Bootstrapping bilinear models of robotic sensorimotor cascades

Andrea Censi   Richard M. Murray

Abstract—We consider the bootstrapping problem, which consists in learning a model of the agent's sensors and actuators starting from zero prior information, and we take the problem of servoing as a cross-modal task to validate the learned models. We study the class of sensors with bilinear dynamics, for which the derivative of the observations is a bilinear form of the control commands and the observations themselves. This class of models is simple, yet general enough to represent the main phenomena of three representative sensors (field sampler, camera, and range-finder), apparently very different from one another. It also allows a bootstrapping algorithm based on Hebbian learning, and a simple bioplausible control strategy. The convergence properties of learning and control are demonstrated with extensive simulations and by analytical arguments.

I. Introduction

Two features of natural intelligence that we are far from emulating in artificial systems are its adaptiveness and generality. The human neocortex is highly uniform and its parts can be repurposed; for example, the visual cortex is repurposed to process tactile information in blind subjects [1]. Reproducing the same adaptability in artificial systems is a vast problem considered in various forms (and by various names) in several fields, such as AI, machine learning, robotics, and the specialized (and fairly distinct) fields of epigenetic and developmental robotics [2], [3].

A rather extreme, yet concrete, version of the problem has been put forward by Kuipers and colleagues in a long series of papers (see [4], [5], [6] and references therein). Suppose that an agent starts its life with no prior information about its sensors and actuators. It can read the sensors output as a sequence of values, but no semantics is associated to them. Likewise, it does not know how the unlabeled commands it can generate affect the world. The bootstrapping problem concerns creating a model for its sensorimotor cascade from scratch, and using it to achieve useful tasks.

Bootstrapping can be seen as an extreme form of system identification/calibration. Currently, there exist autocalibration techniques that can estimate parametric models of the dynamics (for example, the odometric parameters) or the extrinsic sensor configuration, (although the solutions, rather than general, tend to be tailored to specific sensors or groups of sensors), but always the type of sensors/actuators being calibrated is known a priori. Can a robot learn to use an unknown sensor and unknown actuators? Can the *same* learning algorithm work for a range-finder and a camera? These are, at the moment, open questions. They are important for practical robotics applications: it would be extremely convenient, if you could just attach any sensor to a robot, and the robot

would learn how to use it, without tedious programming. More in general, we believe that robotic systems are the perfect benchmark for supposedly "universal learning agents", which so far have been studied for only perception/classification tasks [7], or as body-less agents [8].

The most complete exemplification of Kuipers and colleagues' idea of a bootstrapping agent is the Spatial Semantic Hierarchy [9]. They show that it is possible to start from uninterpreted sensor data and build successive layers of representation up to a topological map of the environment. The crucial transition from continuous sensor values to symbolic representations depends on the definition of *trackers*, distinctive features in the sensory stream whose behavior is predicted by the agent's action. Their work offers several opportunities for extension. One concern is that their results remain largely anecdotal, in the sense that they are illustrated by simulations/experiments, but not by proofs; this makes it hard to build on them. The other concern is that most of the results regard the case of range-finders: while they showed, in principle, that the same design pattern could be applied to different sensor modalities, it is unclear whether exactly the same algorithm could be run unchanged for different sensorimotor cascades. The two aspects of provability and generality are our focus in this paper.

This paper shows that it is possible to design *unsupervised* learning algorithms that learn *generative* models of a robotics sensorimotor cascade, for a wide range of sensors, and use that model to perform useful tasks. Moreover, the model we consider is simple enough that many of its properties can be proved theoretically.

Throughout the paper, we consider three classes of sensors: *cameras; range-finders*, devices sensing the distance to the closest obstacle; and *field samplers,* devices that sample a generic spatial field, such as the concentration of a chemical substance. We call these three "canonical" sensors, in the sense that they are representative of many others. For example, the range-finder abstraction encompasses both sparse
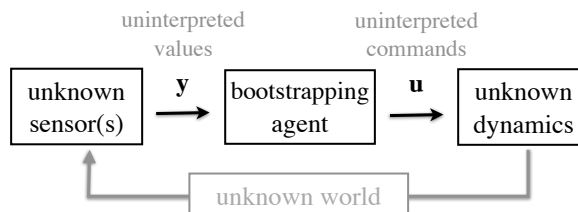

Figure 1. The bootstrapping problem. Can an agent with no prior information about its sensors and actuators learn a model of its sensorimotor cascade, and use it for useful tasks? Being able to solve this problem would allow us to realize truly zero-calibration plug-and-play robotics.

Department of Control & Dynamical Systems, Division of Engineering and Applied Science, California Institute of Technology, Pasadena, MC 107-81, 91125, CA, USA. E-mail: {andrea,murray}@cds.caltech.edu

sonar-like sensors, as well as dense lidar-like sensors. The camera abstraction encompasses RGB and infrared cameras. The field-sampler is general enough to represent olfactory and temperature sensors (see, e.g., [10], [11]). The idea is that, even if these three sensors do not capture all possible sensors, they cover enough ground such that, if we present an algorithm that can work for all of these, then it would be fair to think it is a "generic" algorithm. As for the robot dynamics, we limit the analysis to fully actuated robots controlled in velocity (still, the agent does not know which command is which).

Models are only as good as the actions they generate. To show that the bootstrapping agent has acquired a useful model of its sensorimotor cascade, we consider its performance in the servoing task. Let $\boldsymbol{y}$ be the observations (for example, the raw pixel intensities returned by a camera), and let $\boldsymbol{u}$ be the commands. We define servoing as follows.

Problem I.1. *Given a goal observation $\boldsymbol{y}_\star$, choose $\boldsymbol{u}(t)$ such that $\boldsymbol{y}(t) \to \boldsymbol{y}_\star$.*

This is an interesting task to consider because the problem statement is simple, it makes sense for all sensor modalities, and the ability to solve it means that the agent has captured a significant part of its sensorimotor cascade's actual model.

