Diffeomorphic Transforms for Generalised Imitation Learning

Weiming Zhi WEIMING.ZHI@SYDNEY.EDU.AU
School of Computer Science, The University of Sydney, Australia

Tin Lai TIN.LAI@SYDNEY.EDU.AU
School of Computer Science, The University of Sydney, Australia

Lionel Ott LIONEL.OTT@MAVT.ETHZ.CH
Autonomous Systems Lab, ETH Zurich, Switzerland

Fabio Ramos FABIO.RAMOS@SYDNEY.EDU.AU
School of Computer Science, The University of Sydney, Australia and NVIDIA, USA

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

We address the generalised imitation learning problem of producing robot motions to imitate expert demonstrations, while adapting to novel environments. Past studies have often focused on methods that closely mimic demonstrations. However, to operate reliably in novel environments, robots should be able to adapt their learned motions accordingly. Motivated by this, we devise a framework capable of learning a time-invariant dynamical system to imitate demonstrations, and generalise to account for changes to the surroundings. To ensure the system is robust to perturbations, we need to maintain its stability. Our framework enforces stability in a principled manner: we start with a known stable system and use differentiable bijections (diffeomorphisms) to morph the system into the desired target system. We modularise robot motion and develop *diffeomorphic transforms* to encode individual actions. A composition of transforms produces generalised behaviour that complies with multiple requirements, such as mimicking demonstrations while avoiding obstacles. We evaluate our framework in both simulation and on a real-world 6-DOF manipulator. Results show our framework can produce a stable system that is collision-free and incorporates user-specified biases, while closely resembling demonstrations.

1. Introduction

Imitation learning is an approach where robots are taught to execute novel motions from a few human expert demonstrations (Billard and Grollman, 2012). This approach is natural for generating complex motions, as it circumvents hand coding movements or specifically designing control costs (Osa et al., 2018). Crucial to imitation learning is a generative model mimicking the demonstrations, which can typically be described as having a time-dependent or state-dependent representation. In this paper, we focus on the state-dependent category, where motions are generated by integrating a time-invariant dynamical system. A major challenge of representing robot motion with a state-dependent system is the need to ensure the system's asymptotic stability. Methods (Khansari-Zadeh and Billard, 2011) have been developed to learn stable dynamics. In particular, recent methods such as (Rana et al., 2020b; Urain et al., 2020) attempt to construct inherently stable dynamical systems by learning a differentiable bijective mapping (i.e. a diffeomorphism) linking a stable

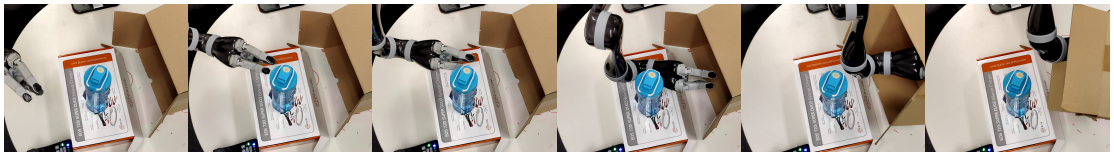


Figure 1: A motivating application for our work is generalising stable systems learned from expert demonstrations, when surroundings change. Here, a JACO arm is shown 8 recorded human demonstrations. Then, an obstacle bottle is placed right in the path of the recorded demonstrations. Our proposed method allows the arm to warp smoothly around obstacle and successfully reach and lift the box.

system to a target system, using invertible neural networks (INN). However, it is unclear how to inject additional knowledge into these INN models to produced generalised behaviour without providing demonstrations of these generalisations. This makes it difficult to use INNs to encode additional *generalised behaviour*.

We contribute the diffeomorphic transforms framework for generalised imitation learning, which account for environment changes, by factoring in other sources of information, such as occupancy data or additional user specification. A specific example is described in fig. 1, where additional obstacles cause the previous demonstrations to be in collision. We represent each behaviour such as reproducing demonstrations and obstacle avoidance, as individual diffeomorphic transforms, which take the form of integral curves of vector fields. These can be built in various ways, such as learning from examples obtained from continuous occupancy models or be hand-crafted. By combining transforms, generated motions can be learned from demonstrations and adjusted to avoid new obstacles or warp towards a designated point, all while conserving the stability of the resulting system.

2. Related Work

Imitation learning aims to replicate motion based on demonstrations, allowing human expert to transfer skills to robot systems. One class of imitation learning methods represents robot motion by a stable state-dependent dynamical system (Khansari-Zadeh and Billard, 2011; Khansari-Zadeh and Billard, 2014). Such a representation is more robust to perturbations, particular in regions with no data, as the system is guaranteed to stabilise at some point. A pioneer example of a stable state-based representation is SEDS (Khansari-Zadeh and Billard, 2011), which to constrains the learned system to quadratic Lyapunov functions via sequential quadratic programming. Further work in (Khansari-Zadeh and Billard, 2014) expands the family of Lyapunov functions for candidates. Recent approaches often take a different strategy. Instead of explicitly parameterising and learning the class of stable systems, these approaches learn diffeomorphisms which map between a stable base system and some target, "morphing" of the base system, and preserves the asymptotic stability of the original system. A similar morphing approach has also been used to speed up the integration of differential equations (Zhi et al., 2021). Our method, along with those in (Urain et al., 2020; Rana et al., 2020b) take this approach. Despite the advancements made in imitation learning, there have been few work addressing adapting imitations after learning. The authors of (Calinon, 2016) propose to adjust parameters of Gaussian mixture models based on tasks, while the authors of (Rana et al., 2017) make attempts to alter generated trajectories, by post-processing with motion planning. Early dynamical system-based collision avoidance approaches have been proposed in (Hoffmann et al., 2009; Khansari-Zadeh and Billard, 2012), although no consideration is given to C-space avoidance, with obstacles defined in task space. A more recent method in (Rana et al., 2019), introduces learnable weights for hand-coded policies within the Riemannian Motion Policy (RMP) framework (Ratliff et al., 2018) which combines policies from different task-spaces in C-

space. Subsequent work in (Rana et al., 2020b,a) provide more flexibility by learning to imitate, but still rely on hand-coded "distance-to-point" repulsors from the RMP framework for avoidance. Diffeomorphic transforms, proposed in this work, not only flexibly model imitated motions, but also encode collision-avoidance directly from continuous occupancy representations, without hand-coded repulsor systems.

