Neural Geometric Fabrics: Efficiently Learning High-Dimensional Policies from Demonstration

Anonymous Author(s)
Affiliation
Address
email

Abstract: Learning dexterous manipulation policies for multi-fingered robots has been a long-standing challenge in robotics. Existing methods are limited to simulation, rely on extensive interactions with the environment, have considerable computational costs, or only involve a floating hand. In this work, we develop a *structured* approach to sample-efficient learning of dexterous manipulation skills from demonstrations by leveraging recent advances in robot motion generation and control. Specifically, our policy structure is induced by Geometric Fabrics - a recent framework that generalizes classical mechanical systems to allow for flexible design of expressive robot motions. To avoid the cumbersome manual design required by existing motion generators, we introduce *Neural Geometric Fabric (NGF)* - a framework that learns Geometric Fabric-based policies from data. NGF policies are provably stable and capable of encoding speed-invariant geometries of complex motions in multiple task spaces simultaneously. We demonstrate that NGF can learn to perform a variety of dexterous manipulation tasks on a 23-DoF hand-arm physical robotic platform purely from demonstrations. Results from comprehensive comparative and ablative experiments show that NGF's structure and action spaces help learn acceleration-based policies that consistently outperform state-of-the-art baselines, like Riemannian Motion Policies (RMPs), and other commonly used networks, such as feed-forward and recurrent neural networks. More importantly, we demonstrate that NGF does *not* rely on often-used and expertly-designed operational-space controllers, promoting an advancement towards efficiently learning safe, stable, and high-dimensional controllers.

Keywords: Imitation Learning, Dexterous Manipulation

1 Introduction

Autonomous and robust dexterous manipulation capabilities could enable robots to perform a wide range of tasks, such as opening a door and using tools, in a world built by and for humans. However, dexterous manipulation involving multi-finger hands continues to be a long-standing challenge in robotics, with human-level dexterity remaining a distant target. Indeed, there are numerous factors contributing to this state of affairs, such as complex dynamics, high-dimensional action spaces, and the difficulties involved in designing bespoke controllers that do not generalize to either novel tasks or variations of the same task, such as new object poses or different targets.

Recent efforts in robot learning have aimed to study and tackle the challenges that plague autonomous dexterous manipulation [1–9]. Specifically, recent learning-based techniques show promise by generating appropriate robot behavior for a variety of tasks without the need for hand-crafted motion policies. However, most existing approaches suffer from one or more of the following disadvantages: i) high computational cost, ii) low sample efficiency, iii) lack of stability guarantees, iv) cumbersome manual design of operational-space controllers, and more importantly v) limited to simulation or hand control only. We provide a detailed discussion of related work in Section 2.

In this work, we tackle the challenge of efficiently learning high-dimensional dexterous manipulation policies for a hand-arm system from demonstrations. We are concerned with learning policies that can generate stable and smooth motions for tasks requiring dexterity, such as grasping and manipulating an object with a multi-fingered robotic hand. Unlike much of prior work that relies on

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

Figure 1: Three tasks used for learning including (a) grasping and lifting a sugar box, (b) touching a target point before pushing a coffee can over it, and (c) extracting a tool tray from a toolbox.

operational-space controllers, we are interested in learning policies that directly and simultaneously coordinate all the *joint-space* degrees of freedom of both the arm and the hand.

Inspired by the growing evidence in support of structured policies for manipulation (e.g., [10, 11]), we propose an algorithm to learn highly-structured and generalizable policies for dexterous manipulation from a limited number of demonstrations. Specifically, we leverage the structure imposed by *Geometric Fabrics* [12], a recent framework that generalizes classical geometric control (e.g., operational space control) using tools from differential geometry, to facilitate stable and flexible robot motion generation [13]. Geometric fabrics are stable second-order dynamical systems, which encode behaviors as speed-invariant paths (via geometries) and generate acceleration actions to generate intended motions. Further, learning to encode dynamical systems via acceleration is known to be a significantly beneficial form of inductive bias in policy learning [14]. However, designing these fabrics for complex dexterous manipulation tasks requires considerable expertise and effort.

To circumvent the need for significant expertise and painstaking human effort in designing Geometric Fabrics for dexterous manipulation tasks, we introduce *Neural Geometric Fabrics* (NGF), a framework to efficiently learn components of Geometric Fabrics, such as subtask maps, geometries, and priority metrics. We discuss details of our architecture and learning pipeline in Section 4.

