for maximizing real-world performance. Note, the design space for geometric fabrics is very large and different designs can influence FGP performance.

V. Conclusion and Future Work

We propose a general paradigm for cascading an RL policy, an artificial dynamical system, and real system dynamics together. We instantiate this paradigm with a powerful second-order control method, geometric fabrics, reinforcement learning, and vectorized simulation at scale to achieve state-of-the-art policy performance for the established cube re-orientation task by a dexterous, high DoA robot hand. Future work includes applying the approach more broadly across different robot platforms, different tasks, and different fabric designs. We will also consider other optimization and planning methods in search of maximizing policy performance.

REFERENCES

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

[2] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.

[3] T. Wimboeck, C. Ott, and G. Hirzinger, "Passivity-based object-level impedance control for a multifingered hand," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 4621–4627.

[4] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[5] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly, "Comparative peg-in-hole testing of a force-based manipulation controlled robotic hand," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 542–549, 2018.

[6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[7] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[8] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam *et al.*, "DeXtreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[9] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without seeing: Towards in-hand dexterity through touch," *arXiv preprint arXiv:2303.10880*, 2023.

[10] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang, "Industreal: Transferring contact-rich assembly tasks from simulation to reality," *Robotics: Science and Systems*, 2023.

[11] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[12] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through multi-task reinforcement learning," *arXiv preprint arXiv:2302.09450*, 2023.

[13] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[14] H. Zhang, Y. Yuan, V. Makoviychuk, Y. Guo, S. Fidler, X. B. Peng, and K. Fatahalian, "Learning physically simulated tennis skills from broadcast videos," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–14, 2023.

[15] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer, 2019, vol. 49.

[16] L. Susskind, "The theoretical minimum: Classical mechanics," 2011.

[17] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[18] A. Li, C.-A. Cheng, B. Boots, and M. Egerstedt, "Stable, concurrent controller composition for multi-objective robotic tasks," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1144–1151.

[19] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, 2022.

[20] N. D. Ratliff and K. Van Wyk, "Fabrics: a foundationally stable medium for encoding prior experience," *arXiv preprint*, 2023.

[21] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[22] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[23] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," *arXiv preprint arXiv:2305.16309*, 2023.

[24] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[25] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju *et al.*, "Robocat: A self-improving foundation agent for robotic manipulation," *arXiv preprint arXiv:2306.11706*, 2023.

[26] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[27] N. D. Ratliff, K. Van Wyk, M. Xie, A. Li, and M. A. Rana, "Generalized nonlinear and finsler geometry for robotics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 206–10 212.

[28] M. Xie, A. Handa, S. Tyree, D. Fox, H. Ravichandar, N. D. Ratliff, and K. Van Wyk, "Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration," in *Conference on Robot Learning*. PMLR, 2023, pp. 1355–1367.

[29] Y. Han, M. Xie, Y. Zhao, and H. Ravichandar, "On the utility of koopman operator theory in learning dexterous manipulation skills," *arXiv preprint arXiv:2303.13446*, 2023.

[30] A. Li, C.-A. Cheng, M. A. Rana, M. Xie, K. Van Wyk, N. Ratliff, and B. Boots, "Rmp2: A structured composable policy class for robot learning," *Robotics: Science and Systems*, 2021.

[31] A. Behal, W. Dixon, D. M. Dawson, and B. Xian, *Lyapunov-based control of robotic systems*. CRC Press, 2009, vol. 36.

[32] Z. Manchester and S. Kuindersma, "Variational contact-implicit trajectory optimization," in *Robotics Research: The 18th International Symposium ISRR*. Springer, 2020, pp. 985–1000.

[33] C. Ott, R. Mukherjee, and Y. Nakamura, "Unified impedance and admittance control," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 554–561.

[34] E. Magrini, F. Flacco, and A. De Luca, "Control of generalized contact motion and force in physical human-robot interaction," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2298–2304.

[35] N. Gruver, M. Finzi, S. Stanton, and A. G. Wilson, "Deconstructing the inductive biases of hamiltonian neural networks," *arXiv preprint arXiv:2202.04836*, 2022.

[36] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus, "Is bang-bang control all you need? solving continuous control with bernoulli policies," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 209–27 221, 2021.

[37] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value iteration in continuous actions, states and time," *arXiv preprint arXiv:2105.04682*, 2021.

[38] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*. PMLR, 2022, pp. 297–307.

[39] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," 2022.

[40] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[41] A. Petrenko, A. Allshire, G. State, A. Handa, and V. Makoviychuk, "DexPBT: Scaling up dexterous manipulation for hand-arm systems with population based training," *Robotics: Science and Systems*, 2023.

[42] O. Eberhard, J. Hollenstein, C. Pinneri, and G. Martius, "Pink noise is all you need: Colored noise exploration in deep reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.