2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Accurate recursive learning of uncertain diffeomorphism dynamics

Adam Nilsson Andrea Censi

Abstract— **Diffeomorphisms dynamical systems are dynamical systems for which the state is an image and each command induce a diffeomorphism of the state. These systems can approximate the dynamics of robotic sensorimotor cascades well enough to be used for problems such as planning in observations space. Learning of an arbitrary diffeomorphism from pairs of images is an extremely high dimensional problem. This paper describes two improvements to the methods presented in previous work. The previous method had required $\mathcal{O}(\rho^4)$ memory as a function of the desired resolution ρ, which, in practice, was the main limitation to the resolution of the diffeomorphisms that could be learned. This paper describes an algorithm based on recursive refinement that lowers the memory requirement to $\mathcal{O}(\rho^2)$. Another improvement regards the estimation the diffeomorphism uncertainty, which is used to represent the sensor's limited field of view; the improved method obtains a more accurate estimation of the uncertainty by checking the consistency of a learned diffeomorphism and its independently learned inverse. The methods are tested on two robotic systems (a pan-tilt camera and a 5-DOF manipulator).**

I. Introduction

Robotic systems are going to be truly safe and reliable only when they have the ability of verifying all their models and their assumptions about themselves and the environment [1]. Clearly, the fewer assumptions an agent has, the fewer assumptions can be violated. The problem of bootstrapping [2] considers the limit as the assumptions tend to zero: a bootstrapping agent assumes nothing except that it is embodied in a robotic body. We think of a bootstrapping agent as able to create increasingly more complex representation starting from uninterpreted observations and commands [3, 4]. For example, a bootstrapping agent can start from scrambled pixels and reconstruct the shape and calibration of its sensors [2, 5–7]. At a higher level, the agent should learn the dynamics of the unknown commands–observations black box. A robotic system can be idealized as the series of actuators and sensors: commands to the actuators produce motion, and motion makes the sensors observations change. Learning the actuator dynamics (how commands produce motion) is relatively easy because in a robotic system commands and states are low dimensional, thus established generic learning techniques can be used (e.g., [8]). Instead, learning the nonlinear and high-dimensional observations dynamics (how observations change following motions) requires methods specific to robotics.

Previous work has shown that it is possible to find families of models that can represent the dynamics of the set of "canonical" robotic sensors (range-finders, cameras, and field-samplers) [9]. These classes occupy different points in the trade-off of efficiency *vs* assumptions/prior information. We have shown classes of models that learn generic bilinear relations among commands, observations, and their derivatives [10, 11], and their use in applications such as instantaneous servoing and fault/anomaly detection [12]. The most structured class of models represents the sensorimotor dynamics as a set of diffeomorphisms of the observations space [13]. Successively, we have used these models for planning in observations space [14], even dealing with a sensor's limited field of view. This line of work is related to deep learning approaches to unsupervised learning of transformations [15, 16], but the focus is on dynamics and control rather than classification tasks.

Contribution: This paper presents two improvements to previous work. The first improvement is in the learning of diffeomorphisms. In practice, the naive learning method described in [13] was limited by the memory required, which was $\mathcal{O}(\rho^4)$ as a function of the resolution ρ. This paper describes a method based on recursive refinement that is more efficient, requiring $\mathcal{O}(\rho^2)$ storage and $\mathcal{O}(\rho^2 \log \rho)$ computation, thus allowing learning of higher-resolution diffeomorphisms. Moreover, the paper describes an improvement to the method for computing the scalar uncertainty field associated to the learned diffeomorphism. This makes it possible to have a better classification of areas that can be predicted or not due to the sensor's limited field of view.

Paper outline: Section II recalls some differential geometry preliminaries. Section III recalls the class of Diffeomorphism Dynamical Systems (DDS): these are discrete-time dynamical systems for which the state $\boldsymbol{x}$ is a function on a manifold $\mathcal{S}$, the observations are the part of the state visible through a viewport $\mathcal{V} \subset \mathcal{S}$, and each command $\boldsymbol{u} \in \mathcal{U}$ induces a diffeomorphism $\Phi(\boldsymbol{u}) \in \mathsf{Diff}(\mathcal{S})$ of the state $\boldsymbol{x}$. In the case of a camera, the manifold $\mathcal{S}$ is the visual sphere $\mathbb{S}^2$ and a point $s \in \mathcal{S}$ corresponds to a pixel.

Section IV shows how we can learn a DDS by learning the diffeomorphism $\varphi = \Phi(\boldsymbol{u})$ associated to each command $\boldsymbol{u}$. To recover the map $\varphi : \mathcal{S} \to \mathcal{S}$, we learn independently the values of $\varphi(s)$, by looking for the pixel v that minimizes a dissimilarity statistics. The method presented in previous work did this procedure naively, leading to a memory requirement which was quartic in the resolution ρ. The improved method achieves a better complexity by recursively refining the solution rather than using a fixed search resolution.

Section V discusses how to represent and learn a scalar-valued uncertainty field. The method presented in previous work looked at the residuals of the minimized dissimilarity. The improvement shown here is based on checking the consistency of the estimated φ and its inverse φ^{-1}.

Section VI describes experiments with two robotic platforms. Section VII concludes and discusses future work. Datasets and source code can be found at http://purl.org/censi/2013/rddl.

A. Censi and A. Nilsson are with the Computing & Mathematical Sciences department, Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA. E-mail: {andrea, adam}@cds.caltech.edu.
Research supported by the US National Science Foundation (NRI program, grant #12018687) and DARPA (MSEE program, grant #FA8650-11-1-7156).

II. Geometry preliminaries

This section recalls basic preliminaries about manifolds, functions defined on manifolds, and diffeomorphisms. Do Carmo [17] is a readable introduction to differential geometry.

Manifolds: Let $\mathcal{S}$ be a differentiable manifold, which will represents the domain in which the sensor observations are defined. For a central camera, the manifold $\mathcal{S}$ is the unit sphere $\mathbb{S}^2$. For range-finders, we can use a construction in which $\mathcal{S} = \mathbb{S}^1 \times \mathbb{R}$ (directions $\times$ distances) [11]. A manifold comes equipped with a distance function $d^{\mathcal{S}} : \mathcal{S}\times\mathcal{S} \to \mathbb{R}^+$, so that $d^{\mathcal{S}}(s_1, s_2)$ is the distance between two points $s_1, s_2 \in \mathcal{S}$.

Images: We call *image* a function from $\mathcal{S}$ to some output space O. For an RGB camera, the output space is $O = [0, 1]^3$. For simplicity, we assume that $O = \mathbb{R}$, but everything easily generalizes to more complex output spaces.

$\mathsf{Im}(\mathcal{S})$ is the set of all images defined on $\mathcal{S}$. The L_p distance $d^{\mathsf{Im}}_{L_p}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ between two images $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathsf{Im}(\mathcal{S})$ is the average of their difference over the domain $\mathcal{S}$:

$$d^{\mathsf{Im}}_{L_p}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \tfrac{1}{|\mathcal{S}|}\big(\int_{\mathcal{S}} |y_1(s) - y_2(s)|^p \,\mathrm{d}s\big)^{\frac{1}{p}}. \qquad (1)$$

There exist also other distances between images that have better performance as heuristics in a planning problem [14].