Our approach has been to study a model which is fairly general, yet simple enough to be learned and analyzed. In Section II we consider the abstract class of *bilinear dynamics sensors* and we design a two-phase bootstrapping strategy. During a first unsupervised learning phase, the agent learns a representation of its sensorimotor cascade using a simple Hebbian-learning based algorithm. In the second phase, the control action solving the servoing task is given as a function of the learned model. In Section III the algorithm is validated in simulation, for various configurations of the three sensors. Section IV concerns actually proving that the algorithm works for the three canonical sensors.

Most of the proofs are omitted and reported in an extended version of this paper [12].

II. Bootstrapping bilinear dynamics sensors

At the heart of bootstrapping, there is a problem of prediction: how do actions change the agent's view of the world? Specifically, how do the actions $\boldsymbol{u}$ change $\boldsymbol{y}$? Any model we decide to use must be general enough to be applied to different sensorimotor cascades, must be informative enough so that we can use it to solve the problem of servoing, and it must be simple enough so that it is easy to learn and analyze. In this section, we argue that the simplest yet useful model for robotics sensorimotor cascades is assuming that $\dot{\boldsymbol{y}}$ is a bilinear form of $\boldsymbol{y}$ and $\boldsymbol{u}$; if this holds exactly, we call the sensor a *bilinear dynamics sensor* (BDS). We then study the problem of learning unsupervisedly the model of a BDS and how to use that model for servoing.

Notation and formal definitions: We consider sensors composed of a set of sensory elements (*sensels*) that are physically related to one another. We write the observations as $\boldsymbol{y} = \{y^s\}_{s \in \mathcal{Y}}$, where s is the sensel position ranging over the *sensel space* $\mathcal{Y}$. In the case of a camera, the sensels span

the visual sphere $\mathbb{S}^2$; s corresponds to a pixel's direction, and y^s to the intensity measured by that pixel. Real robots have discrete sensors with a finite number of sensels; but to make the derivation simpler we pretend that the sensel space $\mathcal{Y}$ is continuous. The values returned by the sensors lie in a certain *output space* $\mathcal{O}$. For a color camera, $\mathcal{O}$ would be the RGB space; for a range-finder, $\mathcal{O}$ would be $\mathbb{R}^+$ (distances). For simplicity, we will just assume $\mathcal{O}$ to be $\mathbb{R}$; everything can be extended to more complicated output spaces. At each time, the sensor returns the observations as a function from $\mathcal{Y}$ to $\mathbb{R}$, assumed differentiable. A formal signature for the observations $\boldsymbol{y}$ is $\boldsymbol{y}$: time $\to C^1(\mathcal{Y}; \mathbb{R})$.

We assume that there is an inner product defined on $C^1(\mathcal{Y}; \mathbb{R})$. This means that we have a way to measure the dissimilarity of two observations by the norm induced by the inner product. We use the tensorial notation to represent the inner product. Let y^s represent the value of $\boldsymbol{y}$ at the sensel $s \in \mathcal{Y}$. We put the index up in "y^s" with analogy to covariant tensors. Given two observations $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, their inner product $\langle\!\langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle\!\rangle$ can be represented by contracting y_1^s, y_2^v with a $(0, 2)$ tensor m_{sv}: $\langle\!\langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle\!\rangle = m_{sv} y_1^s y_2^v$. Here, using the Einstein convention, summation (integration) is assumed over indices that appear twice (up and down). The inner product allows to define a norm $\|\boldsymbol{y}\|^2 = \langle\!\langle \boldsymbol{y}, \boldsymbol{y} \rangle\!\rangle$ as well as a conjugation operation $\boldsymbol{y} \mapsto \boldsymbol{y}^*$ by $y_s^* = m_{sv} y^v$.

Why using a bilinear model: In the most general case, a continuous-time dynamical system can be written as $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$; $\boldsymbol{y} = h(\boldsymbol{x})$, where $\boldsymbol{x}$ represents the hidden state. However, one seldom sees an explicit model of this kind for sensors such as cameras or range finders, because the function h should encode all information regarding the environment, and a closed form is impossible to write except in the simplest of environments. One alternative representation is focusing on the observations dynamics $\dot{\boldsymbol{y}} = g(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{x})$. In Section IV we show that, for the three canonical sensors, this can actually be written in a closed form. In most cases, the function g depends on the underlying unobservable state $\boldsymbol{x}$. An agent that does not have access to the state $\boldsymbol{x}$ (and its dynamics) cannot learn such a model. This motivates us to look at approximating the observations dynamics by disregarding the dependence on the state, thus looking for models of the form $\dot{\boldsymbol{y}} = g(\boldsymbol{y}, \boldsymbol{u})$. Because the agent has access to $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, and $\boldsymbol{u}$, learning the map g from the data is a well-defined problem.

Rather than trying to learn a generic nonlinear g, which appears to be a daunting task, especially for cases where $\boldsymbol{y}$ consists of thousands of elements (pixels of a camera), our approach has been to keep simplifying the model until one obtains something tractable. For example, a second-order linearization of g leads to the expression

$$\dot{\boldsymbol{y}} = \boldsymbol{a} + A\boldsymbol{y} + B\boldsymbol{u} + C(\boldsymbol{y}, \boldsymbol{y}) + D(\boldsymbol{y}, \boldsymbol{u}) + E(\boldsymbol{u}, \boldsymbol{u}). \quad \text{(II.1)}$$

Here A and B are linear operators, but C, D, E are tensors (later we make the tensor notation more precise). If $\boldsymbol{y}$ and $\boldsymbol{u}$ have dimensions n and k, then C, D, E have dimensions, respectively, $n \times n \times n$, $n \times n \times k$, and $n \times k \times k$.

We can ignore some terms in (II.1) by using some assumptions regarding our specific context. For example, if we assume that $\boldsymbol{u}$ represents a "movement" or "velocity" command, in

the sense that if $\boldsymbol{u}$ is 0, then the pose does not change, and $\boldsymbol{y}$ does not change as well ($\boldsymbol{u} = 0 \Rightarrow \dot{\boldsymbol{y}} = 0$), we can omit the terms $\boldsymbol{a}$, $A\boldsymbol{y}$ and $C(\boldsymbol{y}, \boldsymbol{y})$, and we are left with $\dot{\boldsymbol{y}} = B\boldsymbol{u} + D(\boldsymbol{y}, \boldsymbol{u}) + E(\boldsymbol{u}, \boldsymbol{u})$.