The primary contributions of this paper are: 1) a novel framework to learn Geometric-Fabrics-based policies directly from demonstrations, removing the need for manual design and expertise; 2) a thorough investigation of the benefits of additional structure imposed by Geometric Fabrics over existing approaches when learning dexterous manipulation skills from demonstrations; and 3) to the best of our knowledge, this is the first framework to train and deploy policies that directly control all joints of the highly-actuated arm-hand system in the real world without relying on manually-designed operational-space controllers in elevated actions spaces.

We evaluate the effectiveness of our approach on a 23-DoF physical hand-arm robot platform in the real world across three different tasks (see Fig 1), and compare its performance against several baselines inspired by the state-of-the-art in learning from demonstration. Our results show that the policies learned with our structured approach consistently outperform baseline methods with respect to metrics measuring task success, safety of deployment, and sample efficiency.

2 Related Work

A variety of approaches have attempted to tackle the challenges of learning dexterous manipulation polices. Here, we discuss how our approach relates to a few different categories of existing methods.

Reinforcement Learning (RL) approaches [15, 16] learn policies to solve manipulation tasks through repeatedly interacting with environments guided by a reward signal which indicates the performance of a trial. Indeed, deep RL-based methods have generated several impressive results [7–9, 17]. However, most RL algorithms suffer from poor data efficiency. Recent approaches have combined imitation learning (IL) with RL in efforts to improve data efficiency [4–6]. However, these methods are often limited to simulation as collecting exploratory data can be expensive, time-intensive, and potentially damaging to the hardware in real-world robotic systems.

Imitation learning (IL) overcomes the challenges of RL by providing a more sample-efficient framework to learn manipulation skills [3–5], but at the cost of expert demonstrations. Behavior cloning (BC) [18, 19] is the simplest way to learn a policy from demonstrations, but it often suffers from *covariate shift* [20], addressing which requires an oracular expert to monitor the policy roll-out and provide additional demonstrations at newly encountered states [20–22]. Despite the recent advances in imitation learning [1, 2], data efficiency is still not satisfactory, hindering the learning of dexterous manipulation skills with multi-fingered robotic hands. Given these observations and the considerable cost of collecting real-world data, it is critical to develop robot learning frameworks that can learn to generalize from less data. We demonstrate that our structured approach enables the learning of complex dexterous manipulation skills purely via imitation.

Exploiting structures in robotic manipulation problems has shown to improve data efficiency in policy learning even though structured policy parameterizations are more restricted in representation power than their non-structured counterparts [23]. This is due to the fact that, if the restrictions provide appropriate inductive biases for the task of interest, structured approaches provide numerous advantages, such as provable stability, safety guarantees, and improved efficiency, without adversely impacting expressivity. Indeed, there are several successful examples of learning structured policies. The approaches in [24–28] leverage structures from control theory, where learned policies meet optimal control assumptions to provide stability guarantees. Dynamical system based approaches [29–39] learn policies by imposing structures or constraints that enforce stability and convergence while imitating the demonstrations. Recently, it was shown that *Riemannian Motion Policies (RMPs)* [10, 11, 40, 41] can encode the geometric structure of manipulation tasks using structured second-order dynamical systems. One can view the proposed NGFs as specialized versions of RMPs that are stable and capable of encoding more complex behaviors as geometries, providing a more appropriate inductive bias for dexterous manipulation.

The choice of action spaces plays a significant role in learning manipulation policies [23], with the most popular choice being the end-effector pose space [2, 17, 39, 42] as it is more directly relevant to the task and can generalize across different robot kinematics [43, 44]. These approaches rely on an operational space controller to map these desired signals into the joint space. However, it is not trivial to design such a controller for highly kinematically-redundant robots, such as multi-fingered hands. In contrast, our method does not require any hand-crafted operational-space controllers as it leverages the robot's kinematic model to directly learn in multiple actions spaces simultaneously.

3 Background: Geometric Fabrics

Geometric fabrics are the latest incarnation of research in robot motion generation and control as defined by second order equations. Geometric fabrics build upon the more general, intuitive, and flexible aspects of Rimennian Motion Policies (RMPs), but retain the stability guarantees offered by classical mechanical systems (which are often leveraged in control as geometric or operational-space control.) In fact, Geometric Fabrics generalize classical mechanical systems and have been shown to outperform RMPs [12]. Geometric fabrics provide a formal mathematical framework for building control policies and primarily encode behavior as a set of nonlinear *geometries of paths*. This geometrically consistent behavior is known as the nominal behavior of the system and acts as an optimization medium over which additional policies derived from potential functions and damping can be applied to further shift the resulting behavior. Policies of this design are guaranteed to be stable, i.e., the system comes to rest at the minimum of the net potential acting on the system.

