Submitted, 2012 International Conference on Robotics and Automation (ICRA)
http://www.cds.caltech.edu/~murray/papers/cm12-icra.html

Learning diffeomorphism models of robotic sensorimotor cascades

Andrea Censi Richard M. Murray

***Abstract*—The problem of bootstrapping consists in designing agents that can learn from scratch the model of their sensorimotor cascade (the series of robot actuators, the external world, and the robot sensors) and use it to achieve useful tasks. In principle, we would want to design agents that can work for any robot dynamics and any robot sensor(s). One of the difficulties of this problem is the fact that the observations are very high dimensional, the dynamics is nonlinear, and there is a wide range of "representation nuisances" to which we would want the agent to be robust. In this paper, we model the dynamics of sensorimotor cascades using diffeomorphisms of the sensel space. We show that this model captures the dynamics of camera and range-finder data, that it can be used for long-term predictions, and that it can capture nonlinear phenomena such as a limited field of view. Moreover, by analyzing the learned diffeomorphisms it is possible to recover the "linear structure" of the dynamics in a manner which is independent of the commands representation.**

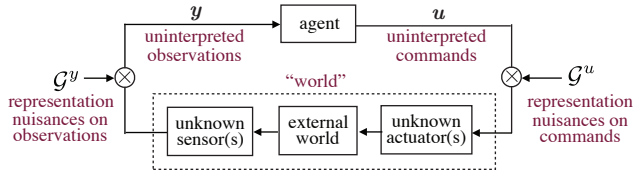


Figure 1. In the bootstrapping scenario, we assume that the agent only has access to uninterpreted streams of observations and commands, and it does not have any prior knowledge of the "world" (the series of the robot actuators, the external world, and the robot sensors). We are looking for models general enough to approximate the dynamics of any robot sensor and actuator, yet simple enough to allow efficient learning. Moreover, we want these models to be invariant to what we call "representation nuisances"; these are static invertible transformations that change the representation but preserve the informative content of the observations and commands signals. Identifying the classes of transformations to which the agent is not invariant allows to recognize the hidden assumptions of the models used by the agent.

I. Introduction

In the problem of bootstrapping, we ask whether it is possible for an agent that wakes up connected to uninterpreted streams of commands and observations, to learn a model of its sensorimotor cascade (the series of its actuators, the world dynamics, and its sensors) and then use it to achieve useful tasks, with no prior information of its unknown (robotic) body and the external world (Fig. 1). Learning to use unknown sensors and actuators is one ability of natural intelligence [1], [2] that we are not able to replicate yet, and it could be the key to make robots more reliable, for example by being able to react to unexpected sensor and actuator faults [3].

The fact that the world states are not observable, and we do not have available behaviorally relevant features of the observations prevents the direct application of established learning techniques, such as reinforcement learning [4]. In fact, one of the key problems of bootstrapping consists in figuring out what are the hidden states in the world that provide a causal explanation of the uninterpreted streams of observations and commands, in terms of an underlying world dynamics. Using the control-theory jargon, we could call this an *identification* problem; however, standard techniques of control theory are not useful in this context, as they assume either linear time-invariant systems (an assumption violated for the simplest of robots) and/or low dimensional data (while robot sensors can provide up to 1GB/s of sensory data).

In the literature, the most promising approach to bootstrapping consists in attacking the problem layer by layer, by constructing hierarchical representations [5]. The first steps consist in understanding from the observed statistics of the data what is a plausible grouping of observations by sensor [6], and recovering the geometry of the sensors [7], [8]; with these steps one can transform scrambled streams into spatially coherent sensor snapshots. The next steps consist in understanding what is the effect of actions on the sensor readings, by progressively abstracting the continuous dynamics into (possibly discrete) primitives [9], [10]. These first steps are independent of the task. Policies based on some form of "intrinsic motivation" [11] allow further understanding of the world, in terms of "distinctive states" [12], "objects" [13], and global world geometry [14]. At this point, given a discrete abstraction in terms of primitives/states, more conventional learning techniques can be used as the top layer in the architecture (e.g., temporal difference learning [15]). While several aspects of what would be a complete bootstrapping architecture have been demonstrated anecdotally, no agent that can *provably* work for general robotic sensors/actuators has been designed.

Previous work: In previous work, we have been trying to approach the problem from a control-theory perspective. We considered three canonical robotic sensors: field-samplers, range-finders, and cameras. Let y^s be the raw readings of such sensors; where $s \in \mathcal{S}$ represent an index over the *sensels* (short for *sensory elements*) which ranges over the *sensel space* $\mathcal{S}$, and y^s is, respectively, field intensity, distance reading, and luminance at sensel s. Surprisingly, at this low level, the three sensor dynamics are quite similar, as shown in Table I, which gives an expression for $\dot{y}^s$ as a function of the robot velocities. This motivated the search for models that could fit all these dynamics at the same time.

In particular, we proposed the class of *bilinear dynamics systems* (BDS) as a possible candidate in [16]. Given a discretized observation vector $\boldsymbol{y} = \{y^i\}_{1 \leq i \leq n_y} \in \mathbb{R}^{n_y}$, and generic commands $\boldsymbol{u} \in \mathbb{R}^{|\boldsymbol{u}|}$, the dynamics are assumed to be

A. Censi and R. Murray are with the Control & Dynamical Systems department, California Institute of Technology, Pasadena, CA. E-mail: {andrea, murray}@cds.caltech.edu.

of the form:

$$\dot{y}^i_t = \sum_j \sum_a M^i_{ja} y^j_t u^a_t + \epsilon^i_t. \qquad \text{(BDS)} \tag{1}$$

The model is parametrized by a $n_y \times n_y \times |\boldsymbol{u}|$ tensor M^i_{ja}. In [17] we studied *bilinear gradient dynamics systems* (BGDS), a more constrained class of models that uses explicitly the spatial organization of the sensels by assuming the dynamics to depend on the spatial gradient $\nabla \boldsymbol{y}$:

$$\dot{y}^s_t = \sum_d \sum_a (G^{ds}_a \nabla_d y^s_t + B^s_a)\ u^a_t + \epsilon^s_t, \qquad \text{(BGDS)} \tag{2}$$

This model is more complicated, but more efficient, as the complexity is linear in n_y instead of quadratic.

These models still suffer from several limitations. The biggest limitation is that the commands $\boldsymbol{u}$ are implicitly assumed to be kinematic velocities (if $\boldsymbol{u} = 0$ then $\dot{\boldsymbol{y}} = 0$). This means that the models are not robust to a change of representation of the commands: even if the commands were kinematic velocities, and the relation (2) held, a reparametrization of the kind $\boldsymbol{u}' = f(\boldsymbol{u})$, for a generic nonlinear function f, would not satisfy a similar relation. The same can be said for nonlinear transformations of the observations. The tolerance of "representation nuisances", shown in Fig. 1 as the transformation groups $\mathcal{G}^u$ and $\mathcal{G}^y$, is the metric with which we measure the power of bootstrapping agents [18]. Another qualitative limitation is that BDS/BGDS models do not represent nonlinear phenomena such as occlusions and limited field of view.