If we assume that $\boldsymbol{u}$ is a symmetric velocity commands, in the sense that applying $+\boldsymbol{u}$ gives the opposite effect of applying $-\boldsymbol{u}$, then we can get rid of the $E(\boldsymbol{u}, \boldsymbol{u})$ term as well. We are left with the model $\dot{\boldsymbol{y}} = B\boldsymbol{u} + D(\boldsymbol{y}, \boldsymbol{u})$, where D is a bilinear operator. We can incorporate $B\boldsymbol{u}$ into the second term by assuming there is a trivial observation whose value is always 1. In conclusion, our *ansatz* for a generic robotic sensor is a bilinear model of the kind $\dot{\boldsymbol{y}} = D(\boldsymbol{y}, \boldsymbol{u})$.

Definition 1. A sensor is a bilinear dynamics sensor (BDS), for a certain choice of control commands $\boldsymbol{u}$, if the derivative of $\boldsymbol{y}$ depends linearly on $\boldsymbol{u}$ and $\boldsymbol{y}$. In formulas, there exists a $(1,2)$ tensor $\mathbf{M}$ such that

$$\dot{y}^s = \mathrm{M}^s_{vi} y^v u^i. \tag{II.2}$$

Bootstrapping and servoing with BDS*:* As explained in the introduction, the task we focus is servoing/homing (Problem I.1). The solutions we study are two-part strategies, composed by a learning and a control phase. In the first learning phase, the agent builds a representation of its sensorimotor cascade. Then, the agent uses the representation it has built to solve the task during the action phase.

In the learning phase, the agent randomly samples control commands $\boldsymbol{u}$ from any zero-mean distribution with positive definite covariance. Meanwhile, it estimates three quantities: the average observation at each sensel $\overline{\boldsymbol{y}}$:

$$\overline{y}^s = \mathbb{E}\{y^s\}, \tag{II.3}$$

the $(2,0)$ covariance tensor $\mathbf{P}$:

$$\mathrm{P}^{sv} = \mathrm{cov}(y^s, y^v), \tag{II.4}$$

and the $(3,0)$ tensor $\mathbf{T}$ defined by

$$\mathrm{T}^{svi} = \mathbb{E}\{(y^s - \overline{y}^s)\,\dot{y}^v u^i\}. \tag{II.5}$$

These expectations can be computed online. For example:

$$\overline{y}^s(k+1) = \frac{k}{k+1}\overline{y}^s(k) + \frac{1}{k+1}y^s(k).$$

We note that these computations can be implemented on a neural architecture; equation (II.5) is similar to three-way Hebbian learning between $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, and $\boldsymbol{u}$.

The following proposition establishes that the tensor $\mathbf{T}$, tends to approximate the tensor $\mathbf{M}$ in (II.2).

Lemma 2. *Let $\mathbf{P}$, $\mathbf{Q}$ be the covariance of $\boldsymbol{y}$ and $\boldsymbol{u}$. Then the tensor $\mathbf{T}$ tends asymptotically to:*

$$\mathrm{T}^{svi} = \mathrm{M}^s_{qj}\mathrm{P}^{qv}\mathrm{Q}^{ij}. \tag{II.6}$$

In the expression (II.6), we can observe a general pattern. Every quantity that the agent learns ultimately depends on three factors:

1) *The agent's sensorimotor cascade.* In this case, $\mathbf{M}$.
2) *The environment statistics.* In this case, the covariance $\mathbf{P}$ represents the effect of the specific environment in which

learning takes place. For example, in the case of a camera, the covariance $\mathbf{P}$ depends on the statistics of the environment texture, and it would change in different environments.
3) *The experience the agent had in such environment.* In this case, $\mathbf{Q}$ captures the kind of "training" the agent had in the environment.

In the control phase, the agent uses the learned tensors $\mathbf{T}$ and $\mathbf{P}$ to generate the commands. The following proposition is the main result of this section.

Proposition 3. *Assume that the agent is equipped with a* BDS *(Definition 1), that it has learned $\mathbf{P}$ and $\mathbf{T}$ using equations* (II.4)–(II.5)*, and $\mathbf{Q}$ is positive definite. Then the control law*

$$u^i = -(y^v - y^v_\star)^* \mathrm{T}^{svi} \mathrm{P}^{-1}_{vq} y^q \tag{II.7}$$

corresponds to a descent direction of the error metric $\|\boldsymbol{y} - \boldsymbol{y}_\star\|$. Moreover, if the operators $\{\mathrm{M}^s_{vi} y^v\}^k_{i=1}$ commute, $\boldsymbol{y}_\star$ is asymptotically stable.

Remark 4. It is a classic result [13] that, if a system such as (II.2) is nonholonomic, there exists no smooth controller that stabilizes $\boldsymbol{y}_\star$ asymptotically. In particular (II.7) is smooth in $\boldsymbol{y}$, therefore it cannot work in the nonholonomic case. Instead, the requirement that the operators $\{\mathrm{M}^s_{vi} y^v\}^k_{i=1}$ commute is a technical necessity for having a compact proof and can probably be relaxed.

The bootstrapping strategy has a couple of interesting properties: the tensors $\mathbf{T}$ and $\mathbf{P}$ can be learned using simple Hebbian learning, and the resulting control strategy is a bilinear form of the observations $\boldsymbol{y}$ and the error $\boldsymbol{y} - \boldsymbol{y}_\star$. These two properties allow a very efficient engineering implementation, and at the same time make the algorithm implementable using neural networks ("bioplausible"). The only operation that is not bioplausible is computing $\mathbf{P}^{-1}$. This motivates looking for ways to get around such computation.

Whitening the observations. If the observations had covariance equal to the identity, we could omit the term $\mathbf{P}^{-1}$. This suggests one way to proceed: find a transformation $\boldsymbol{z} = W\boldsymbol{y}$ such that $\boldsymbol{z}$ has unit covariance. In the signal processing community, the problem of finding a suitable transformation W is well known and it is called *whitening*. Numerous algorithms exist for whitening, most of them having a neural implementation [14].