3.1 Nonlinear Geometries

A generalized nonlinear geometry is an acceleration policy $\ddot{\mathbf{x}}+\pi(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{0}$ where π is *homogeneous of degree* 2 (HD2), meaning for any $\lambda \geq 0$ we have $\pi(\mathbf{x},\lambda\dot{\mathbf{x}}) = \lambda^2\pi(\mathbf{x},\dot{\mathbf{x}})$. The HD2 property ensures that every integral curve starting from a given task-space position $\mathbf{x}_0$ with velocity $\dot{\mathbf{x}}_0 = \eta\widehat{\mathbf{n}}$ will follow the same path. The geometric consistency property turns π into a *geometry of paths*. From a imitation learning perspective, this additional structure promotes learning speed-invariant *paths* through space instead of *trajectories*, which are sensitive to the speed of travel. We postulate the added structure will be beneficial for sample-efficiency in policy learning.

3.2 Geometric Fabrics: encode nominal behaviors

A Geometric Fabric is a system that evolves according to $\mathbf{M}(\mathbf{q}, \dot{\mathbf{q}})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}$ in root coordinates $\mathbf{q}$, where $\ddot{\mathbf{q}} = -\mathbf{M}^{-1}\mathbf{f}$ is a nonlinear geometry that defines a nominal behavior and $\mathbf{M}$ is the priority metric (we drop function variables for notational simplicity). As discussed in Theorem IV.5 of [12], we can energize this Geometric Fabric to conserve energy: $\ddot{\mathbf{q}} = -\mathbf{P}_e\left[\mathbf{M}^{-1}\mathbf{f}\right]$, where $\mathbf{P}_e = \mathbf{M}^{\frac{1}{2}}\left[\mathbf{I} - \hat{\mathbf{v}}\hat{\mathbf{v}}^T\right]\mathbf{M}^{-\frac{1}{2}}$ is a metric-weighted projection matrix with $\mathbf{v} = \mathbf{M}^{\frac{1}{2}}\dot{\mathbf{q}}$ and $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$, which projects acceleration to the orthogonal space of velocity, hence conserves energy by performing no work. The energized fabric defines a nominal behavior independent of a specific task.

3.3 Optimize a potential: solve a task

The nominal behavior can be pushed away from by a potential function to achieve task goals, and damping dissipation ensures convergence. The desired equation of motion of the system then becomes: $\ddot{\mathbf{q}}_d = -\mathbf{P}_e\left[\mathbf{M}^{-1}\mathbf{f}\right] - \mathbf{M}^{-1}\partial_{\mathbf{q}}\psi(\mathbf{q}) - \mathbf{B}\dot{\mathbf{q}}$, where $\psi(\mathbf{q})$ is a potential function, and $\mathbf{B}$ is a positive definite damping matrix. The potential function induces kinetic energy, resulting in motion, and together with damping, drives the system towards the potential's local minima, where motion ultimately comes to rest at the minima of the potential function, hence its optimization.

3.4 Policy Design with Geometric Fabrics

For robot manipulators, a desired behavior may involve coordinated motion of different parts. Geometries can be added to spaces of a transform tree [45] for the modular design of composite behaviors. We can construct a transform tree with root node in the configuration space $\mathcal{C}$ (typically joint space) of the robot, and add a leaf node in each subtask space $\mathcal{T}_k$ in which the task is more convenient to be described. For instance, a goal-reaching task can be defined in the 3-D Euclidean space describing the position of the end-effector in relation to the goal. We denote subtask spaces $\{\mathcal{T}_k\}_{k=1}^{K}$, where K is the number of subtasks. Let $\phi_k : \mathcal{C} \to \mathcal{T}_k$ be the mapping from the configuration space $\mathcal{C}$ to subtask space $\mathcal{T}_k$. An example of ϕ_k is the forward kinematic mapping, where the subtask space $\mathcal{T}_k$ is the end-effector frame. We then populate each node with a priority metric and geometry pair, $[\mathbf{M}_k, \pi_k]$, which are pulled-back and combined at the root, resulting in a desired configuration space policy $\ddot{\mathbf{q}}_d = \pi_{\mathcal{C}}(\mathbf{q}, \dot{\mathbf{q}})$. In the case where we only have an end-effector space, solving for $\ddot{\mathbf{q}}_d$ is equivalent to a pseudo-inverse as described in [46]. We include additional details about composing policies from sub-task spaces in Appendix A.

4 Neural Geometric Fabrics

In this section, we introduce our NGF policy architecture and its learnable components. We illustrate our design process in Fig. 2, and elaborate on each step in the sections below.