On the more practical side, we have observed that learning the model parameters from an instantaneous relation between derivative and commands like (1) or (2) is not efficient when the robot motion is very slow, because the change in observations is small with respect to the sensor noise. For example, consider a robot moving at 0.5m/s. If the robot has a range-finder updating at 50Hz, the maximum change that we expect to see in the readings is around 1cm, which is comparable to the sensor error.

In this paper, we try to solve some of these limitations by modeling the dynamics of robot sensors as global diffeomorphisms between possibly large time steps. Instead of modeling $\dot{\boldsymbol{y}}_t$ as a function of $\boldsymbol{y}_t$ and $\boldsymbol{u}_t$, we try to model $\boldsymbol{y}_{t_2}$ from $\boldsymbol{y}_{t_1}$ where the interval $t_2 - t_1$ is not necessarily small, and the commands $\boldsymbol{u}_t$ are assumed to be held fixed for $t \in [t_1, t_2]$.

Overview: Section II describes diffeomorphisms dynamical systems (DDS), the classes of models that we use in this work, and how to learn it from data.

Section III describes the application of the theory to camera data. It is shown that the model can represent the basic motions of a differential-drive robot; that the learned model can be used for long-term prediction of observations given a sequence of commands; and that this prediction correctly takes into account the uncertainty due to the limited field of view.

Table I
CONTINUOUS DYNAMICS OF CANONICAL ROBOTIC SENSORS

sensor	$\mathcal{S}$	*continuous dynamics (far from occlusions)*
field sampler	$\mathbb{R}^3$	$\dot{y}^s = \sum_i \nabla_i y^s \boldsymbol{v}^i + \sum_i (s \times \nabla y^s)_i \boldsymbol{\omega}^i$
camera	$\mathbb{S}^2$	$\dot{y}^s = \mu^s \sum_i \nabla_i y^s \boldsymbol{v}^i + \sum_i (s \times \nabla y^s)_i \boldsymbol{\omega}^i$
range-finder	$\mathbb{S}^2$	$\dot{y}^s = \sum_i (\nabla_i \log y^s - s^*_i) \boldsymbol{v}^i + \sum_i (s \times \nabla y^s)_i \boldsymbol{\omega}^i$

$\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ are the robot linear and angular velocities; s is a continuous index ranging over the "sensel space" $\mathcal{S}$; $y(s)$ is the raw sensor data (field intensity value, pixel luminance, or range reading); ∇_i is the i-th component of the spatial gradient with respect to s; $\mu(s)$ is the nearness (inverse of the distance to the obstacles).

Section IV describes how properties of the dynamics can be recovered from the learned models. In particular, using the distance and "anti-distance" between diffeomorphisms, one can identify redundant commands, as well as identify couples of commands that have the opposite effect and that should be grouped together as part of a "reversible" command. Section V describes the application of the theory to range-finder data, using a real robot with a differential-drive, as well as two simulated robots with unicycle and car-like dynamics.

Section VI discusses the invariance properties of the method with respect to transformations of the observations and commands. Finally, Section VII concludes the paper and discusses directions for future work.

II. LEARNING DIFFEOMORPHISMS DYNAMICAL SYSTEMS

This section describes the mathematical preliminaries, the definition of *diffeomorphism dynamical systems* (DDS), and how to learn their parameters from data.

Preliminaries: Let $\mathcal{S}$ be a smooth Riemannian manifold, and let $d_{\mathcal{S}}(s_1, s_2)$ be the geodesic distance between points $s_1, s_2 \in \mathcal{S}$. Let $\mathbb{F}(\mathcal{S})$ be the set of smooth scalar fields defined on $\mathcal{S}$. Let $\mathsf{Diff}(\mathcal{S})$ be the sets of all diffeomorphisms (smooth invertible maps) from $\mathcal{S}$ to itself. A diffeomorphism $\varphi \in \mathsf{Diff}(\mathcal{S})$ maps each point $s \in \mathcal{S}$ to another point $\varphi(s) \in \mathcal{S}$. We define the following distance between diffeomorphisms:

$$\mathcal{D}(\varphi_1, \varphi_2) = \int d_{\mathcal{S}}(\varphi_1(s), \varphi_2(s))\, d\mathcal{S}. \tag{3}$$

A. Diffeomorphisms dynamical systems

We define a *diffeomorphisms dynamical system* (DDS) as a discrete-time dynamical system with state $\boldsymbol{x}_k \in \mathbb{F}(\mathcal{S})$, and a finite commands alphabet $\mathcal{U} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{|\mathcal{U}|}\}$. Each command $\boldsymbol{u}_j$ is associated to a diffeomorphism $\varphi_j \in \mathsf{Diff}(\mathcal{S})$. The transition function from the state $\boldsymbol{x}_k$ at time k to the state $\boldsymbol{x}_{k+1}$ is given by

$$x^s_{k+1} = x^{\varphi(s)}_k, \qquad \text{(DDS)} \tag{4}$$

where φ is the diffeomorphism associated to the command given at time k. The observations $\boldsymbol{y} = \{y^s\}_{s \in \mathcal{S}}$ are assumed to be a *censored* version of $\boldsymbol{x}$, in the sense that we only can see the state in a subset $\mathcal{V} \subset \mathcal{S}$:

$$y^s_k = \begin{cases} \boldsymbol{x}^s_k + \epsilon^s_k & \text{if } s \in \mathcal{V}, \\ 0 & \text{if } s \notin \mathcal{V}, \end{cases} \tag{5}$$

where ϵ^s_k is assumed to be additive gaussian noise. This censoring is needed to model sensors with a limited field of view.

B. Representing and learning DDS

Suppose that we are given training examples, consisting of tuples $\langle \boldsymbol{y}_k, j_k, \boldsymbol{y}_{k+1} \rangle$, meaning that at time k we observed $\boldsymbol{y}_k$, then, after applying the command $\boldsymbol{u}_{j_k}$, we observed $\boldsymbol{y}_k$. Our objective is estimating the diffeomorphisms φ_j.

We assume that the $\mathcal{S}$ domain has been discretized into a finite number n_y of cells $\{s^i\}_{1 \le i \le n_y} \subset \mathcal{S}$, such that the i-th

cell has center s^i. We represent a diffeomorphism φ by[1] its discretized version $\tilde{\varphi} : [1, n_y] \to [1, n_y]$ that associates to each cell s^i another cell $s^{i'}$, such that $i' = \tilde{\varphi}(i)$.

Learning can be done independently cell by cell. In fact, when the j-th command is applied, we expect that, as given by (4)–(5):

$$y^s_{k+1} = y^{\varphi_j(s)}_k.$$

This can be discretized in the following way:

$$y^{s^i}_{k+1} = y^{s^{\tilde{\varphi}_j(i)}}_k.$$

Therefore, the value $\tilde{\varphi}_j(i)$ can be found by maximum-likelihood as follows:

$$\hat{\tilde{\varphi}}_j(i) = \arg\min_{i'} \mathbb{E}_{\boldsymbol{u}=\boldsymbol{u}_j}\{\|y^{s^i}_{k+1} - y^{s^{i'}}_k\|\}. \tag{6}$$

Assuming we have a bound M on the maximum displacement over all diffeomorphisms:

$$\max_{j,s} d_{\mathcal{S}}(s, \varphi_j(s)) \leq M,$$

then the search for i' will not be needed to be extended to all cells in the domain, but only on the neighbors of i such that $d_{\mathcal{S}}(s^i, s^{i'}) \leq M$. This is illustrated in Fig. 2c–2d for 2D domains. In practice, for each command and for each cell, we consider a square neighborhood of cells in which to search for the matching cell.