3. Preliminaries

We first introduce necessary background concepts, then present our framework in section 4.

Motion Representation and Stability of Systems: Following state-dependent imitation learning literature (Khansari-Zadeh and Billard, 2011; Khansari-Zadeh and Billard, 2014; Rana et al., 2020b), we represent motion as a mapping between the robot state $\boldsymbol{x} \in \mathbb{R}^d$ and its velocity $\dot{\boldsymbol{x}} \in \mathbb{R}^d$. We model the motion as a time-invariant first-order dynamical system: $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)), \boldsymbol{x}(0) = \boldsymbol{x}_0$, where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a non-linear mapping, and $\boldsymbol{x}_0$ is the initial state. A trajectory of the system is given by $\xi(t, \boldsymbol{x}_0) = \boldsymbol{x}_0 + \int_0^t \dot{\boldsymbol{x}}(s)\mathrm{d}s$. An equilibrium point, $\boldsymbol{x}^*$, is a point where velocity is zero, $\boldsymbol{x}^* \in \mathbb{R}^d : f(\boldsymbol{x}^*) = 0$. A system is *locally asymptotically stable* in a region $S \subset \mathbb{R}^d$ if trajectories starting in S converge to some equilibrium $\boldsymbol{x}^* \in S$, i.e. $\lim_{t\rightarrow\infty} \xi(t, \boldsymbol{x}_0) = \boldsymbol{x}^*, \forall \boldsymbol{x}_0 \in S$. A system is *globally asymptotically stable* if $S = \mathbb{R}^d$, and trajectories converge to a unique equilibrium point.

Imitating demonstrations: Provided a set of N demonstrations, Ξ, where the i^{th} demonstration is given by a sequence of states, of length l_i, i.e. $\Xi = \{\{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \ldots, \boldsymbol{x}_{i,l_i}\}\}_{i=1}^N$. The velocities, $\dot{\boldsymbol{x}}_{i,j}$, are assumed to be available. To imitate demonstrations, we aim to learn an $f(\boldsymbol{x})$ such that trajectories generated match the empirical demonstrations Ξ. Assuming $\boldsymbol{\theta}$ parameterises f, we can optimise $\boldsymbol{\theta}$ via least squares: $\min_{\boldsymbol{\theta}} \sum_{i=1}^N \sum_{j=1}^{l_i} ||\dot{\boldsymbol{x}}_{i,j} - f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i,j})||_2^2$, such that $\dot{\boldsymbol{x}} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is stable.

Preserving Stability with Diffeomorphisms: An elegant method, as presented in (Rana et al., 2020b), of ensuring stability of $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, is by viewing the target system as a *natural gradient system* defined on some d-dimensional manifold $\boldsymbol{M}$, where integral curves of the target system on $\boldsymbol{M}$ correspond to integral curves of another known simple system in Euclidean space. A bijective mapping is a *diffeomorphism* if ψ and ψ^{-1} are continuously differentiable. A diffeomorphism $\psi : \boldsymbol{M} \rightarrow \mathbb{R}^d$ maps points on the manifold to Euclidean space. Given a differentiable potential function, $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}$, and diffeomorphism ψ, and following (Rana et al., 2020b), we can define the *natural gradient descent* system (Amari, 1998) on $\boldsymbol{M}$, corresponding to the system $\dot{\boldsymbol{z}} = \nabla_{\boldsymbol{z}}(\boldsymbol{z})$ defined in Euclidean space as:

$$\dot{\boldsymbol{x}} := f(\boldsymbol{x}) = -\boldsymbol{G}(\boldsymbol{x})^{-1} \nabla_{\boldsymbol{x}} \Phi(\psi(\boldsymbol{x})), \tag{1}$$

where $\boldsymbol{G} : \boldsymbol{M} \rightarrow \mathbb{R}^{d\times d}_{++}$ is a real positive definite matrix *Riemannian metric* which varies smoothly on $\boldsymbol{M}$. We take the metric as $\boldsymbol{G}(\boldsymbol{x}) = \boldsymbol{J}_\psi(\boldsymbol{x})^\top \boldsymbol{J}_\psi(\boldsymbol{x})$, where $\boldsymbol{J}_\psi(\boldsymbol{x})$ is the Jacobian of ψ. It has been shown (Rana et al., 2020b), if the system defined by $\dot{\boldsymbol{z}} = -\nabla_{\boldsymbol{z}} \Phi(\boldsymbol{z}) \in \mathbb{R}^d$ is globally asymptotically stable, then the system defined by the natural gradient descent in eq. (1) is globally asymptotically stable. Building inherently stable systems can be done in a principled fashion by specifying a stable potential function $\boldsymbol{\Phi}$ in Euclidean space, and constructing diffeomorphisms, ψ, such that the natural gradient system of $\boldsymbol{\Phi}$, as defined in eq. (1), matches a desired *target system*.

4. Diffeomorphic Transforms

Beyond simply mimicking the provided demonstrations, we are also interested in modifying f when the situation changes and no further demonstrations are provided. We introduce diffeomorphic

transforms (DTs) as a tool to generalise imitation learning. DTs are diffeomorphisms which encode desired behaviour. By composing DTs, we can build a stable system which is capable of incorporating various behaviours such as imitating demonstrations and generalising to novel environments. This section is structured as follows: We start by discussing how multiple DTs can sequentially be composed together (section 4.1). Next, we outline the general theory on which DTs are built and how to imbue DTs with desirable properties (section 4.2). Thereafter, we derive DTs with the following behaviours: (i) imitating demonstration data (section 4.3), (ii) avoiding collision (section 4.4), and (iii) biasing towards specified coordinates (section 4.6).

4.1. Composition of Diffeomorphic Transforms

Diffeomorphisms provide smooth bijective mappings between manifolds. If we have two manifolds which are linked by a diffeomorphism and a system on one manifold, we can find a natural gradient system on the other manifold via eq. (1). In this paper, diffeomorphisms map between manifolds of the same dimension, and can be thought of as a change of coordinate systems. We denote diffeomorphisms by ψ and manifolds by $\boldsymbol{M}$ with superscripts for specific diffeomorphisms or manifolds. Previous work (Ratliff, 2013), define the forward mapping ψ to be from some warped manifold to Euclidean space. We adopt this convention and define the diffeomorphisms ψ of a DT as a mapping from some manifold which encodes desirable behaviour to one that does not. Correspondingly, the inverse ψ^{-1} maps to a manifold with encoded behaviour.

