Learning Orbitally Stable Systems for Diagrammatic Teaching

Weiming Zhi[1,*] Kangni Liu[1] Tianyi Zhang[1] Matthew Johnson-Roberson[1]

***Abstract*— Diagrammatic Teaching is a paradigm for robots to acquire novel skills, whereby the user provides 2D sketches over images of the scene to shape the robot's motion. In this work, we tackle the problem of teaching a robot to approach a surface and then follow cyclic motion on it, where the cycle of the motion can be arbitrarily specified by a single user-provided sketch over an image from the robot's camera. Accordingly, we introduce the *Stable Diffeomorphic Diagrammatic Teaching* (SDDT) framework. SDDT models the robot's motion as an *Orbitally Asymptotically Stable* (O.A.S.) dynamical system that learns to follow the user-specified sketch. This is achieved by applying a *diffeomorphism*, i.e. a differentiable and invertible function, to morph a known O.A.S. system. The parameterised diffeomorphism is then optimised with respect to the Hausdorff distance between the limit cycle of our modelled system and the sketch, to produce the desired robot motion. We provide theoretical insight into the behaviour of the optimised system and also empirically evaluate SDDT, both in simulation and on a quadruped with a mounted 6-DOF manipulator. Results show that we can diagrammatically teach complex cyclic motion patterns with a high degree of accuracy.**

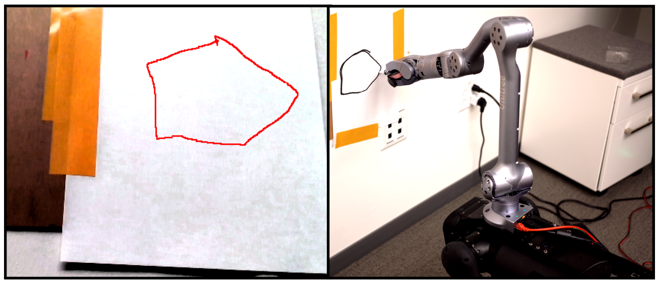


Fig. 1: Diagrammatic teaching is a paradigm to interface with robots by drawing sketches over camera images. We contribute SDDT to diagrammatically teach robots robot policies that approach a surface in view and stabilise at cyclic motions of the provided shape on the surface. (Left) A sketch of the desired pentagon-shaped cycle (in red) is provided by the user from the egocentric view of the robot. (Right) The resulting policy forces the end-effector to quickly approach the surface, and then stabilise to continuously trace out the shape of the provided sketch.

I. Introduction

Specifying the desired behaviour for a robot has traditionally involved crafting a cost function and solving an optimisation problem. This process can often be complicated and require trial and error. Another way to generate desired robot motion has been Learning from Demonstration (LfD) [1], where an expert demonstrates the movement to the robot. The demonstrations are often provided by *Kinesthetic Teaching*, where the user physically handles the robot, or via teleoperation, which requires an additional remote controller. Both approaches face challenges when operating on robots with high degrees of freedom. Diagrammatic Teaching [2] is a recently introduced paradigm that circumvents physical contact and teleoperation, where the user specifies robot skills by sketching examples of the robot's end-effector motion on images of the scene. Correspondingly, in this paper, we seek to use user sketches as a medium for the user to shape the motion of the robot.

We focus on generating robot end-effector motion that approaches a flat surface and converges to a continuous periodic motion on it. This motion pattern arises in many tasks, such as painting, wiping or sanding a surface. We represent the robot's end-effector motion as a dynamical system where trajectories of this system will eventually converge to the limit cycle. Dynamical systems with this convergence property are known to be *Orbitally Asymptotically Stable* (O.A.S.). This paper aims to develop methodologies to learn robot policies, represented by O.A.S. dynamical systems, that are shaped by diagrammatic sketches provided by the user.

We introduce *Stable Diffeomorphic Diagrammatic Teaching* (SDDT), a novel framework for diagrammatic teaching to learn policies that produce periodic and surface-approaching motions. SDDT allows the user to provide input by sketching the shape of the desired limit cycle onto an image of the surface. We use the insight that when two dynamical systems map to one another via a *diffeomorphism* (a differentiable and invertible function), these systems have the same stability properties [3], [4]. SDDT learns O.A.S. systems by optimising a parameterised diffeomorphism to "morph" a system that is known to be O.A.S., such that the limit cycle matches the desired shape. We develop a loss for the corresponding optimisation. Then, we provide both theoretical guarantees into what classes of shapes the limit cycle can be "morphed" into and empirical evidence of the efficacy of our proposed framework, including real-world experiments on a quadruped with a mounted manipulator. SDDT is particularly appealing for mobile manipulators (like quadrupeds with installed arms fig. 1), where an egocentric view, readily available from the onboard vision system, is used to prompt diagrammatic sketches from the user.