This simple algorithm has complexity $\mathcal{O}(\rho^{2m} AM)$, where A is the area of the sensor, ρ the resolution, and $m = \dim(\mathcal{S})$, because for each of the $n_y = \rho^m A$ cells, we have to consider a number of neighbors proportional to $\rho^m M$. Still, it is embarrassingly parallel, as the expectations in (6) can be computed separately, therefore it has decent performance if one uses vectorized operations, also in interpreted languages like Python (we obtain 12fps for 100×100 domains and $M = 15\%$ of the domain).

C. Estimating and propagating uncertainty

In principle, one could represent each value $\tilde{\varphi}_j(i)$ as a random variable, and estimate the full posterior distribution. In practice, this complexity is not needed in our application, and we limit ourselves to keeping track of a single scalar measure of uncertainty $\Gamma^{s^i}_j$, which we interpret as being proportional to $\mathrm{Trace}(\mathrm{cov}(\varphi_j(s^i)))$. This uncertainty is computed from the value of the cost function (6):

$$\Gamma^{s^i}_j \simeq \mathbb{E}_{\boldsymbol{u}=\boldsymbol{u}_j}\{\|y^{s^i}_{k+1} - y^{s^{\tilde{\varphi}_j(i)}}_k\|\}.$$

In practice, this simple model allows to represent the uncertainties due to the limited field of view, in fact we will have that $\Gamma^{s^i}_j$ is very large for cells s^i such that $\varphi_j(s^i) \neq \mathcal{V}$; that is, for cells whose values cannot be predicted because they depend on observations outside the field of view $\mathcal{V}$.

[1]The only problem we encountered with this representation concerns the computation of the diffeomorphism inverse. In general, the maps $\tilde{\varphi} : [1, n_y] \to [1, n_y]$ are not invertible because not surjective. The approximation we use for $\tilde{\varphi}^{-1}$ consists in: 1) Averaging over cells if multiple cells correspond to the same cell (that is, for the cells i' such that there are multiple cells for which $\tilde{\varphi}(i_1) = \tilde{\varphi}(i_2) = \cdots = i'$); 2) Interpolate valid neighbor values for cells not in Image($\tilde{\varphi}$) (that is, for the cells i' such that there exist no i for which $\tilde{\varphi}(i) = i'$).

We will also need to propagate the uncertainty. Note that the DDS representation allows to compress a series of commands into one supercommand whose diffeomorphism is the composition of the individual diffeomorphisms. The composition of commands can keep track of the uncertainty as well. Suppose that we have two commands $\boldsymbol{u}_a$ and $\boldsymbol{u}_b$ and that we learned the two corresponding uncertain diffeomorphisms $\langle \tilde{\varphi}_a, \Gamma_a \rangle$ and $\langle \tilde{\varphi}_b, \Gamma_b \rangle$. Then the composite command $\boldsymbol{u}_c = \boldsymbol{u}_b \circ \boldsymbol{u}_a$ will be represented by the pair $\langle \tilde{\varphi}_c, \Gamma_c \rangle$, where $\tilde{\varphi}_c$ is just the composition of $\tilde{\varphi}_a$ and $\tilde{\varphi}_b$:

$$\tilde{\varphi}_c(i) = \tilde{\varphi}_b(\tilde{\varphi}_a(i)),$$

but Γ_c takes into account both a *transport* and a *diffusion* component:

$$\Gamma^{s^i}_c = \Gamma^{s^{\tilde{\varphi}_b}}_a + \Gamma^{s^i}_b.$$

III. Application to camera data

This section describes the application of the theory to camera data for a mobile robot. It is shown that the learned diffeomorphisms capture the motions, as well as the uncertainties due to the limited field of view of the camera. Based on the model, one can obtain long-term predictions of the observations given a sequence of commands, and these predictions correctly take into account the uncertainty due to the limited field of view.

Platform: We use an Evolution Robotics ER1 robot (Fig. 2a), with a cheap web-cam on board that gives 320x240 frames at ~7.5Hz. The robot is driven through a variety of indoor and outdoor environments (Fig. 2b) for a total of around 50 minutes[2]. The robot linear and angular velocities were chosen among the combinations of $\omega \in \{-0.2, 0, +0.2\}$ rad/s and $v \in \{-0.3, 0, +0.3\}$ m/s.

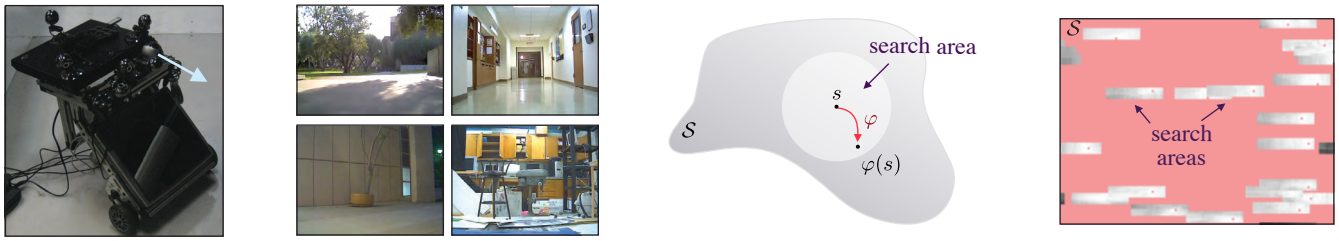
Results: Fig. 3 shows the resulting of the diffeomorphisms learning applied to this data. Not all combinations of commands are displayed; in particular, those corresponding to the robot backing up were not chosen frequently enough to obtain reliable estimates of the corresponding diffeomorphisms.

Because of the fact that we also learn the uncertainty of the diffeomorphisms, we can correctly predict effects due to the limited field of view for the camera. For example, for the first command, corresponding to a pure rotation to the right, we can predict that for 8 time steps we can predict the left half of the image, but we will not know anything about the right half.

IV. Inferring the intrinsic linear structure of the commands space

Assuming a class of models, such as BDS/BGDS, where the commands have a linear effect on the dynamics (for example, if they correspond to kinematic velocities) automatically gives rich structural properties to the commands space. For example, the effect of applying $\boldsymbol{u}' = 2\boldsymbol{u}$ is twice larger than the effect of $\boldsymbol{u}$; the effect of $\boldsymbol{u} = 0$ corresponds to a null action, and the effect of $\boldsymbol{u}' = -\boldsymbol{u}$ is the opposite of $\boldsymbol{u}$. If the commands are kinematic, this structure can be lost if they are represented in a nonlinear way, for example if one has available the commands $\boldsymbol{u}' = f(\boldsymbol{u})$ instead of $\boldsymbol{u}$. In this section, we show how this structure of the commands space is recovered from

[2]The log files used are available at http://purl.org/censi/2011/diffeo.