Omitting "$\mathbf{P}^{-1}$" from (II.7). One could also ask whether it is possible to simply omit $\mathbf{P}^{-1}$ even when it is different from the identity. We can prove the following technical condition on $\mathbf{P}$ and $\mathbf{M}$ that makes it possible to omit the $\mathbf{P}^{-1}$ from (II.7) and still obtain a suitable minimization strategy.

Proposition 5. *Assume that $\mathbf{P}$ and $\mathbf{M}$ commute, in the sense that, defining $\mathrm{P}^q_v = \mathrm{P}^{qr}m_{rv}$, it holds that $\mathrm{M}^\cdot_{qj}\mathrm{P}^q = \mathrm{P}^\cdot_s\mathrm{M}^s_{\cdot j}$. Then the control strategy*

$$u^i = -(y^s - y^s_\star)^* \mathrm{T}^{svi} (y^v)^* \tag{II.8}$$

minimizes the error metric $V = \frac{1}{2}e^*_s \mathrm{P}^s_v e^v$, where $e^s = y^s - y^s_\star$.

We shall discuss the meaning of the commutation condition when we get to discussing the actual sensors. We will see that

in some cases the covariance $\mathbf{P}$ acts as a smoothing operation, and that the tensor $\mathbf{M}$ is similar to a gradient operation, and we will be able to interpret the condition $\mathbf{MP} = \mathbf{PM}$ in more intuitive terms as "the gradient commutes with smoothing". Also note that now the error function depends on the covariance $\mathbf{P}$; in contrast with the previous proposition, now the environment statistics influence the actions.

III. SIMULATIONS AND EXPERIMENTS

This section shows that the bootstrapping strategy seems to work for the three canonical sensors considered; the next section will justify theoretically some of these findings.

We simulate a planar omnidirectional robot controlled in velocity. The commands are $\boldsymbol{u} = (u_1, u_2, u_3) = (\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{\omega})$. In our case, simulations are precious because we can try several different sensor configurations. We simulate a 180deg range finder, an omnidirectional camera, and a field sampler, with the sensels placed on a ring. In the three cases, the observations $\boldsymbol{y}$ are, respectively, the range readings, the luminance readings, and the field intensity. The extended version reports more simulations varying the intrinsic and extrinsic sensor configuration; the results are consistent. We use a custom simulator. [1] The bootstrapping part consists in placing the robot in a randomly-generated map at random places (simulated as a uniform variable on the subset of $\mathrm{SE}(2)$ that does not intersect any obstacle), and simulate the sensor output $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$ when the robot chooses a random command $\boldsymbol{u}$ (simulated as a Gaussian random variable with spherical covariance).

We found out that, if one uses environment shapes which are too simple, the learned model will pick up the characteristic of the environment. For example, if a robot with a 360deg range-finder is always placed in a room of the same size, say 10m, it will learn that, if the readings in front measure 1m, it is likely that the readings in the back measure 9m — this knowledge is represented implicitly in the estimated covariance matrix, which will report strong negative correlation between readings in front and in the back. We do not want to see these effects, which are not representative of the real world, and to prevent them we simulate random environments composed of randomly sampled polygonal walls (the extended version has pictures of the random environments). It seems that, as long as there is some variability, the results are largely independent of the details of how the randomness is introduced. These observations motivated the definition of the environment's *symmetry group* (Definition 11) and how it affects the observation covariance.

For simulating a camera sensor, one should choose a random texture for the surfaces: for example, sample a luminance value independently for each 20cm section of the surface, and smooth the result. Choosing structured inputs such as sinusoids introduces unwanted correlation. This motivated the definition of *monotone environment* (Definition 15).

For the field sampler, we simulated a random distribution of point sources with quadratic decay.

Figures 2 onward show the learned tensors. In all figures, blue means negative and red positive; each figure is normal-ized independently from the others. Subfigures *b–c* show the covariance ($\mathbf{P}$) and information ($\mathbf{P}^{-1}$) tensors of the simulated sensors. These tensors are given as a function of the sensels position s, v. In these three simulated agents, where the sensels are disposed on a semicircle, we let s, v be the angle with respect to the robot front. The covariance encodes information on the sensel topology. For the camera and range-finder, the covariance is sparse and local: only sensels that are very close to each other are correlated, and the correlation is a function of the sensels distance. The covariance/information matrix act very similarly to convolution/deconvolution operators.

Subfigures *d–f* show the learned tensor $\mathbf{T}$. If there are n sensels and 3 commands, then $\mathbf{T}$ is a $n \times n \times 3$ tensor. We show the 3 bidimensional slices $\mathrm{T}^{sv(1)}$, $\mathrm{T}^{sv(2)}$, $\mathrm{T}^{sv(3)}$. For example, the slice $\mathrm{T}^{sv(1)}$ describes how the linear velocity $\boldsymbol{v}_x$ is related to $y(s)$ and $\dot{y}(v)$. Subfigures *g-i* show the normalized tensor $\mathbf{TP}^{-1}$ used in the control law (II.7). While $\mathbf{T}$ depends on the environment (because of the internal dependence on the covariance $\mathbf{P}$, which depends on the environment statistics), $\mathbf{TP}^{-1}$ cancels out the environmental contribution. Perhaps the results for range-finder and camera are easier to interpret. If one imagines the slices T^{svi} as linear operators that, applied to $\boldsymbol{y}$, give $\dot{\boldsymbol{y}}$, it is evident that the agent learns local spatial gradients; we shall see this theoretically.

We also tried the learning algorithm with real range-finder data from the Rawseeds project[2] [15], and we obtained similar results, which convinced us of the validity of the simulations. In this case, rather than static pictures, we give links to videos that show the learning in real time. `video:LaserDisplay`[3] shows the observations, which consists in the data from 2 Sick range-finders spliced together. `video:LaserBDSLearning`[4] shows the evolution of the tensor $\mathbf{T}$, which is qualitatively similar to the simulation results. `video:LaserCorr`[5] shows the evolution of the covariance $\mathbf{P}$: due to the fact that the robot visits regular, structured environments (e.g., corridors), the variance is different at each sensel, and the correlation is not only a function of the sensels distance (see [16] for analogous conclusions for camera data).