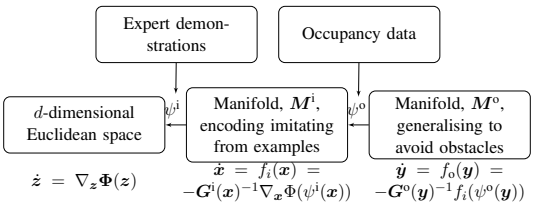


Figure 2: To obtain a stable system that imitates demonstrations and avoids collisions, we compose two DTs: ψ^{i} mapping from $\boldsymbol{M}^{\mathrm{i}}$ to Euclidean space, and ψ^{o} from $\boldsymbol{M}^{\mathrm{o}}$ to $\boldsymbol{M}^{\mathrm{i}}$. The system on $\boldsymbol{M}^{\mathrm{i}}$ reproduces the demonstrations, while that on $\boldsymbol{M}^{\mathrm{o}}$ additionally avoids obstacles.

Compositions of DTs can be done in a sequentially. A specific example of such a composition is given in Figure 2, where we obtain a system which imitates demonstrations while also avoiding collisions. We start with a stable attractor in Euclidean space, with potential $\Phi(\boldsymbol{z}) = ||\boldsymbol{z} - \boldsymbol{z}_e||_2$. The DT, ψ^{i}, is learned from demonstration data, and maps from the d-dimensional manifold, $\boldsymbol{M}^{\mathrm{i}}$ to Euclidean space. Let $\boldsymbol{G}^{\mathrm{i}} = \boldsymbol{J}_{\psi^{\mathrm{i}}}(\boldsymbol{x})^{\top}\boldsymbol{J}_{\psi^{\mathrm{i}}}(\boldsymbol{x})$, the natural gradient system $\dot{\boldsymbol{x}} = f_i(\boldsymbol{x}) = -\boldsymbol{G}^{\mathrm{i}}(\boldsymbol{x})^{-1}\nabla_{\boldsymbol{x}}\Phi(\psi^{\mathrm{i}}(\boldsymbol{x}))$, $\boldsymbol{x} \in \boldsymbol{M}^{\mathrm{i}}$ generates trajectories that mimic expert demonstrations. The DT ψ^{o} is constructed from occupancy data, and maps from another d-dimensional manifold $\boldsymbol{M}^{\mathrm{o}}$ to $\boldsymbol{M}^{\mathrm{i}}$. The resulting natural gradient system $\dot{\boldsymbol{y}} = f_{\mathrm{o}}(\boldsymbol{y}) = -\boldsymbol{G}^{\mathrm{o}}(\boldsymbol{y})^{-1}f_i(\psi^{\mathrm{o}}(\boldsymbol{y}))$ thus encodes both the imitation and collision-avoidance properties. Throughout the paper, we denote system states in Euclidean space by $\boldsymbol{z}$, those on a manifold encoding demonstrated behaviour by $\boldsymbol{x}$, and those on a manifold which encodes generalisation behaviour, such as collision avoidance, by $\boldsymbol{y}$. Thoughout this paper, the system in Euclidean space is always given as an attractor with potential $\Phi(\boldsymbol{z}) = ||\boldsymbol{z} - \boldsymbol{z}_e||_2$, with equilibrium $\boldsymbol{z}_e \in \mathbb{R}^d$. Subscripts or superscripts of i, o, b, in our notation relate to the *imitation*, *obstacle-avoiding*, *bias-incorporation* capabilities we wish to endow DTs.

4.2. Infinitesimal Generators of Flows

In this section we introduce the building block of diffeomorphic transforms (DTs): Flows (integral curves) generated by infinitesimal generators (vector fields).

Definition 1 (Flows) *A (global) flow on $\mathbb{R}^d$ is a continuous mapping $\gamma : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, such that $\forall \boldsymbol{s} \in \mathbb{R}^d$ and $\forall t_1, t_2 \in \mathbb{R}$, there is $\gamma(0, \boldsymbol{s}) = \boldsymbol{s}$ and $\gamma(t_2, \gamma(t_1, \boldsymbol{s})) = \gamma(t_1 + t_2, \boldsymbol{s})$.*

Definition 2 (Infinitesimal Generator) *The infinitesimal generator, $V : \mathbb{R}^d \to \mathbb{R}^d$, of a smooth flow γ is a smooth vector field mapping from each $\boldsymbol{s} \in \mathbb{R}^d$ to the tangent $\gamma'(0, \boldsymbol{s})$.*

We shall use the following property of flows to build DTs: For every infinitesimal generator V that is Lipschitz continuous, the flow for time $t \in \mathbb{R}$, $\gamma(t, \cdot)$, is a diffeomorphism. Specifically, for any initial point $\boldsymbol{s}_0 \in \mathbb{R}^d$ and time t, if $\boldsymbol{s}_t = \gamma(t, \boldsymbol{s}_0) \in \mathbb{R}^d$, then $\boldsymbol{s}_0 = \gamma(-t, \boldsymbol{s}_t)$. That is, the inverse of $\gamma(t, \cdot)$ is unique and given by $\gamma(-t, \cdot)$. This is a direct consequence of the existence and of the existence and uniqueness of ODE initial value problems (Picard–Lindelöf theorem (Coddington and Levinson, 1955)), and also given as Theorem 9.12 of (Lee, 2012). Intuitively, following an infinitesimal generator V for time t from $\boldsymbol{s}_0$ leads to a new location $\boldsymbol{s}_t$. To return to $\boldsymbol{s}_0$ from $\boldsymbol{s}_t$ requires the reversed vector field $-V$, which is equivalent to following the vector field V for the negative amount of time, i.e. $-t$. This property provides us with a principled mechanism to construct a diffeomorphism: The smooth infinitesimal generator V and its associated flow γ form our diffeomorphism and its inverse. Invertible functions constructed by integrating vector fields, parameterised by neural networks, have also been explored in the density estimation literature as *Continuous Normalising Flows* (Chen et al., 2018; Grathwohl et al., 2019), with the additional requirement of volume preservation. The following sections describe how to construct infinitesimal generators with different properties.