The remainder of the paper is organised as follows: we begin by reviewing related work in section II, and then provide some background knowledge to understand SDDT in section III. We introduce the methodology of SDDT and provide theoretical results in section IV, followed by empirical results in section V. We end the paper with conclusions and future directions in section VI.

*email: wzhi@andrew.cmu.edu.
[1] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA

II. Related Work

A. Robot Motion Generation and Diagrammatic Teaching

Generating robot motion is a central problem in robotics. This can be done in a motion planning fashion [5], where an optimisation problem needs to be constructed and a motion planner [6], [7], [8] is then used to find a solution to the problem. Motion planning approaches are beneficial in that once the constraints and costs for the motion have been specified, the task of motion generation is primarily “off-loaded” to the planner, and the solution inherits theoretical guarantees, such as probabilistic completeness [9]. However, many natural motion patterns cannot be easily distilled into a simple cost function, and additionally, the construction of the optimisation problem requires technical expertise. Another approach for specifying robot motion has been Learning from Demonstration (LfD) [1], where a human expert physically handles the robot to trace out the desired motion (i.e. kinesthetic teaching) [10], [11] or teleoperation via specialised remote controllers [12]. To allow demonstrations to be more easily and intuitively collected, *Diagrammatic Teaching* [2] has been introduced as an alternative interface for the user to specify movement patterns to robots: the user is provided with images and prompted to provide sketches, which are subsequently used to construct a model of robot motions. This work falls within the Diagrammatic Teaching paradigm, where the robot’s motion is shaped by a sketch from the user.

B. Stable Dynamical Systems as Robot Policies

In many LfD and motion generation problem formulations, robot policies are modelled by state-dependent dynamical systems [13], [14]. Enforcing the convergence properties of dynamical systems has been shown to increase the robustness of the robot policy and enables prior knowledge to be imbued into the system [15]. However, previous methods exclusively focus on systems that converge to a single fixed point, instead of an orbit. Additionally, these systems are learned in a LfD setup where the training data is a set of collected expert trajectories, in the form of sequences of end-effector or joint positions. Examples of such methods include [16], [13], [17], [18], [3], In particular, methods [3], [18] also take a diffeomorphic learning approach, but only considered stability with respect to convergence to a fixed point. A similar approach was provided in [19] which provided extensions to learning stochastic systems with stability properties from multiple kinesthetic demonstrations and examined stable orbits. Our work is unique from these previous approaches, in that we study learning stable systems within the diagrammatic teaching paradigm. Our approach does not require multiple kinesthetic demonstrations, and instead simply requires a single sketch provided by the user.

III. Preliminaries

Here, we introduce the necessary background concepts for the presentation of SDDT in section IV.

A. Robot Motion Generation via Dynamical Systems

In this work, we shall directly model the robot’s end-effector position $\mathbf{x} \in \mathbb{R}^3$. We represent the robot’s policy as a first-order time-invariant dynamical system.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \qquad \dot{\mathbf{x}}(0) = \mathbf{x}_0, \tag{1}$$

where $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a non-linear mapping between $\mathbf{x}$ and its time derivative $\dot{\mathbf{x}}$, and $\mathbf{x}_0$ is the initial condition. Individual motion trajectories ξ of time duration $t \in \mathbb{R}$ can be obtained via integration,

$$\xi(t, \mathbf{x}_0) = \mathbf{x}_0 + \int_0^t \dot{\mathbf{x}}(s)\mathrm{s}, \tag{2}$$

where the integral can be evaluated using a numerical ODE integrator, such as Euler’s method. Modelling the robot’s policy, rather than individual trajectories, has the benefit of being robust to perturbations. That is, at any end-effector state after perturbation, the robot can follow the dynamical system and does not track a pre-determined trajectory. In our problem setup, we may wish to additionally constrain the end-effector to be orthogonal to the surface, fixing its rotation.

B. O.A.S. Systems

We are interested in understanding the long-term behaviour of the dynamical system, namely, what happens to the trajectories after a long integration duration. Will the solution eventually converge to fixed points, a limit cycle, or diverge and blow up? We are interested in robot motion which approaches a surface and converges onto a cyclic motion on that surface. This requires the dynamical system to be *Orbitally Asymptotically Stable (O.A.S.)* with a limit cycle. Here, we give a definition for O.A.S.