Figures 5 onward show the convergence properties of the control law (II.7) and the simplified control law (II.8), in simulation. We are interested in evaluating the radius of convergence. We sample numerous environments and goal configurations; then we compute the control law in a volume around the goal configuration. We show the results as "success maps": we slice the $\boldsymbol{q} = (x, y, \theta)$ volume at the three planes (x, y), (x, θ), (θ, y), and we count the percentage of times the control law pointed the robot in the right direction, meaning that it would have decreased the distance to the goal[6]. Light green means $> 99\%$; see the caption for the other values. For the field sampler and camera (Fig. 5-6), we can see that the control law (II.7) gives local convergence, while the simplified

---

[1] The Python/C++ source code is available at http://purl.org/censi/2010/boot.

[2] Data available at http://www.rawseeds.org.

[3] http://purl.org/censi/2010/be#LaserDisplay

[4] http://purl.org/censi/2010/be#LaserBDSLearning

[5] http://purl.org/censi/2010/be#LaserCorr

[6] In formulas: let the goal be $\boldsymbol{q} = (x, y, \theta) = (0, 0, 0)$. Then a "successful" command is one for which $d\|\boldsymbol{q}\|/dt \propto xu_1 + yu_2 + \theta u_3 < 0$. Note that here we treat $\mathrm{SE}(2)$ locally as a subset of $\mathbb{R}^3$.

(a) Robot   (b) P^{sv} (c) P^{-1}_{sv} (d) $\mathrm{T}^{sv(1)} - \boldsymbol{v}_x$ (e) $\mathrm{T}^{sv(2)} - \boldsymbol{v}_y$ (f) $\mathrm{T}^{sv(3)} - \boldsymbol{\omega}$ (g)$(\mathrm{TP}^{-1})^{s(1)}_v - \boldsymbol{v}_x$ (h) $(\mathrm{TP}^{-1})^{s(2)}_v - \boldsymbol{v}_y$ (i) $(\mathrm{TP}^{-1})^{s(3)}_v - \omega$
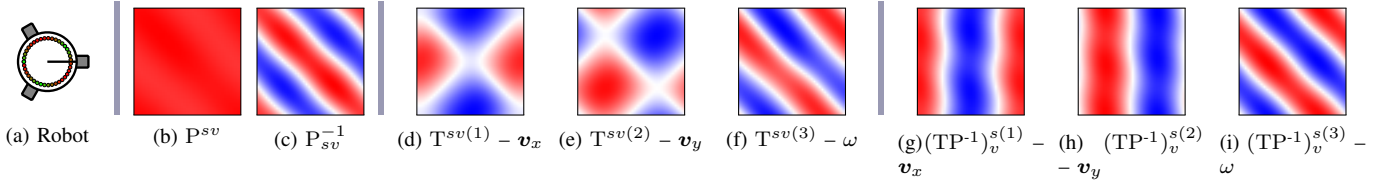
Figure 2. Tensors learned for robot with a field sampler, with sensels placed on a 360deg ring. Each axis corresponds to an angle on the ring. Red means positive; blue negative; white zero. Subfigures b–c show correlation and information matrix, as a function of the sensel positions $s, v \in \mathcal{Y}$, which can be thought of as the angle on the sensor ring. The correlation is almost identically 1; this depends on the statistics of the field we simulated. Figure d–f show the three slices of the learned tensor $\mathbf{T}$, for the three commands $(\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{\omega})$. The tensor element T^{isv} represents the interaction between the i-th command, the observation y^s and the derivative $\dot{y}^v$. Figures g–i show the 3 slices of the normalized tensor $\mathbf{TP}^{-1}$. (Images best seen in color.)

(a) Robot   (b) P^{sv} (c) P^{-1}_{sv} (d) $\mathrm{T}^{sv(1)} - \boldsymbol{v}_x$ (e) $\mathrm{T}^{sv(2)} - \boldsymbol{v}_y$ (f) $\mathrm{T}^{sv(3)} - \omega$ (g)$(\mathrm{TP}^{-1})^{s(1)}_v - \boldsymbol{v}_x$ (h) $(\mathrm{TP}^{-1})^{s(2)}_v - \boldsymbol{v}_y$ (i) $(\mathrm{TP}^{-1})^{s(3)}_v - \omega$
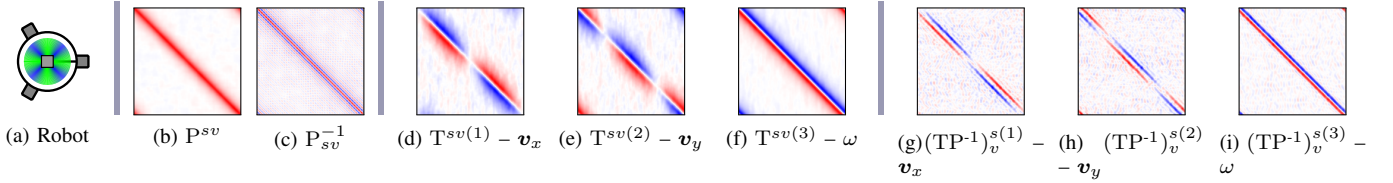
Figure 3. Tensors learned for robot with omnidirectional camera. See Fig. 2 for a general description of the figures. All pictures are a functions of two sensels $s, v \in \mathcal{Y}$, which corresponds to the pixel orientation. In b–c, we can see that the covariance matrix of a camera is sparse and local: only nearby sensels interact. Figures d–f show the learned tensor $\mathbf{T}$; if one interprets each slice as a linear operator that, applied to $\boldsymbol{y}$, gives $\dot{\boldsymbol{y}}$, it is clear that all three represents functions of the gradient of $\boldsymbol{y}$. Depending on the particular environment, the gradient is more or less smoothed by the covariance. This is confirmed theoretically – see Fig. 8. Figures g–i represent the slices of the normalized tensor $\mathbf{TP}^{-1}$: here the effect of the environment statistics are factored away and an even more local operator is obtained.