4.1 Define subtask spaces

We begin by defining two subtask spaces: (1) A 3-D Euclidean space describing the position of the palm in relation to the object, which is stacked with an Eigenspace of the hand. This map is defined as $\phi_1 : \mathbf{q} \mapsto [\texttt{fk}^T(\mathbf{q}) - \mathbf{x}_o^T,\ \texttt{pca}^T(\mathbf{q})]^T \in \mathbb{R}^{8\times 1}$, where $\texttt{fk}(\mathbf{q})$ is the forward kinematics mapping to a palm point and $\mathbf{x}_o$ is the object position. $\texttt{pca}(\mathbf{q})$ is a linear map of the hand, inspired by the concept of Eigengrasp [47], where we first apply Principle Component Analysis (PCA) to the 16-DoF hand joints data, which reduces the 16-DoF joint space to the 5-DoF Eigenspace while the first five principle components account for more than 95% of the variance. This stacked space directly shares information between the arm and hand, allowing them to coordinate properly. (2) The full configuration space of the robot, $\phi_2 : \mathbf{q} \mapsto \mathbf{q} \in \mathbb{R}^{23\times 1}$. The policy in this space additionally shapes the configuration space behavior, filling in the nullspace of the preceding space.

4.2 Define learnable components in subtask spaces

Geometric policies: A priority metric and geometry pair in the stacked space ϕ_1 is defined as $[\mathbf{M}_1, \pi_1]$, encoding object-reaching and coordinated palm-finger behavior. Another priority metric and geometry pair $[\mathbf{M}_2, \pi_2]$ is defined in the configuration space, ϕ_2, that serves as a residual policy, shaping the configuration space behavior and ensuring the combined Geometric Fabric is full rank.

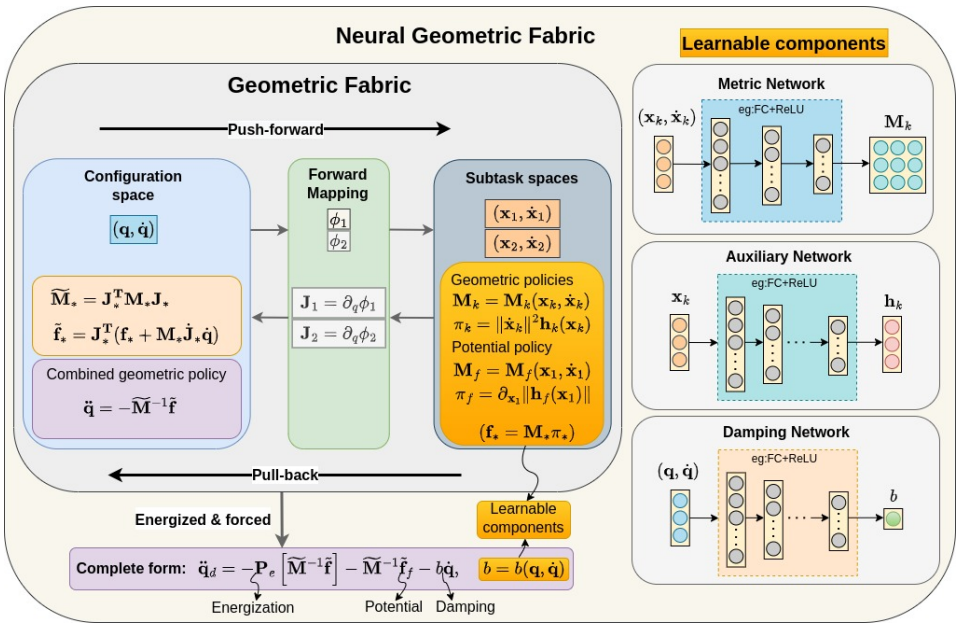

Figure 2: We parameterize NGFs with neural networks and construct them as follows: (1) define each subtask space with a forward mapping ϕ_k that maps coordinates $\mathbf{q}$ in the configuration space into coordinates $\mathbf{x}_k$ in the k-th subtask space, (2) define a geometric policy pair $[\mathbf{M}_k, \pi_k]$ in each subtask space $\mathbf{x}_k$, e.g., $k = 1, 2$, and define a potential policy pair $[\mathbf{M}_f, \pi_f]$ in the desired subtask space, e.g., $\mathbf{x}_1$, (3) pull-back the geometric and potential policy pairs into the configuration space, (4) combine the geometric policies via a metric-weighted average, (5) energize the combined policy (project orthogonal to the direction of motion with $\mathbf{P}_e$) to create a collection of energy-preserving paths (the Geometric Fabric), and (6) force the Geometric Fabric with a potential defined by $[\mathbf{M}_f, \pi_f]$ and damp via b applied along $\dot{\mathbf{q}}$, which ensures convergence to the potential's minima. Note that we parameterize the geometric policy pairs $[\mathbf{M}_k, \pi_k]$, the potential policy pair $[\mathbf{M}_f, \pi_f]$, and the damping scalar b with network networks and learn them from data. We can also define ϕ_k with neural networks if desired.