Diffeomorphisms: A *diffeomorphism* of $\mathcal{S}$ is a map from $\mathcal{S}$ to itself that is differentiable, invertible, and whose inverse is differentiable as well. $\mathsf{Diff}(\mathcal{S})$ is the set of all diffeomorphisms of $\mathcal{S}$. A distance between two diffeomorphisms can be defined as the average distance of the transformed points:

$$d^{\mathsf{Diff}(\mathcal{S})}(\varphi_1, \varphi_2) = \int_{\mathcal{S}} d^{\mathcal{S}}(\varphi_1(s), \varphi_2(s))\,\mathrm{d}s.$$

A norm on $\mathsf{Diff}(\mathcal{S})$ can be defined as the distance of a diffeomorphism from the identity function Id:

$$\|\varphi\| = d^{\mathsf{Diff}(\mathcal{S})}(\varphi, \mathsf{Id}).$$

III. Diffeomorphism dynamics

A *diffeomorphism dynamical systems* (DDS) [13] is a discrete-time dynamical system for which the input commands $\boldsymbol{u}$ belong to a finite alphabet $\mathcal{U}$; the state $\boldsymbol{x} \in \mathsf{Im}(\mathcal{S})$ is an image on $\mathcal{S}$; the observations $\boldsymbol{y}$ are the subset of the state visible through the "viewport" $\mathcal{V} \subset \mathcal{S}$.

Dynamics: The state $\boldsymbol{x} \in \mathsf{Im}(\mathcal{S})$ evolves according to a diffeomorphism that depends on which command is chosen. Let $\Phi : \mathcal{U} \to \mathsf{Diff}(\mathcal{S})$ be the function that associates to each command a diffeomorphism of the manifold $\mathcal{S}$, so that each command $\boldsymbol{u} \in \mathcal{U}$ is associated to a diffeomorphism $\Phi(\boldsymbol{u}) \in \mathsf{Diff}(\mathcal{S})$. The state $\boldsymbol{x} \in \mathsf{Im}(\mathcal{S})$ evolves according to the dynamics

$$x_{k+1}(s) \quad = \quad x_k(\varphi_k(s)), \qquad \text{with } \varphi_k = \Phi(\boldsymbol{u}_k).$$

This can be written in a more compact form as

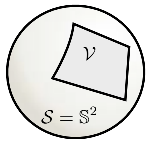
$$\underbrace{\boldsymbol{x}_{k+1}}_{\text{next image}} = \underbrace{\boldsymbol{x}_k}_{\text{previous image}} \circ \underbrace{\Phi(\boldsymbol{u}_k).}_{\text{diffeomorphism } \varphi_k} \qquad (2)$$

TABLE I

Symbols used in this paper

Manifolds, images, and diffeomorphisms	
$\mathcal{S}$	a differentiable manifold
$d^{\mathcal{S}}(s_1, s_2)$	the distance on the manifold $\mathcal{S}$
$\mathsf{Im}(\mathcal{S})$	the set of "images" (functions from $\mathcal{S}$ to $\mathbb{R}$)
$d^{\mathsf{Im}}_{L_p}(\boldsymbol{y}_1, \boldsymbol{y}_2)$	L_p distance between two images $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathsf{Im}(\mathcal{S})$
$\mathsf{Diff}(\mathcal{S})$	the set of all diffeomorphisms of $\mathcal{S}$
$\varphi \in \mathsf{Diff}(\mathcal{S})$	a generic diffeomorphism of $\mathcal{S}$
$d^{\mathsf{Diff}(\mathcal{S})}$	distance between diffeomorphisms
$\|\varphi\|$	norm of a diffeomorphism
Diffeomorphism dynamics systems (DDS)	
$\mathcal{U}$	finite commands alphabet
$\boldsymbol{u}_k \in \mathcal{U}$	command given at time k
$\boldsymbol{x}_k \in \mathsf{Im}(\mathcal{S})$	state of the DDS
$\mathcal{V} \subset \mathcal{S}$	viewport
$\boldsymbol{y}_k \in \mathsf{Im}(\mathcal{V})$	observations
$\Phi : \mathcal{U} \to \mathsf{Diff}(\mathcal{S})$	function that associates the diffeomorphism for each command
$\mathcal{V}^{\varphi}_{\text{unpr}}, \mathcal{V}^{\varphi}_{\text{pred}} \subset \mathcal{V}$	unpredictable/predictable regions
Estimation of diffeomorphisms	
$\delta > 0$	bound on the size of $\|\varphi\|$ as a fraction of $\mathcal{V}$
$\mathbb{S} = \{s^i\}_{i=1}^n$	discretized domain
A_s	search area for a point s
$E_s(v) : A_s \to \mathbb{R}^+$	dissimilarity function
ρ	native image resolution
g	resolution-independent sampling grid
$\rho^\star$	desired resolution for the solution
(H, W)	grid width/height (pixels)
(S, T)	image dimensions on $\mathcal{S}$ (deg)
$\lambda \in (0, 1)$	shrinking factor for RDDL
Uncertainty and uncertainty estimation	
$\mathsf{UIm}(\mathcal{S})$	set of "uncertain images"
$\langle \boldsymbol{y}, \boldsymbol{z}\rangle \in \mathsf{UIm}(\mathcal{S})$	generic uncertain image
$\boldsymbol{z} : \mathcal{S} \to [0, 1]$	scalar uncertainty
$\mathsf{UDiff}(\mathcal{S})$	set of "uncertain diffeomorphism"
$(\varphi, \gamma) \in \mathsf{UDiff}(\mathcal{S})$	generic uncertain diffeomorphism
$\gamma : \mathcal{S} \to [0, 1]$	scalar uncertainty
$\hat{\gamma}_{\text{res}}(s)$	uncertainty estimated using residuals method (Section V-B)
$\hat{\gamma}_{\text{c,a}}(s)$	uncertainty estimated using consistency condition, absolute... (Section V-C)
$\hat{\gamma}_{\text{c,n}}(s)$	...and normalized

Observations: The observations $\boldsymbol{y}_k \in \mathsf{Im}(\mathcal{V})$ are a function defined on the viewport $\mathcal{V} \subset \mathcal{S}$. Alternatively, it is useful sometimes to define $\boldsymbol{y}_k \in \mathsf{Im}(\mathcal{S})$ and set the observations to have value 0 outside $\mathcal{V}$:

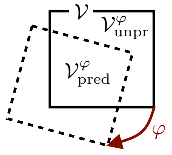


$$\boldsymbol{y}_k(s) = \begin{cases} \boldsymbol{x}_k(s) + \epsilon_k(s) & s \in \mathcal{V}, \\ 0 & s \in \mathcal{S} \setminus \mathcal{V}. \end{cases} \qquad (3)$$

The process $\epsilon_k(s)$ is IID Gaussian noise on the measurements.

Predictable and unpredictable regions: For a given diffeomorphism φ, the viewport $\mathcal{V}$ can be partitioned in two regions: $\mathcal{V}^{\varphi}_{\text{pred}}$ is the region in which the values of the next observations $\boldsymbol{y}_{k+1}$ can be predicted from the previous observations $\boldsymbol{y}_k$, and the region $\mathcal{V}^{\varphi}_{\text{unpr}}$ is its complement:

$$\begin{aligned} \mathcal{V}^{\varphi}_{\text{pred}} &= \{s \in \mathcal{V} \mid \varphi(s) \in \mathcal{V}\}, \\ \mathcal{V}^{\varphi}_{\text{unpr}} &= \{s \in \mathcal{V} \mid \varphi(s) \in \mathcal{S} \setminus \mathcal{V}\}. \end{aligned}$$


$\mathcal{V}$
$\mathcal{V}^{\varphi}_{\text{unpr}}$
$\mathcal{V}^{\varphi}_{\text{pred}}$
φ

IV. Methods for estimating diffeomorphisms

This section describes the previous diffeomorphism estimation algorithm described in previous work [14] and an improved method that achieves better efficiency and sub-pixel accuracy by using recursive refinement.