(a) Robot platform (b) Environments samples (c) Geometry of the problem (d) Algorithmic approximation

Figure 2. (a) For the first set of experiments we use an Evolution Robotics ER1 with an on-board camera. (b) The robot is driven through a variety of indoor and outdoor environments. (c) To learn the diffeomorphisms, we assume to have a bound on the maximum displacement $d(s, \varphi(s))$ on the manifold $\mathcal{S}$. (d) In the implementation, we are limited to square domains. The search area around each point is constrained to be a square, with given width and height, which are tunable parameters that affect the efficiency of the algorithm.

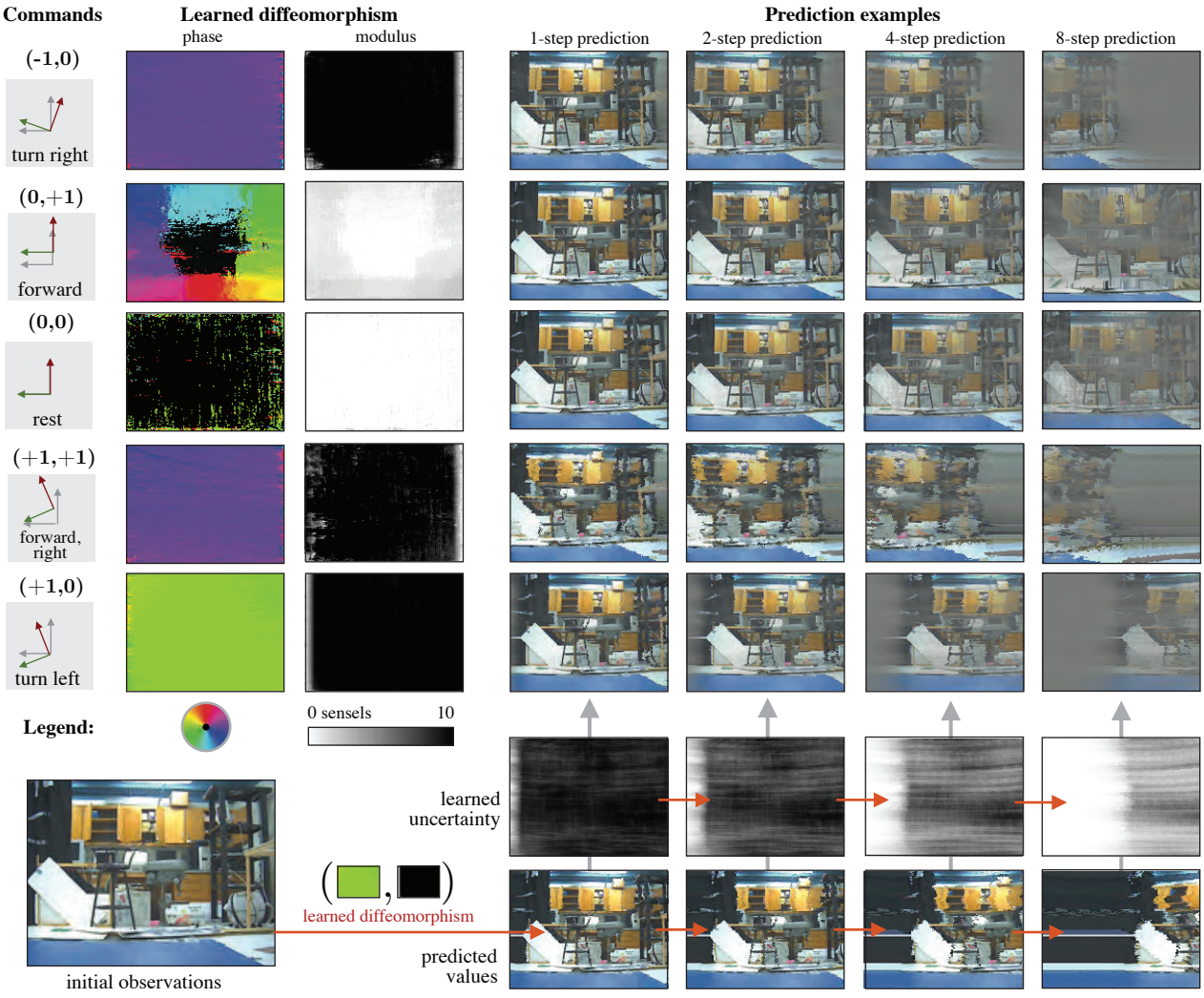


Figure 3. This figure shows a few of the learned diffeomorphisms learned from the camera data. Each row corresponds to a particular command given to the robot. The first column shows the effect of the command on the robot pose. The second and third row show the corresponding diffeomorphism, displayed using phase and modulus. The last four columns show how the learned diffeomorphisms can be used for prediction. The columns show the predicted image following the application of the command for 1, 2, 4, and 8 time steps. The uncertain parts of the predictions are shown in gray. The visualization of this uncertainty is done by propagating both the diffeomorphism uncertainty and the image values, as shown in the last two rows, and then blending the values with a solid gray rectangle according to the predicted uncertainty.

the learned diffeomorphisms if it is present, independently of the commands representation.

Identifying redundant and null commands: We call a pair of commands $\langle \boldsymbol{u}_j, \boldsymbol{u}_k \rangle$ *redundant* if they have the same effect on the observations. If two commands give the same effect, then one of them can be removed from the commands alphabet. Redundant commands can be recognized simply by looking at the distance between the corresponding diffeomorphisms. Formally, we define the distance $\mathcal{D}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k)$ between two commands as the difference between the corresponding diffeomorphisms, normalized by the average distance between all diffeomorphisms:

$$\mathcal{D}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k) = \frac{\mathcal{D}(\varphi_j, \varphi_k)}{\frac{1}{|\mathcal{U}|^2} \sum_l \sum_m \mathcal{D}(\varphi_l, \varphi_m)}.$$

The normalization makes this a unitless quantity that does not depend on the size of $\mathcal{S}$. As a special case, commands that have no effect can be easily identified by considering the distance of their diffeomorphism from the identity diffeomorphism $\mathcal{I}_{\mathcal{S}}$.

Identifying reversible commands: We call a pair of commands $\langle \boldsymbol{u}_j, \boldsymbol{u}_k \rangle$ *reversible* if applying $\boldsymbol{u}_j$ followed by $\boldsymbol{u}_k$ brings the system in the initial state, and vice versa. Two commands would be perfectly reversible if $\varphi_j = \varphi_k^{-1}$ or, equivalently, $\varphi_j^{-1} = \varphi_k$, or $\varphi_k \circ \varphi_j = \varphi_j \circ \varphi_k = \mathcal{I}_{\mathcal{S}}$. All of these conditions are equivalent in the continuum case and without noise, but might give slightly different results in practice due to numerical approximations and estimation noise. Somehow arbitrarily, we choose to define the *anti-distance* of a commands pair $\langle \boldsymbol{u}_j, \boldsymbol{u}_k \rangle$ as

$$\mathcal{A}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k) = \frac{\frac{1}{2}(\mathcal{D}(\varphi_j, \varphi_k^{-1}) + \mathcal{D}(\varphi_j^{-1}, \varphi_k))}{\frac{1}{|\mathcal{U}|^2} \sum_l \sum_m \mathcal{D}(\varphi_l, \varphi_m)}.$$

If the anti-distance is zero, then the command pair is a perfectly reversible pair. The reason for averaging $\mathcal{D}(\varphi_j, \varphi_k^{-1})$ and $\mathcal{D}(\varphi_j^{-1}, \varphi_k)$ choice is that it enforces the symmetry condition $\mathcal{A}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k) = \mathcal{A}_{\text{cmd}}(\boldsymbol{u}_k, \boldsymbol{u}_j)$.