Potential and damping: We additionally add another priority metric and acceleration-based potential policy $[\mathbf{M}_f, \pi_f]$ (π_f is purely a function of position) in the stacked space ϕ_1 to: 1) maintain bounded system energy levels while 2) induce motion (energized geometries cannot induce motion by themselves when at rest), and 3) optimize the corresponding potential function. Finally, a strictly positive damping scalar b in the configuration space, ϕ_2, is also learned and applied that removes energy from the system, ensuring convergence to the minima of the potential function.

All learnable components, including $[\mathbf{M}_*, \pi_*]$ (π_* is constructed with $\mathbf{h}_*$ and $* = 1, 2, f$) and the damping scalar b, are defined with neural networks. To ensure the priority metrics are positive definite, we follow the architecture described in [11]. See Appendix B for detailed parameterizations.

4.3 Complete form

Using the pullback and combination operations as defined in Appendix A, the two geometric policies and potential-based policy are combined at the root along with damping and energization. With the total metric, $\widetilde{\mathbf{M}} = \widetilde{\mathbf{M}}_1 + \widetilde{\mathbf{M}}_2 + \widetilde{\mathbf{M}}_f$, total geometric force, $\widetilde{\mathbf{f}} = \widetilde{\mathbf{f}}_1 + \widetilde{\mathbf{f}}_2$, and potential force $\widetilde{\mathbf{f}}_f$, the desired NGF acceleration takes the form $\ddot{\mathbf{q}}_d = -\mathbf{P}_e \left[\widetilde{\mathbf{M}}^{-1} \widetilde{\mathbf{f}}\right] - \widetilde{\mathbf{M}}^{-1} \widetilde{\mathbf{f}}_f - b\dot{\mathbf{q}}$, where $\mathbf{P}_e = (\mathbf{I} - \hat{\dot{\mathbf{q}}} \hat{\dot{\mathbf{q}}}^T)$ energizes the fabric with a squared velocity norm energy (see equation (10) of [12] concerning this energization), and $b \in \mathbb{R}^+$ is a damper. $\ddot{\mathbf{q}}_d$ is time integrated to produce $\mathbf{q}_d$ and $\dot{\mathbf{q}}_d$, the *command* trajectory for the robot to follow (see algorithms in Appendix C).

5 Learning Policies from Demonstrations

Given the policy architecture in Section 4, here we describe how we train NGF-based policies using demonstrations to solve manipulation tasks on a 23-DoF hand-arm system system (7-DoF arm and a

16-DoF multi-fingered hand). We used the DexPilot teleoperation system [48] to collect task-solving demonstrations, and trained a policy to replicate the demonstrated joint position trajectories.

5.1 Problem Statement

Consider a deterministic discrete-time system with transition model $s_{t+1} = f(s_t, a_t)$ where $s_t \in \mathbb{R}^n$ is the state, and $a_t \in \mathbb{R}^m$ is the action of the system, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system transition function. We want to learn an NGF policy $\pi_\theta : s \mapsto \pi_\theta(s)$, $\Pi := \{\pi_\theta : \theta \in \Theta\}$, given N trajectory demonstrations $\{\tau^{(i)}\}_{i=1}^N$. Each demonstration is defined as a sequence of states, denoted $\tau^{(i)} := \{s_t^{(i)}\}_{t=0}^{T_i}$, where T_i is the horizon for the ith demonstration. Note, the system transition model we refer to here is simply Euler integration defined by Eq. (7) in Appendix D.

5.2 Policy Optimization

To mitigate the distribution shift issue involved in behavior cloning, we formulate the policy learning problem as a multi-step prediction error optimization problem, in which we learn a policy that can reproduce the demonstrated behavior by minimizing the deviation between the demonstrated trajectory and trajectory produced by a learned policy under the system transition function f. That is, given a transition function $s_{t+1} := f(s_t, \pi_\theta(s_t))$ for each policy, we can rollout a policy trajectory and perform back-propagation-through-time [49, 50] with the Adam optimizer [51] to improve the policy parameters. Importantly, the states here are the *commanded* states during demonstration. More details on policy optimization and the exact loss function are included in Appendix D.