(a) Robot   (b) P^{sv} (c) P^{-1}_{sv} (d) $\mathrm{T}^{sv(1)} - \boldsymbol{v}_x$ (e) $\mathrm{T}^{sv(2)} - \boldsymbol{v}_y$ (f) $\mathrm{T}^{sv(3)} - \omega$ (g)$(\mathrm{TP}^{-1})^{s(1)}_v - \boldsymbol{v}_x$ (h) $(\mathrm{TP}^{-1})^{s(2)}_v - \boldsymbol{v}_y$ (i) $(\mathrm{TP}^{-1})^{s(3)}_v - \omega$
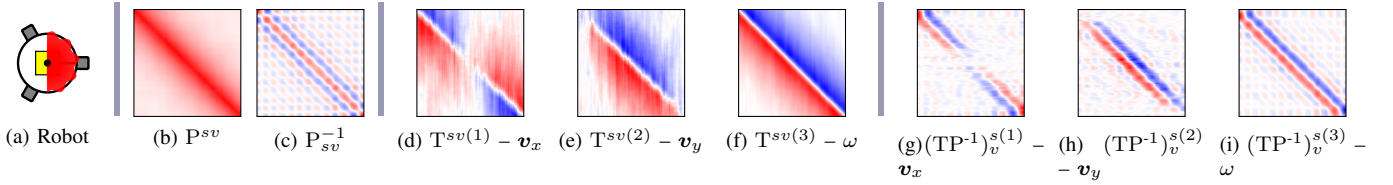
Figure 4. Tensors learned for robot with range-finder (180deg FOV). See Fig. 2 for a general description of the figures. It is interesting to compare these results with the camera results in Fig. 3. The covariance is less local: this means that, in the environment we simulated, the range readings are more correlated than the covariance; that is, they change less abruptly. As a consequence, the tensor $\mathbf{T}$ is less local than the corresponding tensor for the camera. Figures g–i show that the normalized tensor $\mathbf{TP}^{-1}$, where the effect of the environment statistics is removed, has a more local character.

convergence for (II.7) →   (a) x, y plane   (b) y, θ plane   (c) θ, x plane   convergence for (II.8) →   (d) x, y plane   (e) y, θ plane   (f) θ, x plane
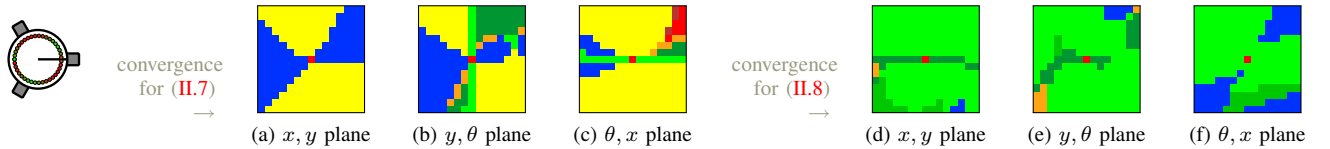
Figure 5. Statistics of the convergence of the two control laws for robot with a field sampler, with sensels placed on a 360deg ring. Figures (a)-(c) show the results for the simplified control law (II.8), the figures (d)-(f) show the results for the control law (II.7). We put the goal at the origin, and considered starting positions sampled in a 1m×1m×45deg parallelepiped around the goal. We show the convergence results along three slices in the planes x, y, y, θ, θ, x. The figures show the percentage of times (over 200 trials with random environments) that the control law indicated a direction decreasing the error metric. The color scale is: ■ 0% □ <25% ■ >25% ■ >50% ■ >75% ■ >95% ■ 100%. These figures are best seen on a computer screen.
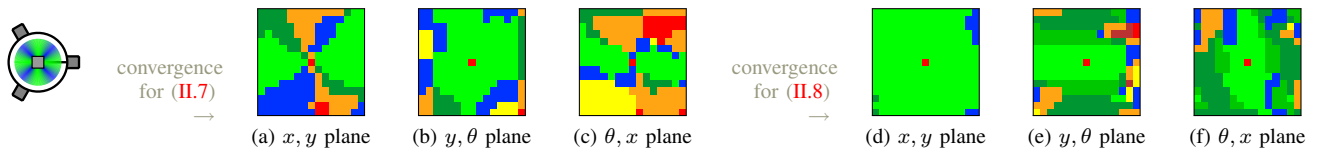

convergence for (II.7) →   (a) x, y plane   (b) y, θ plane   (c) θ, x plane   convergence for (II.8) →   (d) x, y plane   (e) y, θ plane   (f) θ, x plane

Figure 6. Convergence results for robot with omnidirectional camera. See the caption of Fig. 5 for an explanation of the color scales.

convergence for (II.7) →   (a) x, y plane   (b) y, θ plane   (c) θ, x plane   convergence for (II.7) →   (d) x, y plane   (e) y, θ plane   (f) θ, x plane
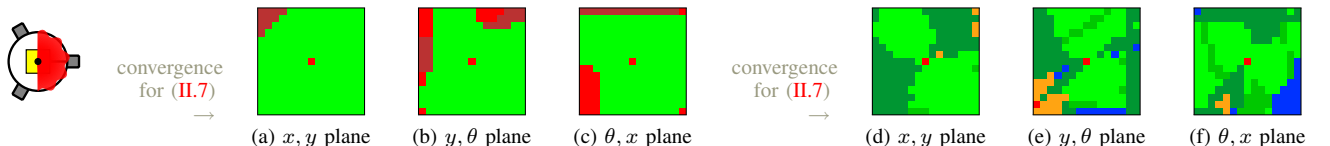
Figure 7. Convergence results for robot with range-finder (180deg FOV). See the caption of Fig. 5 for an explanation of the color scales. For the range-finder, both control laws have large convergence radius.

version (II.8) has worse performance. For the range-finder (Fig. 7), both give robust convergence.

The rest of the paper is dedicated to proving the results suggested by the simulations.

IV. ANALYSIS FOR THE THREE CANONICAL SENSORS

In the previous section, we showed by simulation that bootstrapping works for the three canonical sensors. In this section, we consider the problem of justifying this theoretically. The main results are summarized in Fig. 8. In summary, the three canonical sensors have something in common at a certain level of abstraction, as can been seen by the fact that their bootstrapped tensors are formally very similar. Because the camera and range-finder are not precisely BDS, we have to provide separate proofs of convergence. We need several preliminary results.