Learning scenario: The algorithms share most assumptions about the learning scenario. The training data is a discrete-time stream of observations/commands tuples $\{\langle \boldsymbol{y}_k, \ \boldsymbol{u}_k \rangle\}_k$.

There is a clear trade-off in choosing the discretization interval Δ: estimating large motions gives more accurate results, as the "signal" is larger than the noise, but it is more computationally expensive because the search area for each pixel is larger. In principle, we could learn multi-scale transformations or use a different interval for each command. In this paper, the interval is manually chosen to be $\Delta = 1\,\text{s}$.

From the stream $\{\langle \boldsymbol{y}_k, \ \boldsymbol{u}_k \rangle\}_k$. we assemble tuples of the kind $\{\langle \boldsymbol{y}_k, \ \boldsymbol{u}_k, \ \boldsymbol{y}_{k+1} \rangle\}_k$ that are used as training data. Some examples are shown in Fig. 1. The diffeomorphism corresponding to each command $\boldsymbol{u} \in \mathcal{U}$ can be recovered separately. The temporal index k is not important, because learning only considers one pair of images at a time. Consequently, for the purpose of estimating one diffeomorphism, we can assume to have a set of image pairs $\{\langle \boldsymbol{y}_\text{a}, \boldsymbol{y}_\text{b} \rangle\}$ that are related by the diffeomorphism to be estimated.

Graph-search approaches to motion planning [14] also require to estimate the inverse diffeomorphism φ^{-1}. While it is possible to estimate φ and then invert it numerically, it is simpler and more robust to estimate φ^{-1} directly by using the same procedure with the images swapped: from the sequence swapping the role of two images, thus looking at the stream $\{\langle \boldsymbol{y}_\text{b}, \boldsymbol{y}_\text{a} \rangle\}$ instead of $\{\langle \boldsymbol{y}_\text{a}, \boldsymbol{y}_\text{b} \rangle\}$.

A. Discretized representation of diffeomorphism

It is convenient to think of the transformation to learn as a diffeomorphism of a continuous domain, but in practice we need to have a discretized representation, in which the diffeomorphism φ is represented by its values at a discrete set of points in the domains. More formally, the viewport $\mathcal{V} \subset \mathcal{S}$ is discretized into a set of n points $\mathbb{S} = \{s^i\}_{i=1}^n \subset \mathcal{V}$ arranged in a grid overlaid on $\mathcal{V}$.

In previous work, for each point $s^i \in \mathbb{S}$ we estimated the closest point to $\varphi(s^i)$ contained in $\mathbb{S}$; this is equivalent to learn a *permutation* of the points of $\mathbb{S}$. Hence a diffeomorphism on a 2D domain is represented as an integer tensor `D` of dimensions $H \times W \times 2$, where $H \times W$ is the shape of the image. If a point $s \in \mathbb{S}$ has coordinates `(u,v)` in the grid, then the coordinates of $\varphi(s)$ are `(D[u,v,0],D[u,v,1])`.

In this paper, the representation changes only slightly to accommodate sub-pixel resolution. The tensor D is a 2D array of floating-point values, such that it is possible to interpolate between coordinates.

B. Pointwise Discretized Diffeomorphism Learning (PDDL)

The diffeomorphism estimation method described in previous work, here dubbed "PDDL", which stands for *Pointwise Discretized Diffeomorphism Learning,* estimates the diffeomorphism pointwise.

Let us first describe the procedure assuming a smooth domain. For each point $s \in \mathcal{V}$ we need to find the corresponding $\varphi(s)$. The estimation is done for each pixel separately,

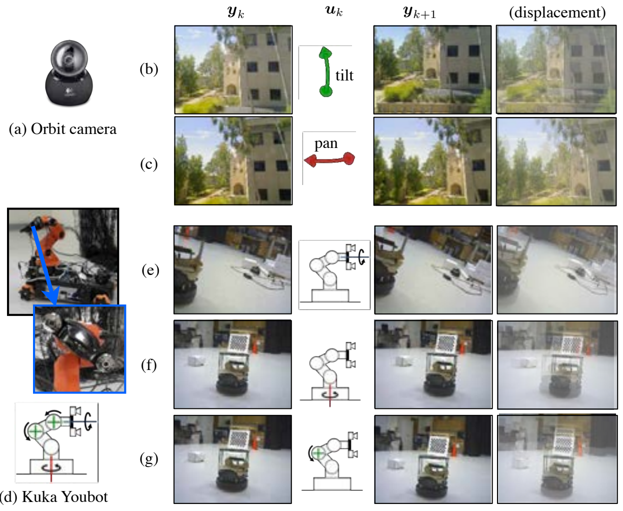


Fig. 1. Robotic platforms used in the experiments and example training data. The first column shows the initial image; the second is a representation of the command; the third shows the final image; the fourth is a superimposition of the two. (a) A pan-tilt Orbit camera. (b) Tilt motion; (c) Pan motion; (d) Kuka *Youbot* with a camera mounted on the end-effector. The arm has 5 degrees of freedom: the rotation of the base, three revolute joints in the same plane, and a final wrist, plus the gripper. (e) A wrist rotation results in an off-centered rotation of the image; (f) Base rotation is equivalent to a pan motion (but this depends on the position of the wrist); (g) The third revolute joint results in a tilt motion.

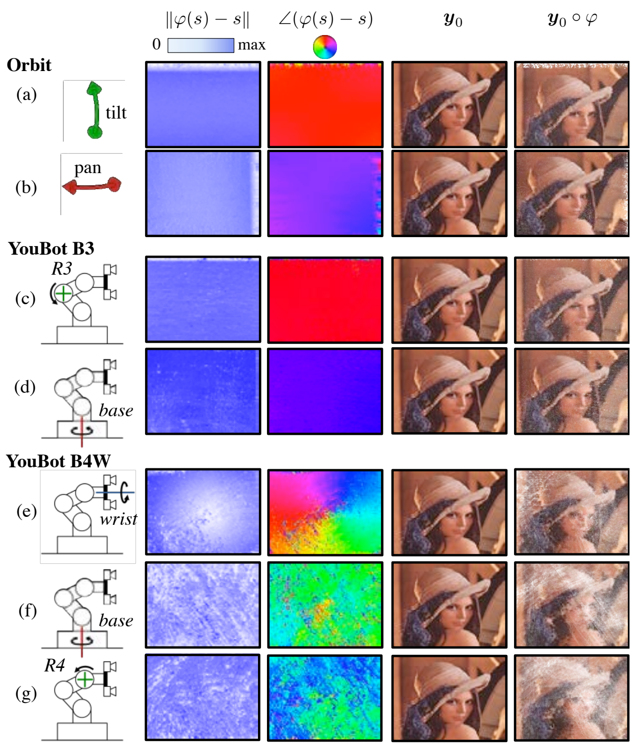


Fig. 2. For visualizing the diffeomorphisms, we show for each command the modulus of the displacement ($\|\varphi(s) - s\|$, with blue $=$ max and white $= 0$), the orientation $\angle(\varphi(s) - s)$ and the prediction of a template image. (a)–(b): Pan and tilt motions of the Orbit camera result in almost pure translations of the visual field, though our learner is precise enough to represent the spherical aberration of the lens. (c)–(d): For the Youbot we consider two different configurations. In the "b3" configuration we move only the base and the third revolute joint, so that the Youbot approximates a very expensive pan-tilt camera. (e)–(g): In the "b4w" configuration, we use the base, the fourth joint, and the wrist. The diffeomorphism corresponding to the wrist can be estimated easily, as shown in (e), and it corresponds to a rotation of the visual field. (f)–(g): It is not possible to associate a unique diffeomorphism to any motion of the base and the fourth joint, because the wrist joint configuration is a hidden state that is not modeled. In those cases the learned diffeomorphisms are just noise.