6 Experiments

Tasks: We studied the effect of i) policy structure and ii) the choice of action space by comparing a variety of policy classes across three different tasks (see Fig 1). The first task illustrates prehensile manipulation by requiring the robot to grasp and lift a randomly placed sugar box from the table. The second task illustrates reaching, collision avoidance, and non-prehensile manipulation by requiring the robot to touch a target point on the table while avoiding collision with a randomly placed coffee can and then push the can over to the target. The third task illustrates constrained prehensile manipulation by requiring the robot to precisely maneuver its fingers through a tool tray handle and extract it from a tool box while avoiding undesirable collisions before placing it onto the table.

Baselines: We compared the performance of NGF against the following baselines: a fully-connected Neural Network policy (NN), a Riemannian Motion Policy (RMP), a recurrent neural network policy with Long Short-Term Memory units (LSTM), and an End-Effector Neural Network policy coupled with a manually-designed Geometric Fabric for operational space control (EEF-GF). We chose these specific baselines as they represent current practices in imitation learning for manipulation with respect to the choices of policy structure and action space. We provide additional details and justifications in Appendix E, and details about the experiment setup and data collection in Appendix F.

6.1 Evaluation Procedure

We trained all policies using the procedure described in Section 5 and deployed them on a physical robot platform to empirically measure their performance in representing the desired behavior implicit in the demonstrations. To be successful, the policies must simultaneously encode approach and grasping behavior by directly coordinating finger, palm, and arm motions for each task.

For each of the first two tasks involving prehensile and non-prehensile manipulation, we trained one instance of each policy class (NN, RMP, LSTM, EEF-GF, and NGF) on 80 demonstrations. We then evaluated each trained policy's ability to generalize to 20 arbitrary object poses on the surface of the table used for demonstrations. Each policy received the object pose estimated by CosyPose [52], a target pose equivalent to the mean pose in the training data, and an initial robot configuration as inputs. For the third task involving tool tray retrieval, we trained one instance of each policy class on 6 demonstrations, in which the toolbox is positioned at 6 different orientations $[10, 30, 50, 70, 90, 110]$ (deg.). We evaluated each trained policy's ability to retrieve the tool tray with the toolbox positioned at 9 different orientations $[-10, 0, 20, 40, 60, 80, 100, 120, 130]$ (deg.). Each policy received the toolbox orientation and the initial robot configuration as inputs.

Metrics: We quantified policy performance by evaluating both task *success rate* and the *safe deployment rate*. To compute safe deployment rate and prevent hardware damage, we used each trained policy to generate open-loop trajectories for a horizon of 13 seconds for Task 1, and 30 seconds for Tasks 2 and 3. We then visualized the robot enacting these trajectories using PyBullet [53] to determine its safety for deployment. We consider a trajectory to be safe if it does not excessively throw the robot into, through, or below the table, and does not lead to violent arm movements. We deployed all safe trajectories on the physical robot using a PD controller. To compute task success rate, we considered an execution to be successful only if the following criteria are met within the deployment horizon. *Task 1*: the robot grasps and lifts the sugar box from the table; *Task 2*: the robot first touches the target point without colliding into the coffee can and then moves the can over to the target location; *Task 3*: the robot grasps and lifts the tool tray out of the toolbox. In addition to these two metrics, we also measured the *imitation error* of each policy (defined in Eq. (12) of Appendix G) to evaluate sample efficiency.

6.2 Real-world Task Performance

Task Success Rate: The following key observations can be made from the results reported in Fig. 3a: **(i)** Most strikingly, NGF consistently offered the highest level of performance against all baselines and outperformed the best baseline by 40% to 100%, indicating the benefits of our specific architecture and our use of multiple action spaces. **(ii)** EEF-GF had the second highest level of performance by matching other baselines on the coffee can and tool tray tasks and outperforming them on the sugar box task. EEF-GF's good performance is explained by the fact that policies trained with elevated action spaces (in this case, palm pose action) are known to be easier to learn. However, it is important to note that EEF-GF enjoys the benefits of a manually-tuned Geometric Fabric-based operational space controller, and yet preforms worse than NGF which does not require such a controller. This result demonstrates the benefits of learning high-dimensional structured policies directly from data. **(iii)** Both second-order baselines, RMP and NN, surprisingly offer relatively same level of task success rates despite the added taskmap structure of the RMP which aims to introduce beneficial inductive biases. Although less clear, RMP and NN seemed to offer marginal improvements over LSTM, suggesting that second-order policies' ability to encode complex motions is comparable to that of a well-established recurrent structure commonly found in the literature.