V. Application to range-finder data

In this section, we apply the theory to sensorimotor cascades with range-finder data. We preprocess the 1D range-data to obtain a 2D population code representation (Fig. 3), which allows to treat range-data using exactly the same code as 2D images. We try the method on three dynamics: a differential-drive robot (where the commands are left/right track velocity), and, in simulation, with a unicycle dynamics (commanded in linear/angular velocity) and a car-like dynamics (commanded with steering angle and driving velocity). We show that the concept of commands distance/anti-distance allows to discover redundant commands and reversible commands pairs independently of the commands representation.

Processing pipeline: The pipeline that we use for processing the range-finder data is shown in Fig. 4. Starting from a planar scan (Fig. 4*a*), we consider the polar representation (Fig. 4*b*). Then, we transform the 1D signal into a 2D signal by using a population code representation (Fig. 4*c*); each reading y^i is assigned a row of cells, and each cell is assigned a center $c^{i,k}$. The activation level of each cell is a function of the distance between y^i and $c^{i,k}$. Denoting the 2D signal $Y^{i,k}$, we set $Y^{i,k} = f(|y^i - c^{i,k}|)$ where f is a small Gaussian kernel ($\sigma = 1\%$ of the range of y^i). Once we have the 2D signal, we forget its origin as a range-finder scan, and we treat it like any other image.

Fig. 4 shows also an example of prediction. Starting from the 2D signal Y_0 in Fig. 4*c,* and a learned diffeomorphism φ (represented here by Lena), we obtain the predicted signal Y_4 in Fig. 4*d* by applying 4 times the diffeomorphism φ (or, by first computing $\varphi' = \varphi \circ \varphi \circ \varphi \circ \varphi$, and then applying φ' to Y_0, which is the same, up to numerical errors). For visualizing the result, we can convert back to range readings (Fig. 4*e*) and range scan (Fig. 4f).

Dynamics considered: We consider three common mobile robot dynamics for wheeled mobile robots: unicycle (Fig. 5*a*), car-like (Fig. 6*b*), and differential-drive (Fig. 7*c*). All three dynamics have two commands: by appropriate normalization we can assume that $\boldsymbol{u} \in [-1, +1] \times [-1, +1]$ for all of them.

The unicycle and differential-drive have the same dynamics, but with different representations of the commands: the linear and angular velocity of the unicycle are linearly related to left/right wheel velocity for a differential drive. For the car-like dynamics, we assume that one command is the driving velocity, and the other is the instantaneous steering angle. The car-like dynamics is more restricted than the other two, as the vehicle cannot turn in place.

Learning data: As an example of a differential-drive robot, we use a Landroid[3] robot, with a Hokuyo [19] range-finder on board. The Hokuyo has a maximum range of 8m and an update frequency of 10Hz. The field of view of a Hokuyo is 270°, but the sensor is partially obstructed by the WiFi antennas. The learning data is taken in a cluttered lab environment, for a total of about 45 minutes. We use simulated data for the unicycle and the car-like, simulating a 360° range-finder with the same range as the Hokuyo. The simulated world is generated randomly from a collection of randomly placed polygons; the simulation is tuned to have approximately the same spatial statistics of the lab environment.

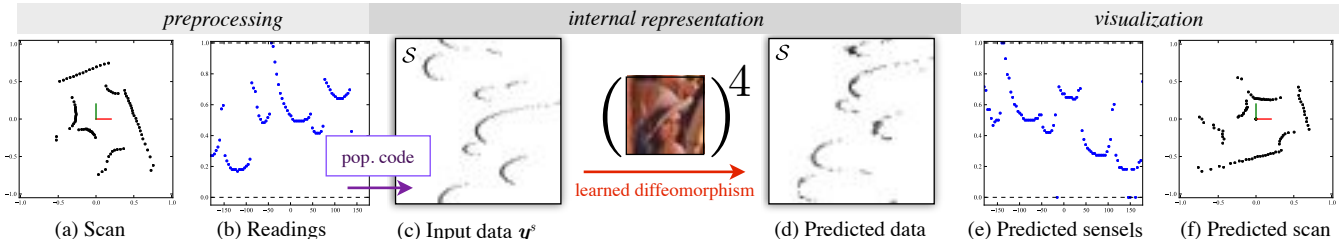
The commands alphabet is composed of the 9 canonical commands of the form (a, b), for $a, b \in \{-1, 0, 1\}$. The effect on the robot pose of choosing each canonical command is sketched in the grids in Fig. 5*b*, 6*b*, 7*b.*

Learned diffeomorphisms: The learned diffeomorphisms are shown in the grids in Fig. 5*c*, 6*c*, 7*c.* Here, the diffeomorphisms are visualized by their effect on the Lena template.

It turns out that learning diffeomorphisms of the population-code representation of range-finder data is more challenging than learning diffeomorphisms of RGB images, because the data is much sparser (see, e.g., Fig. 4*cd*). It was surprising to see that, of all the diffeomorphisms learned, the most noisy result is for the commands that do not move the robot (Fig. 6*c*, middle row), the reason being that the motion we are trying to recover is small (actually, zero) with respect to the sensor noise. This uncertainty is appropriately captured by the estimate of Γ (not shown).

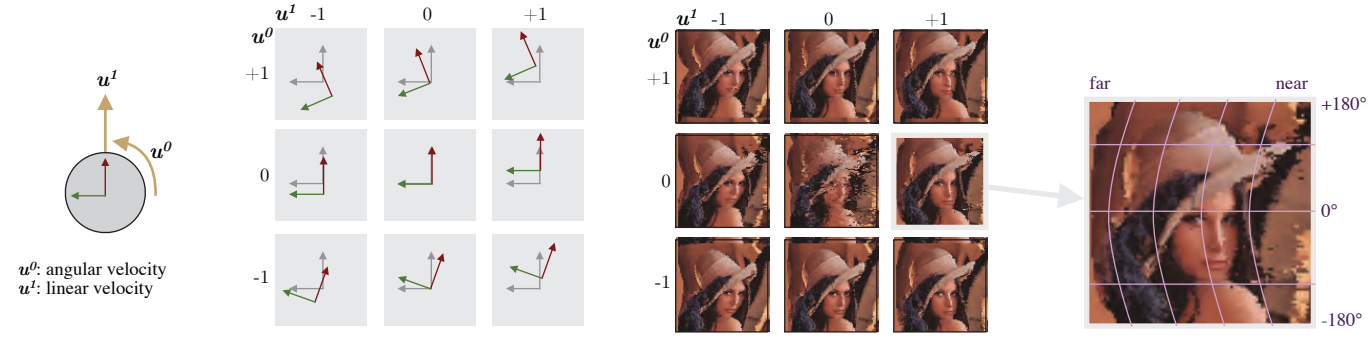
Learned command structure: Tables II, III, IV show the computed distance $\mathcal{D}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k)$ and anti-distance $\mathcal{A}_{\text{cmd}}(\boldsymbol{u}_j, \boldsymbol{u}_k)$ for all commands pairs for the three dynamics considered.