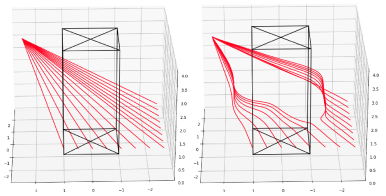
4.3. Learning Infinitesimal Generators from Demonstrations

We outline a method to learn smooth infinitesimal generators which encode demonstrated motions. We exploit smooth Gaussian radial basis functions to parameterise the infinitesimal generator $V^{\mathrm{i}} : \mathbb{R}^d \to \mathbb{R}^d$ with a weighted combination of basis functions, $V^{\mathrm{i}}(\boldsymbol{s}) = \boldsymbol{W}^\top K(\boldsymbol{s}, \hat{\boldsymbol{s}})$, where $K(\boldsymbol{s}, \hat{\boldsymbol{s}}) = [k(\boldsymbol{s}, \hat{\boldsymbol{s}}_1), k(\boldsymbol{s}, \hat{\boldsymbol{s}}_2), \dots k(\boldsymbol{s}, \hat{\boldsymbol{s}}_m)]^\top \in \mathbb{R}^{d \times m}$ is the projection of point $\boldsymbol{s} \in \mathbb{R}^d$ to m pre-determined *inducing states*, $\hat{\boldsymbol{s}}_i \in \mathbb{R}^d$, for $i = 1, 2, \dots, m$. The radial basis functions $k(\boldsymbol{s}, \hat{\boldsymbol{s}}_i) = \exp(-\ell ||\boldsymbol{s} - \hat{\boldsymbol{s}}_i||_2)$ are centered on each inducing state. The vector field V^{i} is parameterised by weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times m}$. Like (Rana et al., 2020b; Urain et al., 2020), the policies are defined for world space position, and are typically coordinates in $\mathbb{R}^3$. The hyper-parameter ℓ of the radial basis function controls its "width". The flow on the infinitesimal generator is obtained by taking integral curves. We assume there exists a d-dimensional manifold $\boldsymbol{M}^{\mathrm{i}}$, where trajectories follow demonstrations. Points on $\boldsymbol{x} \in \boldsymbol{M}^{\mathrm{i}}$ are mapped to the corresponding point in Euclidean space, $\boldsymbol{z}$, by following the flow γ^{i} on V^{i} for time t. We define the diffeomorphism $\psi^{\mathrm{i}}_{\boldsymbol{W}} : \boldsymbol{M}^{\mathrm{i}} \to \mathbb{R}^d$ and its inverse as:

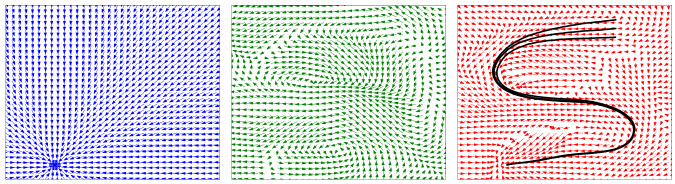
$$\psi^{\mathrm{i}}_{\boldsymbol{W}}(\boldsymbol{x}) := \boldsymbol{x} + \int_0^t V^{\mathrm{i}}(\gamma^{\mathrm{i}}(u, \boldsymbol{x}))\mathrm{d}u = \boldsymbol{z}, \qquad \psi^{\mathrm{i}}_{\boldsymbol{W}}(\boldsymbol{z})^{-1} := \boldsymbol{z} + \int_{-t}^0 V^{\mathrm{i}}(\gamma^{\mathrm{i}}(u, \boldsymbol{z}))\mathrm{d}u = \boldsymbol{x}. \tag{2}$$

As analytical integrals are often difficult to obtain we use numerical integration techniques, Euler's method in our case due to its simplicity. In the context of the imitating demonstrations, we are assumed to have N demonstrations, each a sequence of states. That is, $\hat{\xi}_i = \{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \dots, \boldsymbol{x}_{i,l_i}\}$, where l_i is the length of the i^{th} demonstration. We can formulate the least squares optimisation problem as:

$$\min_{\boldsymbol{W}} \Big\{ \sum_{i=1}^{N} \sum_{j=1}^{l_i} ||\dot{\boldsymbol{x}}_{i,j} + \boldsymbol{G}^{\mathrm{i}}(\boldsymbol{x}_{i,j})^{-1} \nabla_{\boldsymbol{x}_{i,j}} \Phi(\psi_{\boldsymbol{W}}(\boldsymbol{x}_{i,j}))||_2^2 \Big\}, \quad \boldsymbol{G}^{\mathrm{i}}(\boldsymbol{x}_{i,j}) = \boldsymbol{J}_{\psi_{\boldsymbol{W}}}(\boldsymbol{x}_{i,j})^\top \boldsymbol{J}_{\psi_{\boldsymbol{W}}}(\boldsymbol{x}_{i,j}). \tag{3}$$



(*a*) A 3d example of avoiding a cube obstacle (black). (Left) an attractor $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is on $\boldsymbol{M}$; (Right) The corresponding natural gradient system.



(*b*) Left: a simple attractor base system; Center: the infinitesimal generator $V^{\mathrm{i}}_{\boldsymbol{W}}$ learned on demonstrations of "S"; Right: the learned system with roll-outs (black).

Figure 3: Examples of diffeomorphic transforms for collision avoidance and imitation.

We optimise to obtain $\boldsymbol{W}$. By eq. (1), the natural gradient system which generates trajectories, with $\boldsymbol{x} \in \boldsymbol{M}^{\mathrm{i}}$, is $\dot{\boldsymbol{x}} = f_i(\boldsymbol{x}) = -\boldsymbol{G}^{\mathrm{i}}(\boldsymbol{x})^{-1}\nabla_{\boldsymbol{x}}\Phi(\psi^{\mathrm{i}}_{\boldsymbol{W}}(\boldsymbol{x}))$.