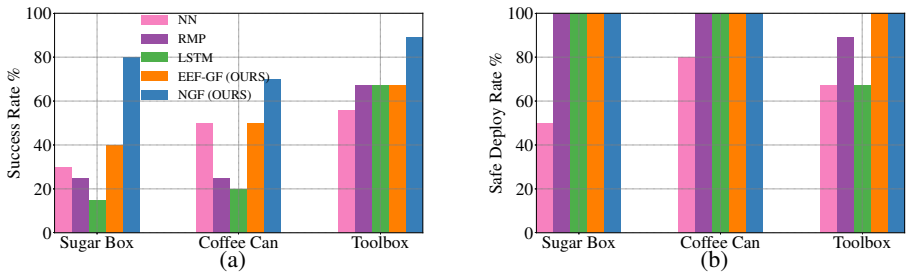

Figure 3: Policy performance on all three tasks based on (a) percentage of deployments that were successful, and (b) percentage of deployments that were safe to deploy on the real robot.

Safe Deploy Rate: We report the safe deployment rates for each policy and task in Fig. 3b. Critically, both NGF and EEF-GF were the safest to deploy across all tasks with 100% deploy rates, indicating that Geometric Fabrics-induced structure helps generate smooth and safe motions. Next, RMP had consistently higher deployment rates over NN, indicating that its taskmap structure helps improve safety. Finally, LSTM was comparable to RMP in generating safe trajectories.

6.3 Analysis on Sample Efficiency

We also studied the sample efficiency of each policy when learning to perform the first task.

Imitation Error: Imitation error captures the difference in trajectories generated by the learned policy and trajectories from the unseen data. We trained each policy on sets of demonstrations with sizes $[10, 30, 60]$, and for every 10 epochs, computed the imitation error on a set of 20 held-out demonstrations. After training for 2000 epochs, we selected the best model based on the imitation error for each policy over 5 random initialization seeds. We report the mean and standard deviation

of the imitation error for each policy across 5 random seeds in see Fig. 4a. We note that NGF is the most sample-efficient as it results in the lowest mean imitation error (and smallest variance) across all quantities of demonstrations.

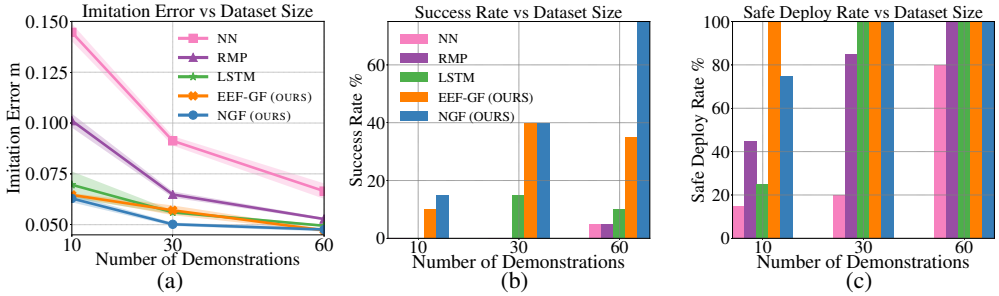

Figure 4: Sample efficiency analysis based on (a) imitation error of each policy (solid lines indicate the mean and the shaded area show mean $\pm$ standard deviation, over the 5 random seeds). (b) task success rate, and (c) safe deployment rate on physical robot.

Task Performance and Safety: Following the procedure in Section 6.1, we report how each policy's task success rate and safe deployment rate change with dataset size in Figs. 4b and 4c, respectfully. We note that real-world performance somewhat correlates with imitation error, but not strictly as observed in [2]. Similar to the results in Section 6.2, NGF consistently outperforms all the baselines across all dataset sizes (except for EEF-GF's better safe deployment rate with only 10 demonstrations, perhaps due to its manually-designed controller). Overall, the superior performance of NGF in all three metrics indicate that the additional structure imposed by Geometric Fabrics helps improve sample efficiency in policy learning. We provide an extended discussion in Appendix G.1.

7 Final Remarks

Discussion The mixture of near-perfect deploy rates and significantly higher task success rates of NGF against all baselines demonstrates the structural power of Geometric Fabrics. We find that EEF-GF is also a very viable architecture which demonstrates how the elevated pose action space when combined with an analytically-derived Geometric Fabric can improve policy learning. Our findings suggest that learning entirely new control structures with multiple action spaces beyond end-effector poses is possible with Geometric Fabrics. Although the policies were deployed open-loop, this is not an intrinsic limitation of NGFs. To illustrate its reactivity, we deployed the NGF policy on the coffee can task in closed-loop. We note that NGF successfully adapted and performed the task, even when we pushed the coffee can during execution (see Appendix H for more details).