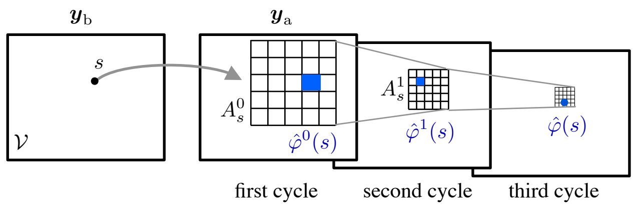


Fig. 3. The method presented in this paper uses a recursive refinement approach. For each pixel s, the search area A_s^j is progressively refined during successive "learning cycles" j. At each learning cycle, the search area is resampled to be a fixed number of pixels g ($g = 5$ in the figure). At the end of a cycle, the best guess for $\varphi(s)$ is used to re-center the search, and the physical area is then shrunk. This is repeated until the resolution matches an arbitrary given resolution, possibly higher than the original image resolution.

in the sense that for two pixels s^1 and s^2, the estimation of $\varphi(s^1)$ can be done in parallel to the estimation of $\varphi(s^2)$.

Fix a point s on the domain. Learning uses the stream of pairs of images $\langle \boldsymbol{y}_\mathrm{a}, \boldsymbol{y}_\mathrm{b} \rangle$. If $s \in \mathcal{V}_\mathrm{pred}$, the DDS dynamics given by (2) and (3) implies that

$$\boldsymbol{y}_\mathrm{b}(s) = \boldsymbol{y}_\mathrm{a}(\varphi(s)).$$

Therefore, to find the unknown $\varphi(s)$ we look for the pixel in $\boldsymbol{y}_\mathrm{a}$ that is most similar to $\boldsymbol{y}_\mathrm{b}(s)$ on average.

The search is restricted by using a bound on the diffeomorphism size, represented as a scalar $\delta > 0$, expressed as a fraction of the size of the viewport $\mathcal{V}$:

$$\|\varphi\| \leq \delta\,|\mathcal{V}|. \tag{4}$$

Define the "search area" A_s as the points in the viewport that are close enough to s to be candidates for $\varphi(s)$:

$$A_s = \{v \in \mathcal{V} \mid d^{\mathcal{S}}(s, v) \leq \delta\,|\mathcal{V}|\}. \tag{5}$$

A dissimilarity function $E_s(v) : A_s \to \mathbb{R}^+$ is computed by averaging on the training data. The value $E_s(v)$ encodes how dissimilar is the sensel s from the sensel v:

$$E_s(v) = \mathbb{E}_{(\boldsymbol{y}_\mathrm{a}, \boldsymbol{y}_\mathrm{b}) \in \mathrm{dataset}}\{|\boldsymbol{y}_\mathrm{a}(v) - \boldsymbol{y}_\mathrm{b}(s)|\}. \tag{6}$$

The minimum of the cost function $E_s(v)$ is taken to be an estimate of the value $\varphi(s)$:

$$\hat{\varphi}(s) \doteq \arg\min_{v \in A_s} E_s(v).$$

The actual implementation uses a discretized domain, such that both s and v range in the discrete set of points $\mathcal{S}$. For convenience, the search area is taken to be rectangular, rather than circular, as (5) would imply.

Because the values of $\varphi(s)$ are estimated independently for each point s, no continuity or smoothness constraints are enforced. This would be actually easy to do, using an energy model or a gaussian process [18], but in the context of bootstrapping we do not want to commit to a particular prior on the diffeomorphism.

In practice, the main issue with this method is the large memory usage. The memory needed is quartic in the resolution of the image. (This will be derived later in Section IV-D). For a computer with 12GB of RAM, for the systems considered in this paper (Fig. 1), we can learn all diffeomorphisms simultaneously only up to a resolution of 160×120.

C. Recursive Discretized Diffeomorphism Learning (RDDL)

The Recursive Discretized Diffeomorphism Learning (RDDL) algorithm retains the same point-wise estimation scheme, but learning is recursively done at increasing resolutions, so that the memory requirement is lower, and, as a bonus, one can also obtain sub-pixel resolution.

A slight difference in the formalization is that RDDL uses "learning cycles". During each learning cycle, the diffeomorphism is estimated at a successively increased resolution. At the end of the cycle, the search area for each pixel is re-centered on the best guess and shrunk by a given factor (Fig. 3). In the experiments presented here, learning is done offline from logs, and each learning cycle passes through all available data.

In addition to the bound δ on the size of the diffeomorphism, the parameters for RDDL include: a desired target resolution $\rho^\star$, potentially higher than the native image resolution; a shrinking factor $\lambda \in (0, 1)$, which describes how quickly the search area is shrunk; an integer $g \geq 3$, which describes the number of pixels to which the search area is down-sampled. At each learning cycle, we change both the search resolution and size of the search area, so that the number of points in the search area is constant and equal to g^2 (Fig. 3).

A high-level description of the algorithm in the continuous domain proceeds as follows. As before, the process is entirely parallel for each point $s \in \mathcal{V}$. The index $j \geq 0$ is used to denote the learning cycles. At the end of each learning cycle there is a guess $\hat{\varphi}^j(s)$ for the value of $\varphi(s)$; the search area is then shrunk and re-centered at that guess. More precisely, let A_s^j be the search area for point s at cycle j. For the first cycle $j = 0$, the search area A_s^0 is centered on s and has radius equal to $\delta\,|\mathcal{V}|$:

$$A_s^0 = \{v \in \mathcal{V} \mid d^{\mathcal{S}}(s, v) \leq \delta\,|\mathcal{V}|\}.$$

The dissimilarity function $E_s^0(v)$ is computed only in the search area A_s^0, and the image is resampled such that a $g \times g$ grid covers the search area. At the end of the learning cycle, the estimate $\hat{\varphi}^0(s)$ of $\varphi(s)$ is computed as the minimum of the dissimilarity function:

$$\hat{\varphi}^0(s) \doteq \arg\min_{v \in A_s^0} E_s^0(v).$$

For successive cycles $j > 0$, the search area A_s^j is centered on the previous guess $\hat{\varphi}^{j-1}(s)$, and it is shrunk of a factor $\lambda \in (0, 1)$, so that at cycle j the search area radius is $\delta\,\lambda^j$:

$$A_s^j = \{v \in \mathcal{V} \mid d^{\mathcal{S}}(\hat{\varphi}^{j-1}(s), v) \leq \delta\,\lambda^j |\mathcal{V}|\}.$$

The image is resampled so that at cycle j the resolution is proportional to $\lambda^{-j}\rho$ and the number of pixels is the search area is constant and equal to $g \times g$. This makes each learning cycle use the same amount of memory.