Definition 3.1 (O.A.S. Stability): A dynamical system $\mathbf{x} = f(\mathbf{x})$ is Orbitally Asymptotically Stable (O.A.S.) if for any initial condition $\mathbf{x}_0$ within a region of state-space $\mathcal{X}$, we have

$$\lim_{t \rightarrow \inf} \min_{\tau \in [0,T]} ||\mathbf{x}(t) - \bar{\mathbf{x}}(\tau)|| = 0, \tag{3}$$

where $\bar{\mathbf{x}}(\tau)$ is a solution of the system. Furthermore, $\bar{\mathbf{x}}(\tau)$ is periodic, i.e. $\bar{\mathbf{x}}(\tau) = \bar{\mathbf{x}}(\tau + T)$, where $T \in \mathbb{R}^+$ is the period of the cycle. Here, $\bar{\mathbf{x}}(\tau)$ is known as a “limit cycle” and $\mathcal{X}$ is known as the “basin of attraction”.

C. Invertible Neural Networks

Diffeomorphisms, which are differentiable and invertible functions, are crucial building blocks of our proposed framework. Diffeomorphisms can be parameterised by Invertible Neural Networks (INNs). INNs are function approximators that are invertible by definition and have easily computable Jacobians [20]. We use *Coupling-based INNs* [21] which contain the reversible block introduced, where the split the input $\mathbf{u}$ into halves, $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2]$, and the output $\mathbf{v}$ into halves, $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2]$. We learn four fully-connected neural networks, p_1, p_2, q_1, q_2 such that,

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(p_2(\mathbf{u}_2)) + q_2(\mathbf{u}_2), \tag{4}$$
$$\mathbf{v}_2 = \mathbf{u}_2 \odot \exp(p_1(\mathbf{v}_1)) + q_1(\mathbf{v}_2), \tag{5}$$

where $\odot$ denotes the Hadamard product. By construction, the inverse is given as follows:

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(p_2(\mathbf{u}_2)) + q_2(\mathbf{u}_2), \tag{6}$$
$$\mathbf{v}_2 = \mathbf{u}_2 \odot \exp(p_1(\mathbf{v}_1)) + q_1(\mathbf{v}_2). \tag{7}$$

As such, the INN is able to enforce invertibility without the functions p_1, p_2, q_1, q_2 being invertible.

IV. Stable Diffeomorphic Diagrammatic Teaching

Stable Diffeomorphic Diagrammatic Teaching (SDDT) first presents the user with an image of the contact surface and prompts the user to sketch a closed shape on the image. The corresponding points in the robot's task space are found via ray-tracing the sketch onto the surface. We then minimise the distance between the limit cycle of a parameterised O.A.S. system and the set of corresponding points.

This section is organised as follows: We shall first elaborate on how to construct a parameterised 3D dynamical system which is O.A.S. with a stable orbit on a surface (section IV-A). We describe how to shape the system to match a sketch provided by the user (section IV-B). Then, we provide theoretical guarantees that our model is sufficiently flexible to model any limit cycle that is smooth and closed (section IV-E).

A. Parameterising O.A.S. Systems via Diffeomorphisms

We can learn a desired O.A.S. system $\dot{\mathbf{x}} = f(\mathbf{x})$, by starting with a hand-designed *base* O.A.S. system $\dot{\mathbf{y}} = g(\mathbf{y})$. We then learn a diffeomorphism F such that $\mathbf{x} = F(\mathbf{y})$. Intuitively, we can think of the diffeomorphism to be "morphing" the base system into the desired system. A simple illustration of this is provided in fig. 2. Throughout this section, we denote the state variables of the base system as $\mathbf{y}$, and that of the desired system as $\mathbf{x}$.

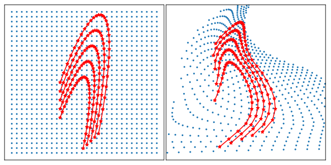


Fig. 2: Diffeomorphisms can be thought of as "morphing" a dynamical system into one another. (Left) Five trajectories (red) of overlaid on grid points (blue); (Right) Morphed trajectories and the corresponding grid.

In this paper, we will by convention define the flat surface as the x, z-plane at $y = 0$. We begin by constructing a simple base system to have a stable circular orbit on the surface. Consider a system where a polar coordinate system (with polar variables r and ω) is defined in the x, z-plane, with an additional attractor in the y-axis:

$$\dot{r} = \mu(1 - \frac{r^2}{R^2})r, \qquad \dot{\omega} = 1, \qquad \dot{y} = -\alpha y, \tag{8}$$

where $\mu > 0$ and $\alpha > 0$ are parameters which control how fast the system converges. Trajectories of this system will converge to an equilibrium at $r = R$, $y = 0$, and any ω, for all $r \geq 0$. This system is O.A.S., with a basin of attraction $\mathcal{X} = \{(r, \omega, y) | r > 0, \omega, y \in \mathbb{R}\}$. Example trajectories of this system are shown in fig. 3. We can transform the coordinates into Cartesian coordinates as the system:

$$\dot{\mathbf{y}} = g(\mathbf{y}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -z + \mu\left(1 - \frac{x^2+z^2}{R^2}\right)x, \\ -\alpha y, \\ x + \mu\left(1 - \frac{x^2+z^2}{R^2}\right)z, \end{bmatrix}. \tag{9}$$

which has a limit cycle $L := \{(x, y, z) \in \mathbb{R}^3 | x^2 + z^2 = R^2, y = 0\}$.