We start with a series of definitions and results common for all sensors. We assume the reader to be familiar with basic Lie group theory [17]. Using the standard notation, given a group $\mathcal{G}$ and two elements $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{G}$, we indicate their product as $\boldsymbol{ab} \in \mathcal{G}$, and $\boldsymbol{a}^{-1}$ is the inverse of $\boldsymbol{a}$.

Definition 6. Let $\mathcal{C}$ be the *configuration space* in which the robot moves. We assume that $\mathcal{C}$ is a subgroup of $\mathrm{SE}(3)$.

Example. Examples of subgroups of $\mathrm{SE}(3)$ are: $\mathrm{SE}(3)$ itself; $\mathrm{SE}(2)$ (planar rototranslations), $\mathrm{SO}(3)$ (pure rotations), $\mathbb{R}^3$ (pure translations).

We assume that the underlying dynamics is a rigid body controlled in velocity. For $\mathcal{C} = \mathrm{SE}(3)$ the commands $\boldsymbol{u}$ are the linear and angular velocity $\boldsymbol{v}, \boldsymbol{\omega}$. Let $\boldsymbol{t} \in \mathbb{R}^3$, $\mathbf{R} \in \mathrm{SO}(3)$ be position and attitude, and $\hat{\cdot}$ be the hat map [17]. Then the dynamics are $\dot{\boldsymbol{t}} = \mathbf{R}\boldsymbol{v}$; $\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}}$.

We already introduced the sensel space $\mathcal{Y}$, but it has not been given any structure yet. Here we characterize it by its interaction with $\mathcal{C}$.

Definition 7. Let $\mathcal{Y}$ be the *sensel space*. We assume that it is a metric space, where for two sensel positions $s_1, s_2 \in \mathcal{Y}$, the function $d(s_1, s_2)$ indicates the distances between the positions. We also assume that there is a left action of $\mathcal{C}$ on $\mathcal{Y}$. In particular, for every $\boldsymbol{q} \in \mathcal{C}$ and $s \in \mathcal{Y}$, we can define the element $\boldsymbol{q}s \in \mathcal{Y}$, and $\boldsymbol{q}_1(\boldsymbol{q}_2 s) = (\boldsymbol{q}_1 \boldsymbol{q}_2)s$.

Example. For example, for a pan-tilt-roll "robotic" camera, $\mathcal{Y} = \mathbb{S}^2$, $\mathcal{C} = \mathrm{SO}(3)$, and the action $\boldsymbol{q}s$ corresponds to applying the rotation $\boldsymbol{q}$ to $s \in \mathbb{S}^2$.

Finally, we have to give some structure to the world around the robot. "World" is everything needed to compute the sensor output, apart from the robot pose. For a range-finder, the world includes the 3D environment structure; for a camera, it includes the texture, reflectance, and illumination information as well. We use a construction typical of stochastic geometry [18]: we assume that the set of worlds $\mathcal{W}$ can be factorized into a "shape" and "pose" component, in the sense that, for each world, there are many others that share the same shape (including color, texture, etc.), but rototranslated. Therefore, we let $\mathcal{W} \equiv \mathcal{S} \times \mathrm{SE}(3)$, where $\mathcal{S}$ is called *shape*

space. We write an element of $\mathcal{W}$ as a tuple $\langle \boldsymbol{s}, \boldsymbol{p} \rangle$, with $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{p} \in \mathrm{SE}(3)$.

All three sensors are "relative", in the following sense.

Definition 8. Given a sensel space $\mathcal{Y}$, a pose space $\mathcal{C}$, and a shape-pose space $\mathcal{W} \equiv \mathcal{S} \times \mathrm{SE}(3)$, the map $y : \mathcal{W} \times \mathcal{C} \times \mathcal{Y} \to \mathcal{O}$ corresponds to a *relative sensor* if the following two properties hold for all $x \in \mathcal{C}$.

$$y(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q}, s) = y(\langle \boldsymbol{s}, x\boldsymbol{p} \rangle, x\boldsymbol{q}, s) \quad \text{[P1]} \quad \text{(IV.9)}$$

$$y(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q}, s) = y(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q}x^{-1}, xs) \quad \text{[P2]} \quad \text{(IV.10)}$$

Remark 9. Property P1 corresponds to the fact that there is an intrinsic ambiguity in choosing the frame of reference. The world and the robot have both a pose with respect to some fixed coordinate frame, but the output of the sensor depends only of the *relative* pose $\boldsymbol{q}^{-1}\boldsymbol{p}$ (let $x = \boldsymbol{q}^{-1}$ in (IV.9) to see this fact). Property P2 describes the fact that the robot is "carrying" the sensor: ultimately the output at sensel s depends only on $\boldsymbol{q}s$, therefore it is invariant if we apply x to s and multiply $\boldsymbol{q}$ by x on the right.

The bootstrapping strategy proposed in Section II, specifically equations (II.3)-(II.5), is described by means of statistical operators such as expectations. To predict the outcome, we have to specify something about the probability distribution of the stimuli that the agent experienced. Firstly, we give it a name.

Definition 10. Let $p_{\mathrm{T}}(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q})$ be the *training probability distribution* of worlds/poses that the agent has experienced during its bootstrapping phase.

We want the training distribution to have some regularity. We describe the regularity with the language of Lie groups.

Definition 11. Define the set *symmetry group* of p_{T} as the subgroup Sym of $\mathcal{C}$ such that, for all $x \in \mathrm{Sym}$, $p_{\mathrm{T}}(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q}) = p_{\mathrm{T}}(\langle \boldsymbol{s}, x\boldsymbol{p} \rangle, \boldsymbol{q})$.

Example. Consider a planar robot ($\mathcal{C} = \mathrm{SE}(2)$), and assume we believe that, at the end of bootstrapping, the robot experience did not privilege one particular orientation over the others. Then we would set Sym to be the group of planar rotations.

At this point we are ready to give a technical definition and relative proposition, on which many other results are based.