[3]The Landroid is a prototype produced by iRobot:
http://www.irobot.com/gi/research/Advanced_Platforms/LANdroids_Robot



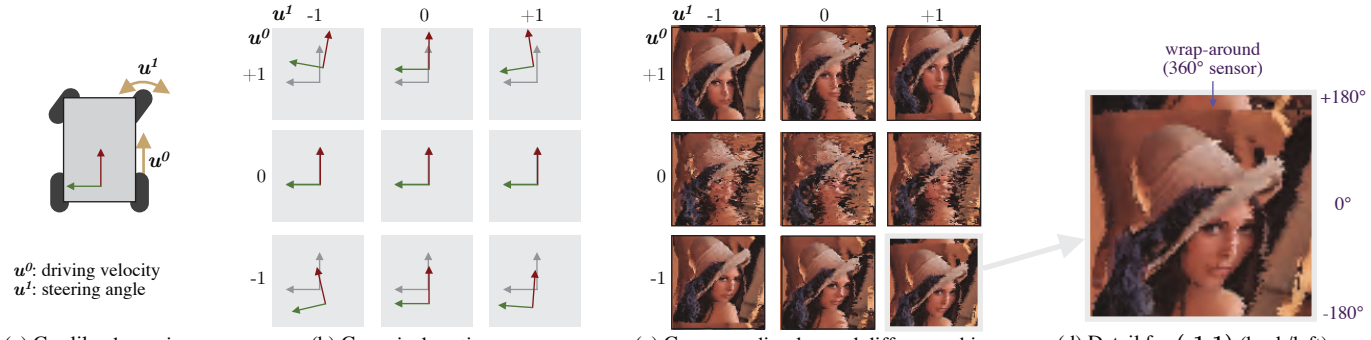
(a) Scan (b) Readings (c) Input data u^s (d) Predicted data (e) Predicted sensels (f) Predicted scan

Figure 4. This figure shows the pipeline that we use for range-finder data. Starting from the scan (subfigure *a*) we consider the raw range-readings (i.e., the polar representation of the scan). Then we use a population code to obtain a 2D image from the 1D data. Once we have the 2D data, we forget about its origin as a range-finder scan, and we use exactly the same code we used for images. Here we show an example of prediction: one learned diffeomorphism, here represented by Lena, is applied 4 times to the image in *c* to obtain the predicted image in *d*. For visualization purposes, from the 2D image we can go back to the range readings (subfigure *e*) and obtain the predicted scan (subfigure *f*), which shows that the learned diffeomorphism corresponded to a pure rotation.



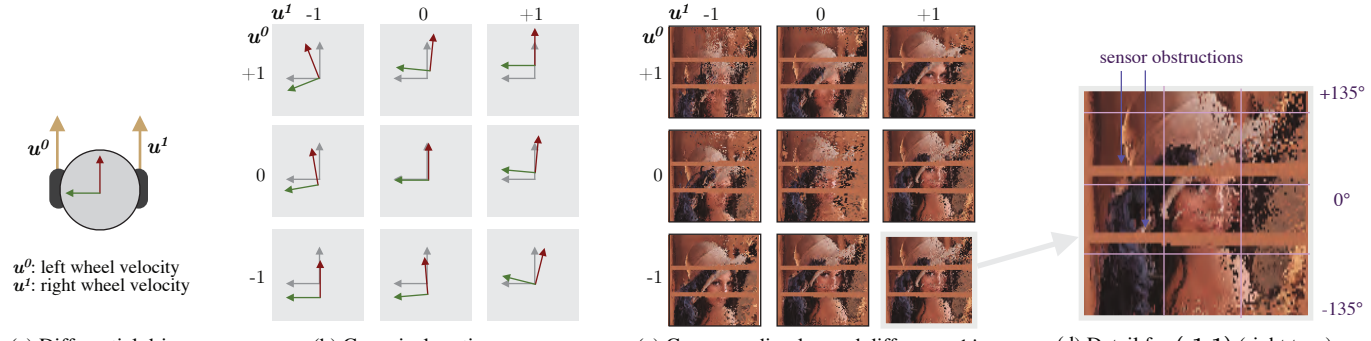
(a) Unicycle dynamics (b) Canonical motions (c) Corresponding learned diffeomorphisms (d) Detail for **(0,1)** (forward)

Figure 5. In this series of figures, subfigure *a* illustrates the robot dynamics; in this case, a unicycle dynamics. Subfigure *b* shows the effect on the robot pose of the 9 canonical commands. The gray arrows denote the initial pose, and the red/green arrows (for the x and y direction, respectively) show the robot final pose after applying the motion. Subfigure *c* shows the effect of the commands on the population code representation for a range-finder scan mounted on the robot (see Fig. 4 for an explanation of the preprocessing to obtain a 2D image from a 1D scan).



(a) Car-like dynamics (b) Canonical motions (c) Corresponding learned diffeomorphisms (d) Detail for **(-1,1)** (back/left)

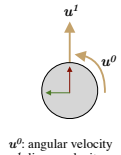
Figure 6. A car-like dynamics, commanded in driving velocity and instantaneous steering angle, is more restricted than a unicycle as the robot cannot turn in place. Note that the three commands $(0,-1)$, $(0,0)$, $(0,+1)$ are equivalent and correspond to the robot staying in place.



(a) Differential drive (b) Canonical motions (c) Corresponding learned diffeomorphisms (d) Detail for **(-1,1)** (right turn)

Figure 7. A differential-drive dynamics is the same as a unicycle dynamics following a change of representation for the commands; compare subfigure *b* with Fig. 5b. The data we use for the differential drive comes from a real robot, mounting a Hokuyo range-finder with a 270° field of view. There are two antennas in front obstructing the range-finder, therefore the learned diffeomorphisms have two missing stripes.