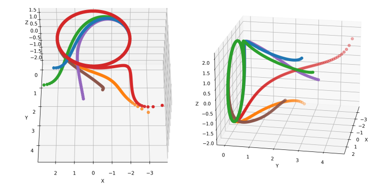


Fig. 3: Trajectories converge to a limit cycle at $y = 0$.

Dynamical systems with state variables satisfying $\mathbf{x} = F(\mathbf{y})$, where F is a diffeomorphism, are topologically conjugates of one another. They can be thought of as the same system under a change of coordinate systems and their stability characteristics are not altered (proof in [4], [18], [22]). We seek a mapping F, such that no change is made on the y axis while shaping the circle on the x, z plane $x^2 + z^2 = R^2$ to match the provided data. Therefore, we decompose F into separate functions, using an INN to learn a diffeomorphism on the x, z axes and leaving the y-axis with the identity function. Specifically,

$$\mathbf{x} = F(\mathbf{y}), \quad \text{where } [x_x, x_z] = \text{INN}_{\boldsymbol{\theta}}([y_x, y_z]), \quad x_y = y_y, \tag{10}$$

where $\mathbf{x} = [x_x, x_y, x_z]$, $\mathbf{y} = [y_x, y_y, y_z]$ and $INN_{\boldsymbol{\theta}}$ is an invertible neural network with parameters $\boldsymbol{\theta}$.

The desired O.A.S. system dynamics $\dot{\mathbf{x}} = f(\mathbf{x})$ is now related to that of the base O.A.S. system $\dot{\mathbf{y}} = g(\mathbf{y})$ via the chain rule:

$$\dot{\mathbf{x}} = f(\mathbf{x}) = J_F(F^{-1}(\mathbf{x}))g(F^{-1}(\mathbf{x})), \tag{11}$$

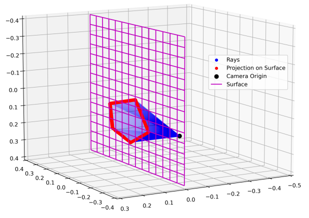
where $J_F(F^{-1}(\mathbf{x}))$ is the Jacobian of F at $F^{-1}(\mathbf{x})$.

B. Learning the Parameterised System via the Hausdorff Distance

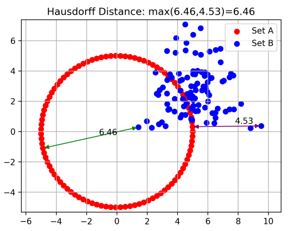
This section elaborates on how to train F defined in eq. (10) such that the limit cycle of $\dot{\mathbf{x}} = f(\mathbf{x})$ matches the user's sketch. This involves projecting the user's sketch to the surface via ray-tracing. Then, defining and minimising a loss between the set of projected points on the surface to the limit cycle of our dynamical system model.

C. Ray-tracing onto Surface

After prompting the user to sketch the desired limit cycle on the surface on the camera image, we are assumed to have a set of n 2D coordinates, $\mathbf{p}$, and corresponding depths, d, to



(a) Visualisation of ray-tracing (rays in blue) a pentagon shape (in red) onto a surface (magenta).



(b) Example contributing distances (green, purple) of the Hausdorff distance between two (red and blue) point sets.

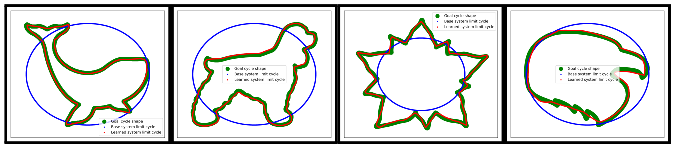


Fig. 5: Our learned cycles on the x, z-plane. Cycle of the base in blue, the target data in green, and the cycle learned in red.

the surface, i.e. $\{\mathbf{p}_i, d_i\}_{i=1}^n$. We call the set of 2D coordinates the *view-space shape*. We follow the [2] and assume a pinhole camera. We construct a ray in 3D which passes through each 2D coordinate, $\mathbf{s}_i = \mathbf{o} + r(\mathbf{p}_i)d_i$, where $\mathbf{o}$ and r are camera origin and projection direction respectively. These are obtainable from the camera parameters and camera position. We collect the set of projected points, $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^n$, and use this as our training data. Note that as we align the flat surface to be the x, z-plane at $y = 0$, by convention, we drop the y-axis of our projected points and simply consider the 2D coordinates of the sketch on the surface. Figure 4a shows an example of projecting a pentagon shape from view-space to a surface, with the traced rays in blue and the pentagon on the surface in red.