Limitations Due to the high cost of running real-world experiments, we partially tested at most three random seeds per policy class per task and selected the best seed for full assessment. We acknowledge that experimental and policy variance could produce artifacts in the results. However, the combination of: 1) screening the best policy for each class and task, 2) the low imitation error and associated variance of NGF (see Fig. 4a), and 3) the consistency of NGF excelling across tasks makes it very unlikely that any of the baselines offer better performance in practice. We suspect that the performance of the EEF-GF policy could be improved by including the underlying Geometric Fabric policy inside the training loop. Finally, we deployed the policies open-loop for two primary reasons: 1) object pose detection becomes unreliable during heavy occlusion, and 2) we could assess the safety of the policy's trajectory before deployment, protecting our hardware. We note that training with much larger and diverse datasets will likely enable reliable closed-loop control.

Future work Our work opens up a number of avenues for future work, including 1) learning NGF policies from significantly larger and more diverse datasets, 2) learning NGFs that can process raw sensory streams including camera data, and 3) investigating the benefits of NGF for reinforcement learning. We believe that NGF-based learning frameworks have the potential to solve a wide variety of manipulation tasks in closed-loop, enabling us to move away from simplified representations of state (e.g., pose) and towards processing raw proprioceptive, tactile, and visual feedback in a structured manner.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:297–330, 2020.

[2] 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.

[3] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[4] R. Jeong, J. T. Springenberg, J. Kay, D. Zheng, Y. Zhou, A. Galashov, N. Heess, and F. Nori. Learning dexterous manipulation from suboptimal experts. *arXiv preprint arXiv:2010.08587*, 2020.

[5] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020.

[6] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[7] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

[8] 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*, 39(1):3–20, 2020.

[9] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[10] 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. *arXiv preprint arXiv:2103.05922*, 2021.

[11] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, and N. Ratliff. Learning reactive motion policies in multiple task spaces from human demonstrations. In *Conference on Robot Learning*, pages 1457–1468. PMLR, 2020.

[12] 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*, 2022.

[13] 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)*, pages 10206–10212, 2021. doi:10.1109/ICRA48506.2021.9561543.

[14] N. Gruver, M. A. Finzi, S. D. Stanton, and A. G. Wilson. Deconstructing the inductive biases of hamiltonian neural networks. In *International Conference on Learning Representations*, 2022.

[15] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[16] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[17] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *arXiv preprint arXiv:2104.11203*, 2021.

[18] M. Bain and C. Sommut. A framework for behavioural cloning. *Machine Intelligence*, 15, 1999.

[19] A. O. Ly and M. A. Akhloufi. Learning to drive by imitation: an overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 2020.

[20] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[21] S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning, 2014.

[22] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3309–3318, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[23] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *J. Mach. Learn. Res.*, 22:30–1, 2021.

[24] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[25] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[26] J. Choi, F. Castaneda, C. J. Tomlin, and K. Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*, 2020.

[27] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, volume 27, pages 1071–1079. Citeseer, 2014.

[28] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search (2015). *arXiv preprint arXiv:1501.05611*, 2015.

[29] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[30] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[31] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005.

[32] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters. Active incremental learning of robot movement primitives. In *Conference on Robot Learning*, pages 37–46. PMLR, 2017.

[33] A. Paraschos, C. Daniel, J. Peters, G. Neumann, et al. Probabilistic movement primitives. *Advances in neural information processing systems*, 2013.

[34] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters. Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Autonomous Robots*, 41(3):593–612, 2017.

[35] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[36] S. M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6): 752–765, 2014.

[37] N. Figueroa and A. Billard. Locally active globally stable dynamical systems: Theory, learning, and experiments. *The International Journal of Robotics Research*, page 02783649211030952, 2022.

[38] H. Ravichandar and A. Dani. Learning position and orientation dynamics from demonstrations via contraction analysis. *Autonomous Robots*, 43(4):897–912, 2019.

[39] H. Ravichandar, I. Salehi, and A. Dani. Learning partially contracting dynamical systems from demonstrations. In *Conference on Robot Learning*, pages 369–378. PMLR, 2017.

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

[41] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Neural autonomous navigation with riemannian motion policy. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8860–8866. IEEE, 2019.

[42] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[43] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[44] D. H. Ballard. Task frames in robot manipulation. In *AAAI*, volume 19, page 109, 1984.

[45] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff. RMPflow: A computational graph for automatic motion policy generation. In *The 13th International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[46] 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.

[47] M. Ciocarlie, C. Goldfeder, and P. Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*. Citeseer, 2007.

[48] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[49] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[50] J. Langford, R. Salakhutdinov, and T. Zhang. Learning nonlinear dynamic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 593–600, 2009.

[51] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[52] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[53] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[54] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017. doi:10.1109/TNNLS.2016.2582924.

[55] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2020. doi:10.1109/LRA.2019.2956365.

[56] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635, 2018. doi:10.1109/ICRA.2018.8461249.