At the end of the cycle, the guess for $\varphi^j(s)$ is given by the minimum over the area A_s^j:

$$\hat{\varphi}^j(s) \doteq \arg\min_{v \in A_s^j} E_s^j(v).$$

This process continues until the search resolution reaches the desired resolution $\rho^\star$.

The desired resolution is arbitrary, but clearly eventually the noise becomes dominant. For the systems considered in this paper, a resolution of 0.5 pixels seems reasonable. It would be interesting to have a precise bound on the identifiable resolution as a function of the statistics of the images and the various sources of noise.

The implementation is relatively straightforward; the only delicate part is bookkeeping of the various coordinate transformations between the search areas at different resolutions.

D. Complexity

RDDL requires less time and memory than PDDL.

Complexity of PDDL: Let (H, W) be the size of the image in pixels. This can be written as $H = \rho S$ and $W = \rho T$ where S, T is the "physical" size of the domain in $\mathcal{S}$ (e.g., in degrees) and ρ is the resolution (pixel/deg). The number of pixels is $H \times W = \rho^2 ST$, which is quadratic in the resolution. If $\delta \in (0, 1]$ is the search area, expressed as a fraction of the size of the domain, each pixel has a search area of size $\rho H \times \rho W = \delta^2 \rho^2 ST$. Therefore, the memory usage for PDDL is quartic in the resolution:

$$\rho^2 ST \times \delta^2 \rho^2 ST = \rho \delta^2 (ST)^2 = \mathcal{O}(\rho^4).$$

The computational complexity is the same.

Complexity of RDDL: Regardless of the current resolution, at each learning cycle the search area is discretized to $g \times g$ grid, thus the memory usage is $\rho^2 ST \cdot g^2 = \mathcal{O}(\rho^2)$, which compares favorably with the $\mathcal{O}(\rho^4)$ complexity of PDDL.

Simple algebra allows us to find how many cycles are needed to obtain the desired resolution $\rho^\star$. At the j-th learning cycle, the resolution of the search area is $g/(\max\{S,T\} A \lambda^j)$. Therefore, the number of cycles needed to reach the resolution $\rho^\star$ is, assuming $S = T$,

$$j^\star = \lceil (\log g - \log \rho^\star S \delta) / \log \lambda \rceil .$$

The computation cost, which is fixed for each cycle, is $\mathcal{O}(g^2)$ for each of the $\rho^2 ST$ pixels in the image. The total cost is $j^\star$ times the cost of one iteration:

$$\lceil (\log g - \log \rho^\star S \delta) / \log \lambda \rceil \rho^2 ST g^2.$$

To compare the two method, suppose the desired resolution $\rho^\star$ to be equal to the native resolution ρ. Then RDDL has computational complexity $\mathcal{O}(\rho^2 \log \rho)$ compared to $\mathcal{O}(\rho^4)$ for PDDL.

V. Estimation of diffeomorphism uncertainty

This section describes how we represent uncertainty in images and in diffeomorphisms, the previous method that was used to estimate it, which was based on looking at the residuals of an error function, and an alternative method, which is based on checking the consistency of the estimated φ and φ^{-1}.

A. Uncertainty of images and diffeomorphisms

It is necessary to have some notion of uncertainty to deal with phenomena such as the sensor's limited field of view: if the viewport $\mathcal{V} \subset \mathcal{S}$ is not the entire domain $\mathcal{S}$, for some motions it is not possible to predict the values of the observations in the whole viewport, as they depend on the "hidden" part of the domain.

In principle, one would need to be able to represents an arbitrary probability distribution over images and diffeomorphisms. Given a starting image, the prediction of our models should be a distribution over images. However, it is intractable to do this in full generality; here, we use a very simple approximation that associates a pointwise scalar "certainty" value to images and diffeomorphisms.

Uncertain images: The set of "images" $\mathsf{Im}(\mathcal{S})$ was defined to be the set of all functions from $\mathcal{S}$ to $\mathbb{R}$. Here we extend it to the set of "uncertain images" $\mathsf{UIm}(\mathcal{S})$, which is meant to be a tractable, very thin slice of all probability measures on $\mathsf{Im}(\mathcal{S})$.

An element of $\mathsf{UIm}(\mathcal{S})$ is a pair $(\boldsymbol{y}, \boldsymbol{z})$, where $\boldsymbol{y} \in \mathsf{Im}(\mathcal{S})$ is a normal image, and $\boldsymbol{z} : \mathcal{S} \to [0, 1]$ is a scalar "certainty" value, with the following semantics: $z(s) = 1$ if we know exactly $y(s)$; and $z(s) = 0$ if we know nothing about $y(s)$.

Values of $z(s)$ between 0 and 1 account for intermediate cases; while, in principle, we could give it a precise probabilistic interpretation, the approximations that are done in learning and propagating this uncertainty do not allow a precise probabilistic semantics to hold in practice.

Given a pair of uncertain images $\langle \boldsymbol{y}_1, \boldsymbol{z}_1 \rangle$ and $\langle \boldsymbol{y}_2, \boldsymbol{z}_2 \rangle \in \mathsf{UIm}(\mathcal{S})$, the distance between them is defined by extending the definition of distance on $\mathsf{Im}(\mathcal{S})$ given by (1) to account for the uncertainty, by discounting the differences between the images $\boldsymbol{y}_1, \boldsymbol{y}_2$ according to their uncertainties $\boldsymbol{z}_1, \boldsymbol{z}_2$:

$$d_{L_p}^{\mathsf{UIm}(\mathcal{S})}(\langle \boldsymbol{y}_1, \boldsymbol{z}_1 \rangle, \langle \boldsymbol{y}_2, \boldsymbol{z}_2 \rangle) \doteq \tag{7}$$

$$\tfrac{1}{|\mathcal{S}|} \left(\frac{\int z_1(s) z_2(s) \left| y_1(s) - y_2(s) \right|^p \, \mathrm{d}s}{\int z_1(s) z_2(s) \, \mathrm{d}s} \right)^{\frac{1}{p}} .$$

Uncertain diffeomorphisms: In the same spirit, we lift the diffeomorphisms $\mathsf{Diff}(\mathcal{S})$ to an enlarged space of "uncertain" diffeomorphisms $\mathsf{UDiff}(\mathcal{S})$, which contains tuples (φ, γ), where $\varphi \in \mathsf{Diff}(\mathcal{S})$, and the scalar field $\gamma : \mathcal{S} \to [0, 1]$ is a scalar uncertainty that represents the "certainty" about $\varphi(s)$.

B. Residuals-based uncertainty estimation method

In previous work we quantified the confidence in the estimate $\hat{\varphi}(s)$ as the value $\hat{\gamma}_{\text{res}}(s)$, which was the minimum of the dissimilarity function E_s (defined in (6)):

$$\hat{\gamma}_{\text{res}}(s) = \min_{v \in A_s} E_s(v).$$

The values of $\hat{\gamma}_{\text{res}}(s)$ are linearly scaled as to obtain values in the interval $[0, 1]$ over the whole domain.