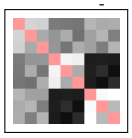
Table II
DISTANCE AND ANTI-DISTANCE MATRICES FOR UNICYCLE DYNAMICS.



u^0: angular velocity
u^1: linear velocity

(A) DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	0.68	0.70	1.19	1.21	1.23	0.86	0.92	0.91
(0,1)		-	0.93	1.04	0.92	1.25	0.97	0.86	1.17
(0,-1)			-	1.04	1.23	0.91	0.98	1.18	0.87
(1,0)				-	0.50	0.53	1.74	1.73	1.75
(1,1)					-	0.92	1.73	1.82	1.66
(1,-1)						-	1.73	1.82	1.66
(-1,0)							-	0.46	0.44
(-1,1)								-	0.84
(-1,-1)									-



(B) ANTI-DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	0.85	0.83	0.97	1.03	1.02	1.34	1.38	1.38
(0,1)		-	**0.36**	1.07	1.26	0.95	1.17	1.35	1.08
(0,-1)			-	1.07	0.95	1.26	1.15	1.06	1.33
(1,0)				-	1.81	1.81	**0.21**	0.58	0.57
(1,1)					-	1.71	0.58	0.97	**0.28**
(1,-1)						-	0.57	**0.30**	0.95
(-1,0)							-	1.90	1.91
(-1,1)								-	1.84
(-1,-1)									-

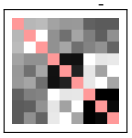
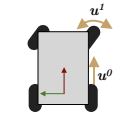


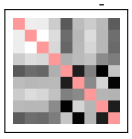
Table III
DISTANCE AND ANTI-DISTANCE MATRICES FOR CAR-LIKE DYNAMICS.



u^0: driving velocity
u^1: steering angle

(A) DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	**0.65**	**0.66**	0.85	1.51	0.94	0.85	0.96	1.49
(0,1)		-	**0.56**	0.75	1.40	0.97	0.76	0.97	1.39
(0,-1)			-	0.78	1.43	0.96	0.79	0.97	1.41
(1,0)				-	1.02	0.96	0.91	1.22	1.30
(1,1)					-	1.85	1.29	2.01	0.89
(1,-1)						-	1.23	0.80	2.00
(-1,0)							-	0.98	0.99
(-1,1)								-	1.85
(-1,-1)									-



(B) ANTI-DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	1.20	1.22	1.05	1.04	1.66	1.01	1.67	1.03
(0,1)		-	1.10	0.95	1.07	1.56	0.91	1.57	1.07
(0,-1)			-	0.98	1.07	1.58	0.93	1.59	1.03
(1,0)				-	1.29	1.40	**0.41**	1.19	1.03
(1,1)					-	0.90	1.04	**0.25**	1.91
(1,-1)						-	1.14	2.02	**0.23**
(-1,0)							-	1.38	1.30
(-1,1)								-	0.91
(-1,-1)									-

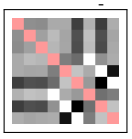
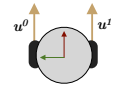


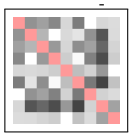
Table IV
DISTANCE AND ANTI-DISTANCE MATRIX FOR LANDROID (DIFFERENTIAL DRIVE).



u^0: left wheel velocity
u^1: right wheel velocity

(A) DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	1.34	1.25	1.41	0.89	1.49	0.52	0.89	0.86
(0,1)		-	1.12	0.55	0.88	0.81	1.34	1.74	0.87
(0,-1)			-	1.29	0.90	1.37	1.24	1.62	0.96
(1,0)				-	0.95	0.67	1.45	1.79	0.96
(1,1)					-	1.06	0.92	1.30	0.54
(1,-1)						-	1.51	1.86	1.03
(-1,0)							-	0.92	0.87
(-1,1)								-	1.24
(-1,-1)									-



(B) ANTI-DISTANCE BETWEEN COMMANDS

	(0,0)	(0,1)	(0,-1)	(1,0)	(1,1)	(1,-1)	(-1,0)	(-1,1)	(-1,-1)
(0,0)	-	1.08	1.46	1.07	1.32	1.33	1.71	1.98	1.26
(0,1)		-	1.76	2.08	1.49	2.21	1.14	1.39	1.46
(0,-1)			-	1.80	1.36	1.92	1.51	1.73	1.33
(1,0)				-	1.56	2.29	**1.06**	1.33	1.51
(1,1)					-	1.70	1.35	1.63	**0.95**
(1,-1)						-	1.36	1.26	1.65
(-1,0)							-	2.01	1.33
(-1,1)								-	1.60
(-1,-1)									-

For example, for the unicycle the reversible pairs are $\langle(-1,-1),(1,1)\rangle$, $\langle(1,0),(-1,0)\rangle$, $\langle(0,+1),(0,-1)\rangle$, $\langle(-1,1),(1,-1)\rangle$, and these are given the smaller values of anti-distance in Table IIIb. The unicycle commands have a native linear structure, so reversible pairs are of the form $\langle(a,b),(-a,-b)\rangle$.

The car-like dynamics has the three null and redundant commands: $(0,0)$, $(0,1)$, and $(0,-1)$; these correspond to the driving velocity set to zero, and their corresponding diffeomorphism is the identity. The detected reversible pairs are $\langle(1,0),(-1,0)\rangle$, $\langle(1,1),(-1,1)\rangle$, $\langle(1,-1),(-1,-1)\rangle$; these are of the form $\langle(a,b),(-a,b)\rangle$, which shows that the dynamics of the car-like is not linear in the original representation. Yet, we are able infer the linear structure from the analysis of the learned diffeomorphisms.

For the differential-drive dynamics, learned with real data, the pairs with the two lowest anti-distance are $\langle(1,0),(-1,0)\rangle$ and $\langle(1,1),(-1,-1)\rangle$. Then, there are a few false matches which have lower anti-distance than the other two pairs of reversible commands ($\langle(0,1),(0,-1)\rangle$ and $\langle(-1,1),(1,-1)\rangle$. This is probably due to the fact that computing the inverse of a diffeomorphism is very sensitive to noise, and currently we do not take into account the estimated diffeomorphism uncertainty, which is very large in this case, due to the limited field of view, and the antennas occlusions.

VI. DISCUSSION

Invariance analysis: Fig. 1 shows the *representation nuisances* $\mathcal{G}^u$ and $\mathcal{G}^y$ acting on the commands and the observations, respectively. These are to be interpreted as static (i.e., fixed in time) invertible transformations that act on the signals, by changing their representation but not their informative content. Representation nuisances are a technical device that allows to characterize the hidden assumptions of agents by their invariance properties [18]. In principle, a generic bootstrapping agent should not care about the representation of the data, however, in practice, one has to choose a particular class of models, and this choice often implies several hidden assumptions about the data. For example, we noticed in the introduction that BDS/BGDS models are not robust to a nonlinear reparametrization of $\boldsymbol{u}$: if the dynamics is linear in $\boldsymbol{u}$, it cannot be affine also in $f(\boldsymbol{u})$, for f a generic nonlinear transformation. This non-invariance can be interpreted as a bias of the class of models towards capturing only a certain representation of a given system. It will likely never be practical to obtain agents which are invariant to all possible representation nuisances; however, it is important to do this invariance analysis to understand the limits of a proposed class of models and to measure progress towards solving the general problem.

For the models used in this paper, we notice the following invariance properties. As for the commands, the method is invariant to any reparametrization of the kind $\boldsymbol{u}' = f(\boldsymbol{u})$, as we only use the commands values as labels.