D. Hausdorff Distance Loss

The main component of the loss function is a measure of similarity between the shape specified by the user and the limit cycle. This requires us to define a distance between the set of sketched points projected onto the surface, S, and the limit cycle, L. Here, we compute a discretised Hausdorff distance [23], which provides distance between two point sets. Intuitively, the Haussdorff distance takes the larger of the maximum distances from one set to the other, and its reverse. A visualisation of this intuition is given in fig. 4b. This is defined as:

$$H(S, L) = \max\left\{\max_{\mathbf{s}\in S}\min_{\mathbf{l}\in L} D(\mathbf{s},\mathbf{l}), \max_{\mathbf{l}\in L}\min_{\mathbf{s}\in S} D(\mathbf{l},\mathbf{s})\right\}, \quad (12)$$

where D is a metric distance between individual elements, here, we simply use the $L2$ distance.

To prevent the INN from learning diffeomorphisms that excessively distort the base system, we can regularise the diffeomorphism towards the identity function. This can be done by adding an additional regularisation term minimising the difference between $\mathbf{x}$ and $F^{-1}(\mathbf{x})$. This term can be computed via uniformly drawing m samples, $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_m$, within a region of interest in state-space and evaluating their mean distances between $F^{-1}(\hat{\mathbf{x}}_1), \ldots, F^{-1}(\hat{\mathbf{x}}_m)$. We arrive at the combined loss function:

$$\ell = H(S, L) + \alpha\frac{1}{m}\sum_{i=1}^{m}||\hat{x}_i - F^{-1}(\hat{\mathbf{x}}_i)||_2^2, \quad (13)$$

where α controls the regularisation strength. We can then apply gradient-based optimisers, such as ADAM [24], to learn the weights of the INN.

E. Theoretical Guarantees on Learning Cycles

A particular question of interest is: What classes of 2D shapes can we find a diffeomorphism, to "morph" our limit cycle of the base system in eq. (9), at $y = 0$ into?

We show that any 2D shape that can be represented by a smooth and non-intersecting closed curve is diffeomorphic with the unit circle, and hence can, in theory, be "morphed" into by the limit cycle of our base system.

Proposition 4.1: Let $\mathcal{C}$ be a smooth and non-intersecting closed curve in $\mathbb{R}^2$. Then, $\mathcal{C}$ is diffeomorphic to the unit circle $\mathcal{S}^1 := \{(u, v) \in \mathbb{R}^2 | u^2 + v^2 = 1\}$.

Proof: **Parameterise the curve, and extend to a periodic function**: Since $\mathcal{C}$ is a smooth curve in $\mathbb{R}^2$, we can find a parameterisation $\gamma : [0, 1) \to \mathbb{R}^2$, such that $\gamma'(t) \neq 0$ for $t \in [0, 1)$. As $\mathcal{C}$ is closed, we can extend γ to be periodic function on all of $\mathbb{R}$, such that $\gamma(t + 1) = \gamma(t)$ for $t \in \mathbb{R}$. Additionally, as $\mathcal{C}$ is also non-intersecting, each point on $\mathcal{C}$ can be uniquely mapped to $t \in [0, 1)$, hence γ is a diffeomorphism.

Map $[0, 1)$ to $\mathcal{S}^1$: Furthermore, consider the mapping $\psi : \mathbb{R} \to \mathcal{S}^1$, given by $\psi(t) = \exp(2\pi i t)$. This is a smooth map that wraps the real line around the circle infinitely many times and has a period of 1. There exists a smooth inverse ψ^{-1} which maps to $[0, 1)$.

Create the diffeomorphism between $\mathcal{C}$ and $\mathcal{S}^1$: Consider the composition $\phi = \gamma^{-1} \circ \psi : \mathcal{C} \to [0, 1) \to \mathcal{S}^1$. This is a diffeomorphism because it is a composition of two diffeomorphisms. ■

Note that this proposition extends to a circle of arbitrary radius as the circle itself would be diffeomorphic to the unit circle. Moreover, the authors of [25] show that coupling-based INNs are *universal diffeomorphism approximators*. Informally, this means that they can approximate any diffeomorphism, ϕ, to arbitrary accuracy.