The rationale for using this measure is that at the minimum the residual dissimilarity should be equal to the variance of the observations noise (the term $\epsilon(s)$ in the observation model (3)), if s belongs to the predictable region $\mathcal{V}_{\text{pred}}$, and should have a higher value for points in the unpredictable region $\mathcal{V}_{\text{unpr}}$.

In practice, however, this confidence measure is very sensitive to several unmodeled phenomena (Fig. 4*a,b*), such as aliasing effects, noise that acts on the commands (thus inducing a slightly different diffeomorphism), and it is not invariant to the image statistics (e.g., the uncertainty is underestimated in regions with more uniform texture).

C. A consistency-based uncertainty estimation method

A simple alternative way to estimate the confidence can be obtained by checking the consistency of the estimated diffeomorphism.

For each command, we learn both an estimate of φ as well as an estimate of φ^{-1}; denote these by $\hat{\varphi}$ and $\widehat{\varphi^{-1}}$, respectively. In principle, we would expect that $\hat{\varphi}$ and $\widehat{\varphi^{-1}}$ are inverse of each other, meaning that, for all s, $\hat{\varphi}(\widehat{\varphi^{-1}}(s)) = s$, as well as $\widehat{\varphi^{-1}}(\hat{\varphi}(s)) = s$. These consistency conditions do not hold if $s \in \mathcal{V}_{\text{unpr}}$, and do not hold exactly because of noise and other unmodeled effects as described before that affect $\hat{\varphi}$ and $\widehat{\varphi^{-1}}$ differently.

Therefore, our estimation of uncertainty $\hat{\gamma}_{\text{c,a}}(s)$ is a quantification of how much the consistency conditions are violated by measuring the distance between s and $\hat{\varphi}(\widehat{\varphi^{-1}}(s))$:

$$\hat{\gamma}_{\text{c,a}}(s) = d^{\mathcal{S}}(s,\ \widehat{\varphi^{-1}} \circ \hat{\varphi}\,(s)). \tag{8}$$

A variation on this idea is to normalize the deviation by the size of the diffeomorphism:

$$\hat{\gamma}_{\text{c,n}}(s) = \frac{d^{\mathcal{S}}(s,\ \widehat{\varphi^{-1}} \circ \hat{\varphi}\,(s))}{d^{\mathcal{S}}\,(s,\ \hat{\varphi}\,(s))}. \tag{9}$$

These pointwise values are then linearly normalized in the $[0, 1]$ range over the whole domain.

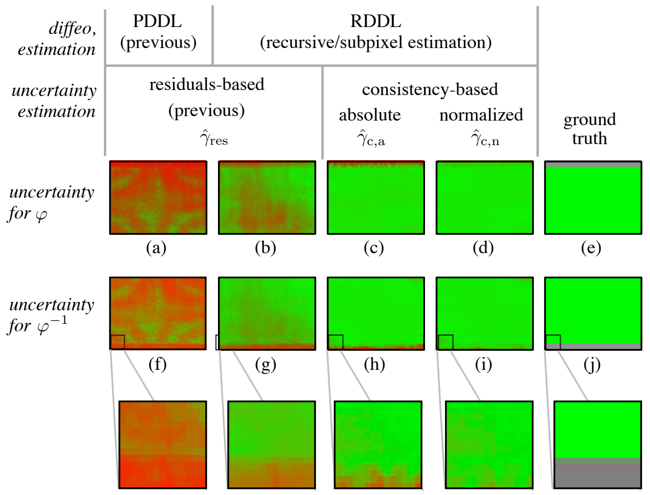

diffeo. estimation
PDDL (previous)
RDDL (recursive/subpixel estimation)
uncertainty estimation
residuals-based (previous) $\hat{\gamma}_{\text{res}}$
consistency-based
absolute $\hat{\gamma}_{\text{c,a}}$
normalized $\hat{\gamma}_{\text{c,n}}$
ground truth
uncertainty for φ
uncertainty for φ^{-1}
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)

Fig. 4. Qualitative comparision among the certainty estimated by the different methods discussed in the paper; a quantative comparison is given in Fig. 7. All data is for the ORBIT dataset and the command "tilt up". Red corresponds to a value of 0 and green to 1. The first row shows the uncertainty for the diffeomorphsim φ, and the second line shows the uncertainty for the inverse diffeomorphisms φ^{-1}. The last column shows the ground truth obtained by prior knowledge of the dynamics. The green area should match $\mathcal{V}_{\text{pred}}$, which is the part that we are not able to predict based on the current image alone. The first column (subfigures *a,f*) shows the uncertainty field estimated by the residuals-based method (Section V-B) based on the previous method PDDL for diffeomorphism estimation (Section IV-B). The residual does not allow to distinguish clearly between $\mathcal{V}_{\text{pred}}$ and $\mathcal{V}_{\text{unpr}}$. The third column shows the same residuals-based method (Section V-B) but with the diffeomorphism estimated using the new methood RDDL (Section IV-C) using parameters $\lambda = 1/5$, $g = 15$, which gives a resolution of 0.5 pixels. The classification is much clearer, probably because the diffeomorphism is more precise. The next two columns show the results using the consistency-based estimated uncertaintity (Section V-C), using the absolute deviation, defined by (8), as well as the normalized relative deviation, defined by (9). Compared to the previous method, there is a much clearer classification in $\mathcal{V}_{\text{unpr}}$ and $\mathcal{V}_{\text{pred}}$.

VI. Experiments

A. Data collection and setup

The ORBIT pan-tilt camera dataset is the same used in previous work [14]. It contains 1000 samples, with a mix of indoor and outdoor environments.

We use two Youbot datasets that differ for the degrees of freedom activated. The first, denoted by YOUBOTB3, uses the base and the third revolute joint (thus making it a very expensive pan-tilt camera) and has 348 samples; the second, denoted by YOUBOTB4W, uses the base, the fourth revolute joint, and the wrist, and has 563 samples.

B. Memory usage and speed

Experiments show that RDDL uses less memory than PDDL. For instance, for the ORBIT system with 4 commands, PDDL can learn simultaneously the diffeomorphisms for all commands only up to a resolution of 160×120, while there is no hard-limit on the resolution for RDDL because of the recursive approach.

RDDL is faster than PDDL, at least asymptotically. Fig. 5*a* shows the time needed for learning from the whole ORBIT dataset as a function of the resolution in a log-log scale. The figure shows the results for: PDDL ($\delta = 0.35$), and two parametrizations of RDDL ($\delta = 0.35$, $\lambda = 1/3$, $g = 9$; and $\lambda = 1/5$, $g = 15$). Fig. 5*b* shows the number of learning cycles for obtaining a desired resolution, which depends on the combination of the parameters λ and g. The different slopes in the log-log graph reflects the difference in the computational complexity ($\mathcal{O}(\rho^4)$ vs $\mathcal{O}(\rho^2 \log \rho)$), as derived in Section IV-D. The graph for PDDL stops at the resolution 160×120.

The relatively unoptimized Python implementation is quite slow; at 160×120 it takes around 1 s for integrating the information in one sample $\langle \boldsymbol{y}_{\text{a}}, \boldsymbol{y}_{\text{b}} \rangle$.

C. Precision

RDDL is as precise as PDDL, as measured by their performance in a prediction task, in which the observations predicted by the learned models are compared with the actual observations.