4.4. Infinitesimal Generators for Collision Avoidance

Beyond reproducing motions from demonstrations, we wish to adapt the generated motions so that they remain collision-free when the environment changes. In this section, we introduce DTs which encode collision-avoiding behaviour, by using smooth obstacle gradients. These are obtained analytically via a continuous maps, such as the representations described in (Ramos and Ott, 2016; Zhi et al., 2019). Using occupancy information about the new environment in the form of a training dataset of n pairs, $\{(\boldsymbol{s}_k, o_k)\}_{k=1}^n$, where each sample $\boldsymbol{s}_k \in \mathbb{R}^d$ is a point in the state-space, and $o_k \in \{0, 1\}$ is a binary variable indicating whether $\boldsymbol{s}_k$ is occupied or free. A continuous map uses kernelised logistic regressor (Bishop, 2007) wrapped around Gaussian radial basis functions to learn a mapping between coordinates to the probability of them being occupied (occupancy). The representation is given as $f^{\mathrm{map}}(\boldsymbol{s}) := p(o = 1|\boldsymbol{s})$, and is smooth as it consists of sigmoid and Gaussian functions. We exploit the smooth obstacle gradients to construct an infinitesimal generator: $V^o(\boldsymbol{s}) = \nabla_{\boldsymbol{s}} p(o = 1|\boldsymbol{s}) = \nabla_{\boldsymbol{s}} f^{\mathrm{map}}$. Following the flow on V^o in reverse time reduces the observed occupancy, i.e. lower collision likelihood. This captures the goal of the inverse diffeomorphism ψ^{o-1}, which is designed to map to a manifold encoding collision-avoiding behaviour. Accordingly, we define a d-dimensional manifold $\boldsymbol{M}^{\mathrm{o}}$ where trajectories exhibit obstacle avoidance behaviour. Provided some manifold $\boldsymbol{M}$, the infinitesimal generator V^o generates flow γ^o. Thus, for some $t \in \mathbb{R}$, let $\boldsymbol{y} \in \boldsymbol{M}^{\mathrm{o}}$ and $\boldsymbol{x} \in \boldsymbol{M}$, we define the diffeomorphism $\psi^o : \boldsymbol{M}^{\mathrm{o}} \to \boldsymbol{M}$ as: $\psi^o(\boldsymbol{y}) := \gamma^o(t, \boldsymbol{y}) = \boldsymbol{y} + \int_0^t V^o(\gamma^o(u, \boldsymbol{y}))\mathrm{d}u = \boldsymbol{x}$, and its inverse $\psi^o(\boldsymbol{x})^{-1} := \gamma^o(-t, \boldsymbol{x}) = \boldsymbol{x} + \int_{-t}^0 V^o(\gamma^o(u, \boldsymbol{x}))\mathrm{d}u = \boldsymbol{y}$, where a numerical integrator is used to evaluate the integral. The natural gradient system on $\boldsymbol{M}^{\mathrm{o}}$ is: $\dot{\boldsymbol{y}} = -\boldsymbol{G}^{\mathrm{o}}(\boldsymbol{y})^{-1} f(\psi^o(\boldsymbol{y}))$, where f corresponds to a system on $\boldsymbol{M}$. This could be, for example, a simple attractor or f_{i} that encodes demonstrations. When rolling out trajectories with the collision-avoiding system, the initial conditions given by $\boldsymbol{y_0} \in \boldsymbol{M}^{\mathrm{o}}$ are assumed to be collision-free. As $\boldsymbol{y}$ approaches the boundary of the obstacle, the natural gradient system smoothly warps around. Figure 3(*a*) gives an example of trajectories on $\boldsymbol{M}^{\mathrm{o}}$ smoothly avoiding an obstacle.

4.5. Infinitesimal Generators in C-space for Collision Avoidance

In imitation learning setups with a manipulator, it is typical to learn a stable dynamical system that governs Cartesian *task-space* coordinates of the end-effector. However, collision avoidance is not limited to the end-effector. We can define DTs which encode avoidance behaviours for body points

on the manipulator, and operate in the configuration space (C-space) (Lavalle, 2006) of the robot, rather than task-space. A $d_{\boldsymbol{q}}$-dimensional configuration of the manipulator is $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^{d_{\boldsymbol{q}}}$, where the C-space is denoted as $\mathcal{Q}$. Robot configurations can be mapped into the task-space coordinates of c body points on the robot by a set of c kinematic functions, $\{\mathbf{x}_i^{\mathrm{f}}\}_{i=1}^{c}$. Each kinematic function, $\mathbf{x}_i^{\mathrm{f}} : \mathcal{Q} \to \mathbb{R}^d$, represents the Cartesian task space coordinates of the i^{th} body point from a given configuration. For multiple-link manipulators, these kinematic functions are modelled by of simple operations on basic trigonometry functions (Denavit et al., 1955), and are Lipschitz continuous with respect to the configurations. We set the manipulator end-effector as the first body point, and $\mathbf{x}_1^{\mathrm{f}}(\cdot)$ is the forward kinematics of the robot. Denoting the end-effector position as $\mathbf{x}^e$, the velocity of the end-effector and the configurations are: $\dot{\mathbf{x}}^e = \boldsymbol{J}_{\mathbf{x}_1^{\mathrm{f}}}(\boldsymbol{q})\dot{\boldsymbol{q}}$, $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}_{\mathbf{x}_1^{\mathrm{f}}}(\mathbf{x}^e)\dot{\mathbf{x}}^e$, $\boldsymbol{J}_{\mathbf{x}_1^{\mathrm{f}}}(\cdot) = \frac{\partial \mathbf{x}_1^{\mathrm{f}}}{\partial \boldsymbol{q}}(\cdot)$. The dimensions of $\boldsymbol{q}$ are typically greater than that of the Cartesian coordinates $\mathbf{x}^e$. To obtain a unique $\dot{\boldsymbol{q}}$ for some $\dot{\boldsymbol{x}}$, we take $\boldsymbol{J}^{\dagger}_{\mathbf{x}_1^{\mathrm{f}}} = \boldsymbol{J}^{\top}_{\mathbf{x}_1^{\mathrm{f}}}(\boldsymbol{J}_{\mathbf{x}_1^{\mathrm{f}}}\boldsymbol{J}^{\top}_{\mathbf{x}_1^{\mathrm{f}}})^{-1}$ as the Moore-Penrose pseudoinverse. The Moore-Penrose pseudoinverse preserves stability of the resulting system (Wang et al., 2018), and *pull* the task-space velocities to C-space: if the end-effector follows a system given by $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, the system in C-space is $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}_{\mathbf{x}_1^{\mathrm{f}}} f(\mathbf{x}_1^{\mathrm{f}}(\boldsymbol{q}))$. Using inverse kinematics, we can find a starting configuration $\boldsymbol{q}_0$, then rollout a sequence of configurations. The derivative of the combined occupancy of body points, wrt to configurations, gives smooth vector field $V^o_{\boldsymbol{q}}(\boldsymbol{q}) := \nabla_{\boldsymbol{q}} \sum_{j=1}^{c} f^{map}(\mathbf{x}_j^{\mathrm{f}}(\boldsymbol{q}))$, where f^{map} is the continuous occupancy representation as outlined in section 4.4. We can use $V^o_{\boldsymbol{q}}$ as the infinitesimal generator of a diffeomorphism, giving $d_{\boldsymbol{q}}$-dimensional manifold $\mathcal{Q}^o$, where natural gradient systems corresponding to systems in $\mathcal{Q}$ are collision-free. The diffeomorphism, $\phi^o_{\boldsymbol{q}}$, mapping from $\mathcal{Q}^o$ to $\mathcal{Q}$ is given by following the flow, $\gamma^o_{\boldsymbol{q}}$, on $V^o_{\boldsymbol{q}}$. For a configuration $\boldsymbol{q} \in \mathcal{Q}$, and its corresponding collision-avoiding configuration $\boldsymbol{p} \in \mathcal{Q}^o$, we define the diffeomorphism: $\psi^o_{\boldsymbol{q}}(\boldsymbol{p}) := \gamma^o(t, \boldsymbol{p}) = \boldsymbol{p} + \int_0^t V^o_{\boldsymbol{q}}(\gamma^o(u, \boldsymbol{p}))\mathrm{d}u = \boldsymbol{q}$ and its inverse: $\psi^o_{\boldsymbol{q}}(\boldsymbol{q})^{-1} := \gamma^o(-t, \boldsymbol{q}) = \boldsymbol{q} + \int_{-t}^0 V^o_{\boldsymbol{q}}(\gamma^o(u, \boldsymbol{q}))\mathrm{d}u = \boldsymbol{p}$. Following eq. (1), we can define our system on $\mathcal{Q}^o$ as:

$$\dot{\boldsymbol{p}} = -\boldsymbol{G}^{\mathrm{o}}_{\mathcal{Q}}(\boldsymbol{p})^{-1}[\boldsymbol{J}_{\mathbf{x}_1^{\mathrm{f}}}{}^{\dagger} f_i(\mathbf{x}_1^{\mathrm{f}}(\psi^o_{\boldsymbol{q}}(\boldsymbol{p})))], \qquad \boldsymbol{G}^{\mathrm{o}}_{\mathcal{Q}}(\boldsymbol{p}) = \boldsymbol{J}_{\psi^o_{\boldsymbol{q}}}(\boldsymbol{p})^{\top}\boldsymbol{J}_{\psi^o_{\boldsymbol{q}}}(\boldsymbol{p}). \tag{4}$$

By using the natural gradient system on the collision-avoiding manifold in C-space, we can roll out $\boldsymbol{p}$ to obtain collision-free configurations.

4.6. Infinitesimal Generators for User-Specified Bias

Sometimes it is desirable to deform learned motions based on user input without recording new expert demonstrations. In this section we outline a method to construct a diffeomorphism which can apply deformation biases to specific coordinates along a trajectory.

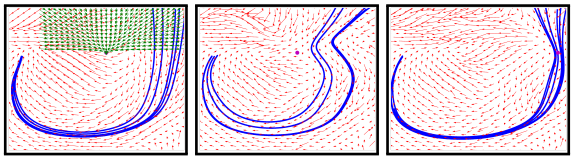


Figure 4: (Left) a system (red vector field) on $\boldsymbol{M}^{\mathrm{i}}$ learned to imitate a "J" (in blue). We bias the system to the magenta point, the green vector field shows, $-V^{\mathrm{b}}(\boldsymbol{s})$; (Center) the biased system; (Right) biasing to another coordinate.

Similar to previous sections, we focus on building DTs by directly crafting their infinitesimal generator. We design the manifold $\boldsymbol{M}^{\mathrm{b}}$ by constructing the diffeomorphism ψ^b such that the inverse $\psi^{b^{-1}}$ maps straight trajectories in a specific region to trajectories with the specified

Figure 5: Imitating expert examples from the LASA dataset (Khansari-Zadeh and Billard, 2011). Reproduced motion in red and shown demonstration in green.

bias. As we define $\psi^{b^{-1}}$ from reverse-time integral curves of the generator, we need to build the generator V^{b} such that the biased direction is given by $-V^{\mathrm{b}}$. This is achieved by regressing specified vectors onto the infinitesimal generator representation. We collect sample points within a fixed region in the state-space and compute a unit vector pointing away from the point of interest, giving us a dataset of $n_{\boldsymbol{s}}$ pairs $\mathcal{D} = \{(\boldsymbol{s}_i^{\mathrm{a}}, \nabla_{\boldsymbol{s}_i^{\mathrm{a}}} ||\boldsymbol{s}_i^{\mathrm{a}} - \boldsymbol{s}^{tar}||_2)\}_{i=1}^{n_{\boldsymbol{s}}}$, where $\boldsymbol{s}^{\mathrm{a}} \in \mathcal{S}$ are sample points in some region $\mathcal{S}$ where the bias takes effect, and $\boldsymbol{s}^{tar}$ is the coordinate of interest. To produce a valid DT, we require an infinitesimal generator to be Lipschitz continuous, by minimising: $\boldsymbol{W}^* := \arg\min_{\boldsymbol{W}} \sum_{i=1}^{n_{\boldsymbol{s}}} ||\nabla_{\boldsymbol{s}_i^{\mathrm{a}}} - \boldsymbol{W}^\top K(\boldsymbol{s}_i, \hat{\boldsymbol{s}})||_2^2$, where we now have a smooth $V^{\mathrm{b}} := \boldsymbol{W}^{*^\top} K(\boldsymbol{s}, \hat{\boldsymbol{s}})$. With eq. (1) we can take natural gradients on the resulting manifold $\boldsymbol{M}^{\mathrm{b}}$. An example of biasing a learned system towards points of interest is shown in fig. 4.