V. Experimental Results

We experimentally evaluate the performance and test the robustness of our proposed SDDT method, both in simulation and on real-world robots. We begin by empirically investigating the learning capacity of SDDT, and stress-testing it to learn complicated limit cycles (section V-A). We then highlight the benefits of enforcing O.A.S. by comparing SDDT against neural ODEs [26], which parameterise the dynamics of the system as a free-form neural network (section V-A). Lastly, we demonstrate the applicability of

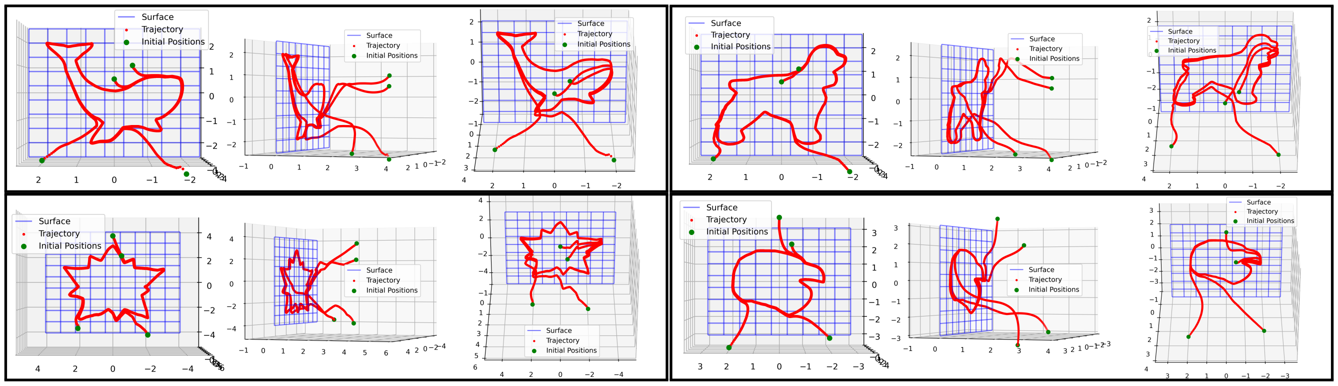


Fig. 6: Qualitative results of learning diffeomorphisms to shape the circular limit cycle into outlines of the **whale**, **dog**, **flower**, and **eagle**. We also show, at different viewing angles, of example trajectories integrated from multiple initial 3D positions (in green). We observe that each trajectory is able to converge onto the shaped limit cycle on the surface.

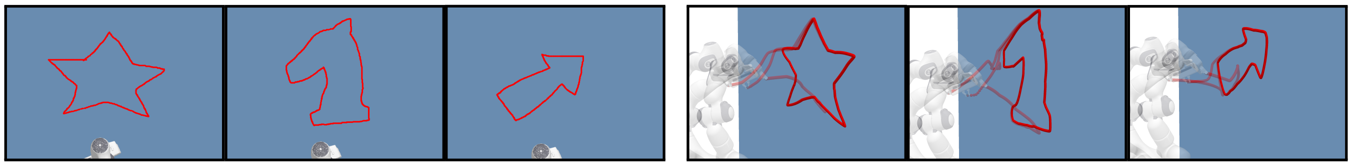


Fig. 7: We evaluate SDDT in the PyBullet Simulator. (Left) The user is provided a view from a camera in the scene and is prompted to sketch a star, a knight chess piece, and an arrow on the image respective. The user sketches (in red) are overlaid over the camera image. (Right) For each different sketch, we generate motion trajectories from SDDT from 3 different robot configurations. We observe that each of these trajectories is able to consistently and accurately converge onto the desired shape.

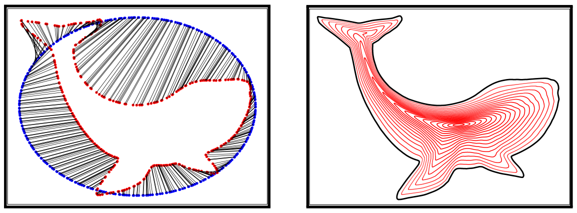


Fig. 8: We visualise how the ambient space is morphed by the learnt diffeomorphism: (Left) Transport map with points on the base system (blue) mapped (shown by grey line) onto those of the learned system (red); (Right) Concentric circles passed through the diffeomorphism to match the desired shape.

TABLE I: Performance of SDDT and baselines, on each task, as measured by Hausdorff distance (lower is better).

	O.A.S.	Star	Knight	Arrow
SDDT (ours)	✓	**0.011**	**0.011**	**0.017**
Neural ODEs [26]	✗	0.040	0.035	0.063
Base System	✓	0.133	0.201	0.209

SDDT in the real world by deploying the framework on a quadruped-mounted manipulator.