Fix a prediction horizon by choosing an integer $\ell \geq 1$. From the log $\{\langle \boldsymbol{y}_k, \boldsymbol{u}_k \rangle\}_k$ we extract a sequence of tuples of the form

$$\langle \boldsymbol{y}_k, \underbrace{\{\boldsymbol{u}_k, \dots, \boldsymbol{u}_{k+\ell-1}\}}_{\text{"plan" } p_k}, \boldsymbol{y}_{k+\ell} \rangle.$$

We call "plan" the the sequence p_k of the ℓ commands between the two images $\boldsymbol{y}_k$ and $\boldsymbol{y}_{k+\ell}$. The evaluation consists in comparing the observations $\boldsymbol{y}_{k+\ell}$ with the prediction given the initial observations $\boldsymbol{y}_k$ and the plan p_k. The predicted $\boldsymbol{y}_{k+\ell}$ is obtained from $\boldsymbol{y}_k$ by applying the ℓ learned diffeomorphisms corresponding to the commands in p_k. For comparing the predicted image with the observed image we use the L_2 distance defined in (1) and the uncertainty-weighted distance defined in (7).

Fig. 6*a* show the L_2 prediction error for: the previous method PDDL, the proposed method RDDL ($\lambda = 1/3, g = 9$), and, for illustration purposes, the result of RDDL after only the first resolution. In all cases, RDDL is as precise as PDDL.

D. Uncertainty estimation

Experiments show that the consistency-based uncertainty estimation method (Section V-C) is more accurate than the residual-based method (Section V-B).

We first discuss a qualitative comparison, where we compare visually the methods to see whether they are able to distinguish between the predictable region $\mathcal{V}_{\text{pred}}$ and the unpredictable region $\mathcal{V}_{\text{unpr}}$.

Here we use the ORBIT dataset, and we look and the command corresponding to "tilt up". For the diffeomorphism induced by this command, the unpredictable region $\mathcal{V}_{\text{unpr}}$ is the top part (Fig. 4*e*). For the inverse diffeomorphism, $\mathcal{V}_{\text{unpr}}$ is the bottom part (Fig. 4*j*).

The uncertainty estimation method is independent of the diffeomorphism estimation method, so we show various combinations. The first two columns show the result for the residuals-based uncertainty estimation method (Section V-C); the first column is for the previous diffeomorphism estimation method PDDL and the second for the improved method RDDL, with a higher target resolution (0.5 pixels). We attribute the much better classification between $\mathcal{V}_{\text{pred}}$ and $\mathcal{V}_{\text{unpr}}$ to the improved accuracy of RDDL. The third and fourth columns show the result for the new consistency-based uncertainty estimation method (Section V-B), for the absolute and relative consistency measures, defined in (8) and (9). The relative estimation method looks qualitatively better.

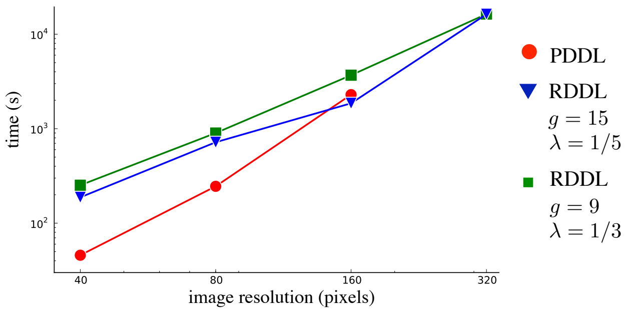
A quantitative comparison among the uncertainty estimation methods is more rigorous but less intuitive.

Fig. 7 shows the prediction error computed using the uncertainty-weighted L_2 distance defined in (7) for all combinations of diffeomorphism and uncertainty estimation. Experimentally we found that, if the uncertainty is not computed well, there are some non-intuitive phenomena. For example, using the usual L_2 distance (not weighted by uncertainty), the prediction error always increases over the prediction horizon (see for example Fig. 6). If the uncertainty is not well estimated, there is a non-intuitive result that the prediction error computed using the weighted distance actually decreases on average with the length of the prediction horizon. Therefore, looking at the trend of the uncertainty as the prediction horizon increases gives a quantitative measure that the estimated uncertainty reflects the true uncertainty. The data in Fig. 7 shows that the consistency-based estimation of uncertainty has a better trend than the residuals-based estimation.

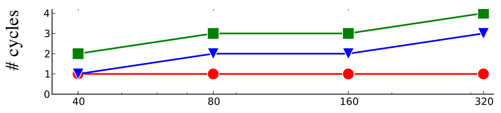
VII. Conclusions

A diffeomorphism dynamical system (DDS), in which each command induces a diffeomorphism of the observations, is a relatively simple approximation of the sensel-level dynamics of robotic sensorimotor cascades.

This paper has presented two improvements for estimating diffeomorphisms and their uncertainty. The recursive learning method (IV-C) allows to learn diffeomorphisms at higher resolution than the previous method because of the reduced memory needed. The consistency-based uncertainty estimation method (V-C) provides a better estimation of the uncertainty, as seen for example in the clearer distinction between predictable and unpredictable regions $\mathcal{V}_{\text{pred}}$ and $\mathcal{V}_{\text{unpr}}$.



(a) Computation time



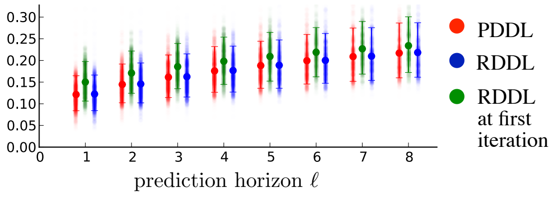
(b) Cycles for reaching a given resolution

Fig. 5. (a) Learning time for PDDL and RDDL for two choices of the parameters ($\lambda = 1/3, g = 9$ and $\lambda = 1/5, g = 15$) as a function of the image resolution (from 40×30 to 320×240), for the ORBIT dataset composed of 1000 samples. The graph for PDDL interrupts at 160×120, which is the largest resolution that is possible to obtain if we want to learn all diffeomorphisms at the same time for a system with 4 commands on a computer with 12 GB of RAM. The methods have different computational complexity: RDDL has cost $\mathcal{O}(\rho^2 \log \rho)$, while PDDL has cost $\mathcal{O}(\rho^4)$ for PDDL. In this Python implementation they take approximately the same time around the 160×120 resolution, PDDL having been more optimized. (b) Number of learning cycles required to reach the required resolution.

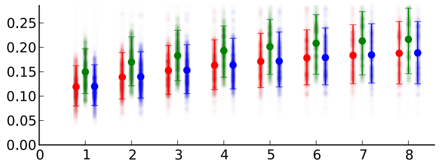
Our current work consists in dealing with hidden states and continuously-varying commands. The DDS dynamics that we learn associates one diffeomorphism to one command. This is only an approximation, because, in general, the diffeomorphism depends also on other states, which could be observable (such as the joints configuration) or hidden (such as the 3D shape of the environment). For instance, this approach fails in learning the diffeomorphism associated to the Youbot base and third joint, because they depend on the configuration of the wrist joint (1*f*–*g*).