5. Experimental Evaluation

We empirically evaluate the performance of diffeomorphic transforms (DTs) to imitate and generalise robot motions, both in simulation and on a real-world 6-DOF JACO manipulator.

Imitating Expert Demonstrations: The first analysis is on the ability of DTs to imitate provided demonstrations. We train DTs from 8 alphabet characters from the LASA handwriting dataset (Khansari-Zadeh and Billard, 2011). There are 7 demonstrations for each character. We use $m = 720$ inducing states for the Gaussian basis function, each with $\ell = 0.005$, arranged in an equally-spaced grid covering the range of the demonstrations. Qualitative results are presented in fig. 5, where generated trajectories are given in red with demonstrations in green. The generated trajectories are smooth and consistent with the demonstrated motions. We note that first-order dynamical systems cannot model intersections which exist in the data (Dupont et al., 2019), and explains some of the differences between generated and ground truth motion. These results demonstrate that our method is sufficiently flexible to morph a simple base attractor from complex demonstrations.

Generalised Collision-Avoiding Motion Reproduction: We evaluate the efficacy of DTs to adapt the motion according to the occupancy of the surroundings and avoid collision. We work with demonstrations from the following tasks: (i) pushing a box (7 demonstrations collected with JACO arm); (ii) lifting a box, by reaching inside and abruptly moving upwards (8 demonstrations collected with JACO arm); (iii) drawing an "S" character (7 demonstrations from LASA dataset). We train DTs to imitate expert demonstrations to complete these tasks. Then, we place an obstacle of diameter 15cm such that the original trajectories are in collision. We then provide occupancy data of the new obstacle, and construct DTs in C-space to provide collision avoidance for the entire manipulator. Velocities in C-space are obtained and updated at 20hz. *Baseline Methods:* We evaluate the performance of the following methods and variants: (i) Our method to imitate demonstrations and avoid collisions. (ii) Our method with only imitation transformation. (iii) Euclideanizing Flows (Eflow) (Rana et al., 2020b). (iv) Eflow modified a repulsive potential field overlaid for collision avoidance; (v) Motion-planning networks (MPNet) (Qureshi et al., 2021); (vi) MPNet with RRT replanning (MPNet-RRT): a variant where results are touched up by a RRT planner (Lavalle, 1998).

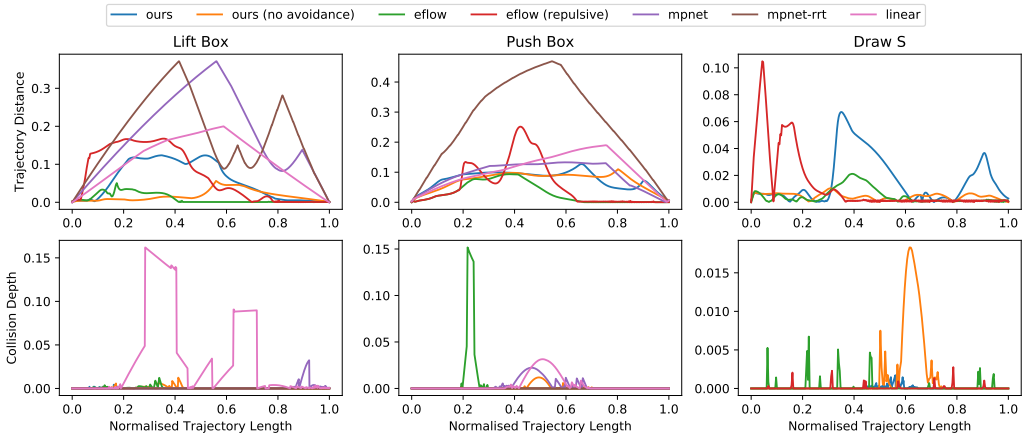


Figure 6: Qualitative results of diffeomorphic transforms along with comparisons. We normalise the generated trajectories by arc-length, and report the trajectory distance and M.C.D. at fixed intervals. For the draw "S" task, results for MPNet, MPNet-RRT, and linear are discarded. These methods were unable to imitate, and departed greatly from the ground truth.

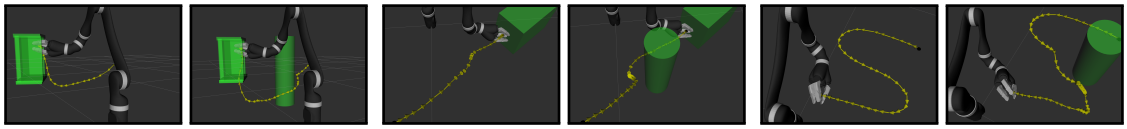
(vi) Linear motion in the C-space. Although MPNet is typically used for planning, at its core, it uses a neural network to imitate a motion planner.

Table 1: Frechet Distance (F.D.) and Maximum Collision Distance (M.C.D.) performance.

	Distance	Push Box	Lift Box	Draw S
Ours	F.D	0.124	0.127	0.067
	M.C.D	0	0	0.002
Ours (no avoidance)	F.D	0.070	0.112	0.011
	M.C.D	0.012	0.012	0.018
EFlow	F.D	0.068	0.093	0.021
	M.C.D	0.012	0.152	0.007
EFlow (repulsive)	F.D	0.168	0.252	0.195
	M.C.D	0	0.001	0.003
MPNet	F.D	0.371	0.133	N/A
	M.C.D	0.038	0.022	N/A
MPNet-RRT	F.D	0.371	0.478	N/A
	M.C.D	0	0	N/A
Linear	F.D	0.242	0.205	N/A
	M.C.D	0.147	0.031	N/A

Evaluation Results: The quality of the generalised imitations produced under environment occupancy changes are shown in fig. 6, and tabulated in table 1. In fig. 6, the metrics used are: (i) trajectory distance (metres), the distance between a point on a generated trajectory and the nearest coordinate of a demonstration; (ii) Maximum Collision Distance (M.C.D.), the robot depth in collision with the obstacle. We evaluate both metrics at fixed intervals along arc-length normalised generated trajectories. In table 1, we report: (i) the Fréchet distance (Alt and Godau, 1995) between the generated and ground truth trajectories, which gives a single value over the entire trajectories; (ii) M.C.D. For pure imitation, our method generates trajectories with a relatively low Frechet Distance to the ground truth, comparable to Eflow. After factoring in the obstacle, our method is able to remain relatively close to the original demonstration, without large changes. In contrast, Eflow (repulsive) typically has a higher distance, particularly when an obstacle is reached, the repulsion results in sharper and larger changes, while our method manages to diverge from the original demonstration in a smoother manner. This is highlighted in the sharper peaks of trajectory distance in fig. 6 by Eflow (repulsive). Additionally, unlike our approach, Eflow with added repulsor does not have theoretical guarentees on stability. Notably the collision-avoidance capability of MPNet (without RRT-replanning) is significantly less, highlighting the difficulty of directly learning from collision-avoiding motions. Particularly in an imitation learning setup, where the number of demonstrations is small. In fig. 7, we illustrate generated motion in both the absence and presence of added obstacles.