As for the observations, we notice that the DDS class of models is closed with respect to diffeomorphisms of $\mathcal{S}$, in the sense that, if the dynamics of the observations y^s can be represented by a DDS, then also the dynamics of $z^s = y^{\alpha(s)}$, for any $\alpha \in \mathrm{Diff}(\mathcal{S})$ can be represented as a DDS. This implies several invariance properties with respect to reparametrization

of the original data. For example, let $\rho(\theta)$ be the range-finder reading as a function of the direction θ. Suppose that, instead of the range-finder readings $\rho(\theta)$, the sensor provided the readings $\tilde{\rho}(\theta) = g(\rho(\theta))$, for any nonlinear invertible function $g : \mathbb{R}^+ \to \mathbb{R}^+$. It is easy to see that, once the readings are transformed to the 2D representation of Fig. 4*c*, the effect of g would simply be a diffeomorphism of the 2D domain. Therefore, we conclude that the method is robust to a (continuous) change of representation of the original readings. As another example, in the case of the camera, robustness to a diffeomorphism α means that the method is not dependent on knowing the precise camera calibration (i.e., the direction of each pixel on the visual sphere), as the unknown camera calibration can be thought as a unknown arbitrary diffeomorphism from the domain $\mathcal{S}$ (the camera frame) to the visual sphere $\mathbb{S}^2$.

Unfortunately, the distance (3) that we used so far is not invariant to diffeomorphisms, in the sense that $\mathcal{D}(\alpha \circ \varphi_1, \alpha \circ \varphi_2) \neq \mathcal{D}(\varphi_1, \varphi_2)$. This means that the thresholds to commands distances and anti-distances to decide if commands pairs are redundant or reversible pairs would have to be re-tuned if the parametrization of $\mathcal{S}$ change.

Relation to discrete diffeomorphisms: The discretization of diffeomorphisms that we used in this paper is intuitive but does not conserve certain important properties. *Discrete geometry* is a discipline concerned with the discretization of objects in differential geometry to a discrete domain, and finds applications in area such as fluid mechanics and computer graphics. Gawlik *et. al.* [20] describe the *discrete diffeomorphism group*: if a certain manifold is approximated with a simplicial complex of n cells, discrete diffeomorphisms are represented as a certain subfamily of $n \times n$ stochastic matrices, in a way such that properties of continuous diffeomorphisms are preserved in the discretized version. In practice, using stochastic matrices allows each cell to correspond to multiple cells, even without considering uncertainty. Using such representation would probably improve the accuracy of the representation, but the possible accuracy gains are to be weighted with the increased computational complexity (from $\mathcal{O}(n)$ of the current method to $\mathcal{O}(n^2)$ of the discrete diffeomorphism group, even without considering uncertainty).

VII. Conclusions and future work

In this paper, we have described a new candidate model for describing the dynamics of robotic sensorimotor cascade. With respect to previous work (BDS/BGDS models), it improves on several fronts: it is less sensitive to instantaneous sensor noise, it does not rely on a linear structure for the commands (but can recover it if it is present), it allows to represent uncertainties due to occlusions or limited field of view, it allows long-term prediction, and its representation allows the compressibility of sequence of commands into one supercommand. However, it is much more expensive to learn.

There are a number of possible directions for future work. In this paper we have assumed that the alphabet $\mathcal{U} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{|\mathcal{U}|}\}$ is given. However, in practice, one has that the commands naturally live in a continuous domain; for example $\mathcal{U} = [-1, +1]^{|\boldsymbol{u}|}$. An open problem is designing an agent that starts from the continuous domain, and automatically chooses a discrete set of actions that are the best representation of the dynamics given finite computational resources and learning time. This appears to be relevant for planning problems (servoing, exploration, etc.) where the dynamics of the platform is abstracted away using this representation, and planning is done completely in observation space. The estimation problem can be improved as well: for now, each cell of the diffeomorphism is estimated separately from the others but introducing some kind of regularization would help. However, it is unclear how to do this in a way which is invariant to reparametrization of the domain, and that can tolerate the noise introduced by faulty sensels.

Acknowledgments. We are grateful to Larry Matthies, Thomas Werne, Marco Pavone at JPL for lending the Landroid platform and assisting with the software development.

References

[1] Cohen *et al.*, "Functional relevance of cross-modal plasticity in blind humans," *Nature*, vol. 389, no. 6647, 1997. DOI.

[2] O. Collignon, P. Voss, M. Lassonde, and F. Lepore, "Cross-modal plasticity for the spatial processing of sounds in visually deprived subjects.," *Experimental brain research*, vol. 192, no. 3, 2009. DOI.

[3] A. Censi, M. Hakansson, and R. M. Murray, "Fault detection and isolation from uninterpreted data in robotic sensorimotor cascades," 2011. Technical report. (link).

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[5] B. Kuipers, "An intellectual history of the Spatial Semantic Hierarchy," *Robotics and cognitive approaches to spatial mapping*, vol. 38, 2008.

[6] D. Pierce and B. Kuipers, "Map learning with uninterpreted sensors and effectors," *Artificial Intelligence*, vol. 92, no. 1-2, 1997. DOI.

[7] J. Stober, L. Fishgold, and B. Kuipers, "Sensor map discovery for developing robots," in *AAAI Fall Symposium on Manifold Learning and Its Applications*, 2009. (link).

[8] J. Modayil, "Discovering sensor space: Constructing spatial embeddings that explain sensor correlations," in *Proceedings of the International Conference on Development and Learning (ICDL)*, 2010. DOI.

[9] J. Stober and B. Kuipers, "From pixels to policies: A bootstrapping agent," in *Proceedings of the International Conference on Development and Learning (ICDL)*, 2008. DOI.

[10] J. Stober, L. Fishgold, and B. Kuipers, "Learning the sensorimotor structure of the foveated retina," in *Proceedings of the International Conference on Epigenetic Robotics (EpiRob)*, 2009. (link).

[11] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, 2010. DOI.

[12] J. Provost and B. Kuipers, "Self-organizing distinctive state abstraction using options," in *Proceedings of the International Conference on Epigenetic Robotics (EpiRob)*, 2007. (link).

[13] J. Modayil and B. J. Kuipers, "The initial development of object knowledge by a learning robot," *Robotics and Autonomous Systems*, vol. 56, no. 11, 2008. DOI.

[14] J. Stober, R. Miikkulainen, and B. Kuipers, "Learning geometry from sensorimotor experience," in *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2011.

[15] B. Boots and G. J. Gordon, "Predictive state temporal difference learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2011. (link).

[16] A. Censi and R. M. Murray, "Bootstrapping bilinear models of robotic sensorimotor cascades," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011. (link).

[17] A. Censi and R. M. Murray, "Bootstrapping sensorimotor cascades: a group-theoretic perspective," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011. (link).

[18] A. Censi and R. M. Murray, "Uncertain semantics, representation nuisances, and necessary invariance properties of bootstrapping agents," in *Joint IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2011.

[19] L. Kneip, F. T. G. Caprari, and R. Siegwart, "Characterization of the compact hokuyo URG-04LX 2d laser range scanner," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Kobe, Japan), 2009.

[20] E. Gawlik, P. Mullen, D. Pavlov, J. E. Marsden, and M. Desbrun., "Geometric, variational discretization of continuum theories," *Physica D: Nonlinear Phenomena*, 2011. To appear. (link).