A. A Qualitative Analysis: Learning Challenging Cycles

We seek to explore the capabilities of our proposed method for learning systems with limit cycles that are intricate and vary greatly from the circular limit cycle of the base dynamical system. Here, we extract silhouette outlines and investigate how well SDDT is able to "morph" the cycle into the outlines.

We use outlines of a **whale**, a **dog**, a **flower**, and an **eagle** and learn an INN to morph the base system's limit cycle to match the outline. Throughout this paper, we use the INN models in the *FrEIA* library [20], which is built on *PyTorch* [27]. In fig. 6, we provide qualitative results of both the limit cycle and trajectories integrated from the resulting dynamical system. We observe that the limit cycle is generally able to be accurately shaped into each of these outlines, and the integrated 3D trajectories are able to approach the surface and smoothly converge onto the limit cycle. We also visualise and compare the original base system and the target 2D shape, on the x, z-plane in fig. 5.

To gain insight into how the diffeomorphism "morphs" the x, y-plane at $y = 0$, in fig. 8, we visualise additional quantitative results on the **whale** outline. On the left, we show a transport map showing examples of points on the orbit of the base system mapped to the learned system, with correspondence between points on the systems indicated by the grey line. On the right, we visualise the result of diffeomorphism operating on concentric circles (in red), starting from a radius of 0.1 to 2. This gives us an intuition of how the

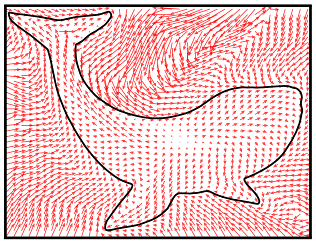


Fig. 10: The learnt O.A.S. vector field on the x, z-plane. The vector field (red arrows) pushes points off the cycle onto the stable cycle (in black).

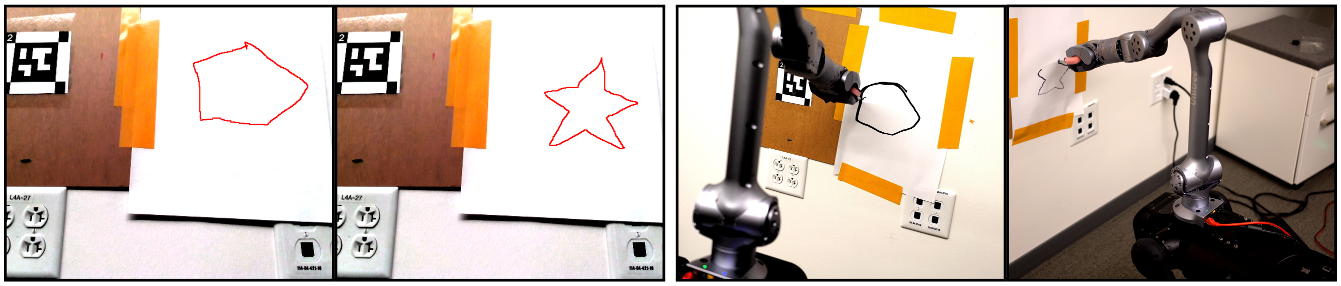


Fig. 9: SDDT is particularly useful when egocentric images, from onboard cameras, are available. We run our real-world experiments on a quadruped with a mounted arm. (Left) We sketch the shapes of a pentagon and a star (in red) on an egocentric view from the onboard camera. (Right) The robot converges to the surface, stabilises at, and traces out the diagrammatically provided shapes.

ambient space is stretched and compressed to match the target shape (in black). In fig. 10, we show the vector field of the resulting learned system, with the limit cycle shown in black. We observe that points not in the vector field shall be attracted towards the stable cycle — points inside the cycle push outwards while those outside push inwards.

B. *SDDT outperforms Free-form Neural ODEs*

We evaluate, in the PyBullet Simulator [28], the performance of the SDDT framework. We simulate a Franka on a mobile base facing a wall, with an RGB-D camera positioned behind the robot. We seek to generate robot motion that approaches the wall and converges onto the shape diagrammatically provided by the user on the camera image. We compare against the following baselines:

1) **Neural ODEs** [26]: Neural ODEs learn dynamical systems by parameterising the dynamics as a neural network and then train on data. We use a Neural ODE to learn the dynamics on the x, z-plane, and retain the attractor towards the surface in the y-axis direction. The dynamics of Neural ODEs are completely free-form, with no assumption made on stability.
2) **Base System** (defined in eq. (9)): We evaluate how well our stable base system performs. We allow the radius and the origin of the base system to be tuned, such that the limit cycle minimises the distance between the data. The base system is highly structured and contrasts with the entirely learning-based Neural ODE.