Evaluation on Real Robot: To evaluate the robustness of our method, we repeat the three tasks on a real JACO arm. For each tasks, the JACO arm warped around new obstacles with no visible collision, successfully completing the task. A video of the generalised motions on the JACO can be found at https://youtu.be/LoLQ0bzfw9I.



(*a*) Lift Box (*b*) Push Box (*c*) Draw S

Figure 7: The simulated environments under the MoveIt framework (Coleman et al., 2014)

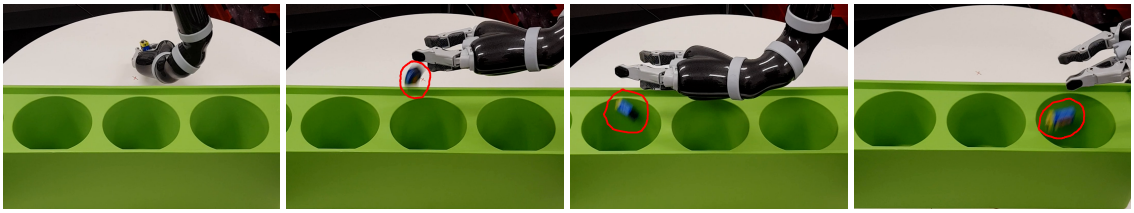


Figure 8: We provide 8 demonstrations of dropping moving over the center pot to drop an object, we can create user-specific DTs to warp the reproduced movement to drop the object in the left or right pots. Left-most image shows starting position. The dropped objects are circled in red.

Generalised User-Influenced Motion Reproduction: We also demonstrate the ability DTs possess to be built based on user specifications. A simulated case studied of warping a dynamic system trained on imitations of drawing "J" characters, based user specified points, is illustrated in fig. 4. The original system is morphed to bias the designated positions. We extend our experiments to real-world robots, with the JACO: We provide demonstrations of moving on top of a pot to drop an object. Then, we craft DTs to morph the motion to either side such that the reproduced motion drops the object in neighbouring pots. Without additional demonstrations, the robot successfully drops the object in the correct pots (fig. 8).

6. Conclusions and Future Work

Diffeomorphisms provide a principled method to transform dynamical systems such that asymptotic stability properties are preserved. We utilise this property and propose a framework which generalises imitation learning when changes in the environment renders collected demonstrations infeasible. We modularise motion using DTs, where each DT encodes specific desirable behaviours, including imitating demonstrations, avoiding obstacles, and incorporating specified biases. We demonstrate the capabilities of this novel approach on a range of generalised imitation learning tasks in both simulation and on a real robot. Avenues of future work may include: (1) isolating movement styles from different sets of demonstrations performing the same action, by a subtraction, rather than a combination of DTs; (2) integrating into our framework invertible mapping with ODEs which have augmented states for additional expressivity, akin to the *augmented neural ODEs* method in (Dupont et al., 2019).

References

H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 1995.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 1998.

Aude Billard and Daniel Grollman. *Imitation Learning in Robots*. Springer US, 2012.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2007.

S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 2016.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

E. Coddington and N. Levinson. Theory of ordinary differential equations. 1955.

D. Coleman, I. A. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering for Robotics*, 2014.

J Denavit, RS Hartenberg, BW Mooring, GR Tang, DE Whitney, and CA Lozinski. 54 kinematic parameter. *Journal of applied mechanics*, 1955.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Advances in Neural Information Processing Systems*, 2019.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

Heiko Hoffmann, Peter Pastor, Dae-Hyung Park, and Stefan Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, 2009.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 2011.

S. M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 2012.

S. M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics Auton. Syst.*, 62:752–765, 2014.

Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Department, Iowa State University, 1998.

Steven M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

J. M. Lee. Introduction to smooth manifolds. 2012.

T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. 2018.

A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 2021.

F. Ramos and Lionel Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 2016.

M. A. Rana, Anqi Li, D. Fox, S. Chernova, Byron Boots, and Nathan D. Ratliff. Towards coordinated robot motions: End-to-end learning of motion policies on transform trees. *ArXiv*, abs/2012.13457, 2020a.

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 *Proceedings of the Conference on Robot Learning*, pages 1457–1468, 2019.

Muhammad Asif Rana, Mustafa Mukadam, Seyed Reza Ahmadzadeh, Sonia Chernova, and Byron Boots. Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020b.

Nathan Ratliff. Learning geometric reductions for planning and control. In *ICML 2013 Workshop on Robot Learning*, 2013.

Nathan D. Ratliff, Jan Issac, and Daniel Kappler. Riemannian motion policies. *CoRR*, abs/1801.02854, 2018.

Julen Urain, Michele Ginesi, Davide Tateo, and Jan Peters. Imitationflows: Learning deep stable stochastic dynamic systems by normalizing flows. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

Wenrui Wang, Mingchao Zhu, Xiaoming Wang, Shuai He, Junpei He, and Zhenbang Xu. An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators. *International Journal of Advanced Robotic Systems*, 2018.

Weiming Zhi, Lionel Ott, Ransalu Senanayake, and Fabio Ramos. Continuous occupancy map fusion with fast bayesian hilbert maps. In *International Conference on Robotics and Automation (ICRA)*, 2019.

Weiming Zhi, Tin Lai, Lionel Ott, Edwin V. Bonilla, and Fabio Ramos. Learning odes via diffeomorphisms for fast and robust integration. *CoRR*, abs/2107.01650, 2021.