Let $\boldsymbol{q} \in \mathcal{Q}$ represents the observable states. In principle, if a state of observable, we could learn a different diffeomorphism for each state and command; instead of the map $\Phi : \mathcal{U} \to \mathsf{Diff}(\mathcal{S})$ we could learn the map $\Phi : \mathcal{U} \times \mathcal{Q} \to \mathsf{Diff}(\mathcal{S})$. However, this would need much more data: if the state space $\mathcal{Q}$ is discretized to 100 states, we would need 100 times the learning data. To avoid the need of more data, we are considering an approach based on Gaussian processes (GP) [18]. Let $E_s^{\boldsymbol{u},\boldsymbol{q}}(v)$ be the dissimilarity for sensels $s, v \in \mathcal{S}$ for a given command $\boldsymbol{u} \in \mathcal{U}$ at an observable state $\boldsymbol{q} \in \mathcal{Q}$. Then we are considering either representing the diffeomorphism $\varphi^{\boldsymbol{u},\boldsymbol{q}}(s)$ as a GP with state $\langle \boldsymbol{u}, \boldsymbol{q}, s \rangle \in \mathcal{U} \times \mathcal{Q} \times \mathcal{S}$ and observations in $\mathcal{S}$, or representing directly the dissimilarity $E_s^{\boldsymbol{u},\boldsymbol{q}}(v)$ as as a scalar-valued GP in the variables $\langle \boldsymbol{u}, \boldsymbol{q}, s, v \rangle \in \mathcal{U} \times \mathcal{Q} \times \mathcal{S} \times \mathcal{S}$. This approach would reduce the data needed for learning, and as a bonus it would allow interpolating between diffeomorphisms associated to different commands.

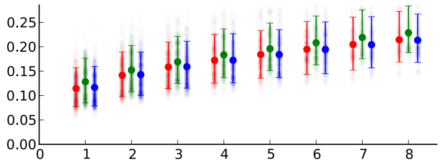
Dealing with states that are not directly observable is a more complicated goal. Similar functionality has been shown in some works in deep learning, which demonstrated learning of a collection of linear dynamical systems with discrete hidden states [19].



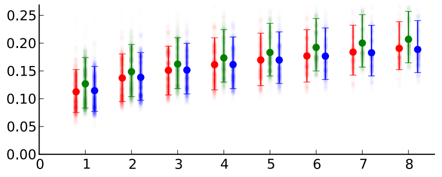
(a) Prediction error, ORBIT dataset, L_2 distance



(b) Prediction error, ORBIT dataset, uncertainty-weighted L_2 distance



(c) Prediction error, YOUBOTB3 dataset, L_2 distance



(d) Prediction error, YOUBOTB3 dataset, uncertainty-weighted L_2 distance

Fig. 6. Quantitative comparison of the two learning methods PDDL and RDDL on a prediction task. Two datasets are used (Section VI-A): (a)-(b) refer to the ORBIT dataset, (c)-(d) refer to the YOUBOTB3 dataset. Once fixed a prediction horizon ℓ, for all images $\boldsymbol{y}_k$ in the dataset, we compare the observed $\boldsymbol{y}_{k+\ell}$ with a prediction based on $\boldsymbol{y}_k$ and the learned model. We use two different ways to measure distances on the set of uncertain images: one based on the normal L_2 distance, used in (a) and (c), and one based on the uncertainty-weighted L_2 distance (defined in (7)), used in (b) and (d). In all cases the precision RDDL is the same as PDDL. The figures also report the precision of RDDL with the approximated result after only the first learning cycle.

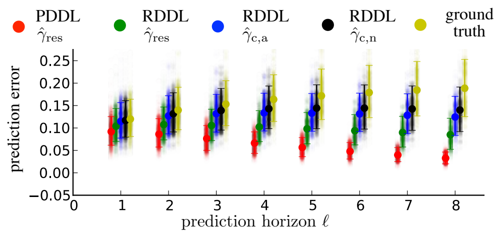


Fig. 7. Quantitative evaluation for uncertainty estimation for the same methods in the qualitative comparison in Fig. 4. The graph shows the prediction error measured with the uncertainty-weighted L_2 norm, defined in (7), as a function of the prediction horizon ℓ. We have noticed that the quality of the uncertainty estimation can be quantified by the trend over a long prediction horizon. We expect that the prediction error grows with the horizon, as it happens for the ground truth uncertainty. When the uncertainty is not well approximated, we find the paradoxical behavior that the trend is negative, like for the residuals-based method. The better behavior of the consistency-based methods reflects the qualitative measurement in Fig. 4, except that the relative measure (Fig. 4*c*) performs slightly worse than the absolute (Fig. 4*d*).

References

[1] A Stoytchev. "Some Basic Principles of Developmental Robotics". In: *IEEE Transactions on Autonomous Mental Development* 1.2 (2009) DOI:10.1109/ TAMD.2009.2029989.

[2] D. Pierce and B. Kuipers. "Map learning with uninterpreted sensors and effectors". In: *Artificial Intelligence* 92.1-2 (1997) DOI:10.1016/ S0004-3702(96)00051-3.

[3] B. Kuipers. "An intellectual history of the Spatial Semantic Hierarchy". In: *Robotics and cognitive approaches to spatial mapping* 38 (2008).

[4] 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:10.1109/ DEVLRN.2008.4640813.

[5] M. Boerlin, T. Delbruck, and K. Eng. "Getting to know your neighbors: unsupervised learning of topography from real-world, event-based input". In: *Neural computation* 21.1 (2009) DOI:10.1162/ neco.2009.06-07-554.

[6] 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:10.1109/ DEVLRN.2010.557885.

[7] A. Censi and D. Scaramuzza. *Calibration by correlation using metric embedding from non-metric similarities*. Tech. rep. CaltechAUTHORS:20120805-103228127. (To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013). California Institute of Technology, 2012 (url).

[8] J. Sturm, C. Plagemann, and W. Burgard. "Body Schema Learning for Robotic Manipulators from Visual Self-Perception". In: *Journal of Physiology* (2009).

[9] A. Censi. *Bootstrapping Vehicles: A Formal Approach to Unsupervised Sensorimotor Learning Based on Invariance*. Tech. rep. California Institute of Technology, 2012 (url).

[10] 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)*. Shanghai, China, 2011 DOI:10.1109/ ICRA.2011.5979844.

[11] A. Censi and R. M. Murray. "Bootstrapping sensorimotor cascades: a group-theoretic perspective". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, 2011 DOI:10.1109/ IROS.2011.6095151.

[12] A. Censi, M. Hakansson, and R. M. Murray. "Fault detection and isolation from uninterpreted data in robotic sensorimotor cascades". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, 2012 DOI:10.1109/ ICRA.2012.6225311.

[13] A. Censi and R. M. Murray. "Learning diffeomorphism models of robotic sensorimotor cascades". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, 2012 DOI:10.1109/ ICRA.2012.6225318.

[14] A. Censi, A. Nilsson, and R. M. Murray. "Motion planning in observations space with learned diffeomorphism models." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2013.

[15] R Memisevic and G. Hinton. "Learning to represent spatial transformations with factored higher-order Boltzmann machines". In: *Neural Computation* 22.6 (2010) DOI:10.1162/ neco.2010.01-09-953.

[16] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. "Convolutional Learning of Spatio-temporal Features". In: *Proceedings of the European Conference on Computer Vision*. 2010 DOI:10.1007/ 978-3-642-15567-3_11.

[17] M. do Carmo. *Riemannian Geometry*. Birkhauser, 1994. ISBN: 3-540-20493-8.

[18] C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006.

[19] I. Sutskever, G. E. Hinton, and G. W. Taylor. "The Recurrent Temporal Restricted Boltzmann Machine". In: *Advances in Neural Information Processing Systems (NIPS)*. 2008.