After collecting sketches of a **star**, a **knight**, and an **arrow**, we train each of these models and integrate trajectories at three different initial robot configurations. Here, we use the implementation of Neural ODEs provided in the *torchdiffeq* library [26]. To measure how well the trajectories from the traced motion match the diagrammatic sketch provided by the user, we take points that have y-values under 10^{-4} to be in contact with the surface, and compute the Hausdorff distance between the trajectory and sketch ray-traced onto the surface. A qualitative evaluation of our generated trajectories, as well as the user-provided sketches, can be seen in fig. 6, with results provided in table I. We observe that SDDT imbues knowledge of stability into the system via enforcing O.A.S., and outperforms Neural ODEs which treat the dynamics as a black-box. SDDT is also sufficiently flexible to learn complex patterns, greatly outperforming the inflexible stable base system.

C. *SDDT on Real Robots*

We demonstrate the applicability of SDDT on real-world robots, by applying SDDT on a *Unitree Aliengo* quadruped with an attached 6-DOF $Z1$ manipulator. The user is shown egocentric views of the environment via the RGB-D camera on board the quadruped and is asked to sketch a **pentagon** and a **star**. SDDT is then used to learn stable systems shaped by the projection of the drawing. We then integrate a trajectory from the current end-effector position and track the trajectory with a marker pen to trace out the corresponding motion patterns. We observe that in both instances the quadruped mount manipulator was able to approach the surface and stabilise on a cycle that matched the diagrammatically specified shapes, despite minor inaccuracies introduced by contact forces. In fig. 9, we overlay the provided sketches onto the egocentric view images from the quadruped and show the manipulator converging onto the surface and tracing the desired shapes.

VI. Conclusions and Future Work

In this work, we tackle the problem of learning robot policies that approach a surface and trace on it periodically. Robot motions of this kind are applicable to painting, wiping, and sanding tasks. We take a *diagrammatic learning* approach, where the type of periodical pattern is provided by the user providing a 2D sketch on an image of the scene. We contribute the novel *Stable Diffeomorphic Diagrammatic Teaching* (SDDT) method, where ray-tracing is used to project the user's sketch onto the surface, and an Orbitally Asymptotically Stable (O.A.S.) dynamical system, which converges to a cyclic orbit, is learned as a policy for the robot's motion. SDDT learns O.A.S. systems by learning a diffeomorphism that morphs a known stable base system into the desired system. We provide theoretical insight into the classes of 2D shapes the stable limit cycle can be shaped into, and provide extensive empirical evaluations of SDDT, both in simulation and on a real-world quadruped with a mounted manipulator. Future avenues of research include: (1) extending SDDT beyond flat surfaces, and to ensure stable motion patterns of curved surfaces; (2) extending SDDT to allow the specification of forces applied onto the surface.

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*, 2020.

[2] W. Zhi, T. Zhang, and M. Johnson-Roberson, "Learning from demonstration via probabilistic diagrammatic teaching," *arXiv*, 2023.

[3] W. Zhi, T. Lai, L. Ott, and F. Ramos, "Diffeomorphic transforms for generalised imitation learning," in *Learning for Dynamics and Control Conference, L4DC*, 2022.

[4] W. Zhi, T. Lai, L. Ott, E. V. Bonilla, and F. Ramos, "Learning efficient and robust ordinary differential equations via invertible neural networks," in *International Conference on Machine Learning, ICML*, 2022.

[5] S. M. LaValle, *Planning Algorithms*. USA: Cambridge University Press, 2006.

[6] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, 2001.

[7] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation*, 2009.

[8] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, 2011.

[10] J. Kober and J. Peters, "Learning motor primitives for robotics," in *IEEE International Conference on Robotics and Automation*, 2009.

[11] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013.

[12] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *Conference on Robot Learning*, 2018.

[13] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, and N. D. Ratliff, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, 2022.

[14] W. Zhi, I. Akinola, K. van Wyk, N. Ratliff, and F. Ramos, "Global and reactive motion generation with geometric fabric command sequences," in *IEEE International Conference on Robotics and Automation, ICRA*, 2023.

[15] V. Sindhwani, S. Tu, and M. Khansari, "Learning contracting vector fields for stable imitation learning," 2018.

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

[17] M. Saveriano, "An energy-based approach to ensure the stability of learned dynamical systems," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[18] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020.

[19] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflows: Learning deep stable stochastic dynamic systems by normalizing flows," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[20] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," in *International Conference on Learning Representations*, 2019.

[21] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations*, 2017.

[22] J. M. Lee, *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics, 2000.

[23] F. Hausdorff, *Grundzüge der Mengenlehre*. Leipzig: Veit & Co, 1914.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv*, 2015.

[25] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama, "Coupling-based invertible neural networks are universal diffeomorphism approximators," in *Advances in Neural Information Processing Systems*, 2020.

[26] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, 2018.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[28] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2019.