Definition 12. Consider two couples of sensels (s_1, v_1), (s_2, v_2), where $s_1, v_1, s_2, v_2 \in \mathcal{Y}$. We call the training distribution *mixing* if $d(s_1, v_1) = d(s_2, v_2)$ implies that there exists a $x \in \mathrm{Sym}$ such that $(s_2, v_2) = (xs_1, xv_1)$.

Proposition 13. *For a mixing training distribution, the expectation of any function of two sensels $s, v \in \mathcal{Y}$ is only a function of their distance; for all functions ϕ, we can write $\mathbb{E}\{\phi(y^s, y^v)\}$ as $\varphi(d(s, v))$ for some other function φ.*

Corollary 14. For a relative sensor in the mixing case, the covariance of two sensels is a function of only their distance: $\mathrm{cov}(y^s, y^v) = f(d(v, s))$.

We define a property of the environment useful in the future.

	Pure BDS?	$\mathcal{Y}$	Sensor dynamics	Bootstrapped tensors	Convergence proof for (II.7)	Convergence proof for (II.8)
Field sampler	Yes.	$\mathbb{R}^3$	$\dot{y}^s = \nabla_i y^s \boldsymbol{v}^i$ $+ (s \times \nabla y^s)_i \boldsymbol{\omega}^i$	$\mathrm{T}^{svi} = \nabla_j^{\mathbb{R}^3} \mathrm{P}^{sv} \mathrm{Q}^{ij}$ $\mathrm{T}^{sv(i+3)} = (s \times \nabla^{\mathbb{R}^3} \mathrm{P}^{sv})_j \mathrm{Q}^{ij}$	Yes.	Yes for translation.
Camera	No. (hidden state)	$\mathbb{R}^3 \times \mathbb{S}^2$	$\dot{y}^s = \mu^s \nabla_i y^s \boldsymbol{v}^i$ $+ (s \times \nabla y^s)_i \boldsymbol{\omega}^i$	$\mathrm{T}^{svi} = \overline{\mu}^s \nabla_j^{\mathbb{S}^2} \mathrm{P}^{sv} \mathrm{Q}^{ij}$ $\mathrm{T}^{sv(i+3)} = (s \times \nabla^{\mathbb{S}^2} \mathrm{P}^{sv})_j \mathrm{Q}^{ij}$	Yes.	Yes for rotation.
Range-finder	No. (nonlinearity)	$\mathbb{R}^3 \times \mathbb{S}^2$	$\dot{y}^s = (\nabla_i \log y^s - s_i^*) \boldsymbol{v}^i$ $+ (s \times \nabla y^s)_i \boldsymbol{\omega}^i$	$\mathrm{T}^{svi} = \nabla_j^{\mathbb{S}^2} \beta(\mathrm{P}^{sv}) \mathrm{Q}^{ij}$ $\mathrm{T}^{sv(i+3)} = (s \times \nabla^{\mathbb{S}^2} \mathrm{P}^{sv})_j \mathrm{Q}^{ij}$	Yes for rotation.	Yes for rotation.

Figure 8. Summary of the results in this section. The third column shows the equation for the sensor dynamics: as one can see, these three apparently different sensors have a somewhat similar dynamics. The next column shows the tensor learned; for the camera and range-finder, which are not exactly BDS, these tensors could be considered a "projection" of the nonlinear dynamics to the BDS space. The last two columns indicates whether we have a formal proof that the general control laws for BDS work for the particular sensors—see the text for details. The tensor $\mathbf{P}$ is the covariance of $\boldsymbol{y}$; the tensor $\mathbf{Q}$ is the covariance of $\boldsymbol{u}$; $\overline{\mu}^s$ is the average nearness (inverse of distance) in direction s.

Definition 15. We call the environment *monotone* if the covariance of the values of two sensels is a monotone function of the distance between the sensels[7].

4.1 Analysis of field sampler: We start with our definition of a field sampler.

Definition 16. Let the sensels space be $\mathcal{Y} = \mathbb{R}^3$. The sensor $\boldsymbol{y}$ is a field sampler if there exists a field $\mathcal{H} : \mathbb{R}^3 \to \mathbb{R}$ such that $y^s = \mathcal{H}(\boldsymbol{t} + \mathbf{R}s)$, where $\boldsymbol{t} \in \mathbb{R}^3$ and $\mathbf{R} \in \mathrm{SO}(3)$ are the agent position and attitude.

We can show that a field tensor is indeed a perfect BDS, and therefore all the relevant results from Section II apply.

Proposition 17. *A field sampler is a* BDS, *because its observations dynamics are bilinear in $\boldsymbol{y}$ and $\boldsymbol{u} = (\boldsymbol{v}, \boldsymbol{\omega})$:*

$$\dot{y}^s = (\nabla_i y^s) \boldsymbol{v}^i + (s \times \nabla y^s)_i \boldsymbol{\omega}^i.$$

We can compute the exact form for the learned tensor $\mathbf{T}$. Assuming the general case of a fully actuated rigid body in $\mathrm{SE}(3)$, the tensor $\mathbf{T}$ has 6 components for the last index. The first three ($1 \leq i \leq 3$) correspond to linear velocity, and the last three ($4 \leq i \leq 6$) to the angular velocity.

Proposition 18. *The learned tensor for a field sampler i*s

$$\mathrm{T}^{svi} = \nabla_j \mathrm{P}^{sv} \mathrm{Q}^{ij}, \quad (IV.11)$$
$$\mathrm{T}^{sv(i+3)} = (s \times \nabla \mathrm{P}^{sv})_j \mathrm{Q}^{ij}. \quad (IV.12)$$

Proof: We show the computation for the linear velocity components:

$$\begin{aligned} \mathrm{T}^{svi} &\triangleq \mathbb{E}\{(y^s - \overline{y}^s)\dot{y}^v \boldsymbol{v}^i\} = \mathbb{E}\{(y^s - \overline{y}^s)(\nabla_j y^v)\boldsymbol{v}^j \boldsymbol{v}^i\} \\ &= \nabla_j \mathbb{E}\{(y^s - \overline{y}^s) y^v\} \mathbb{E}\{\boldsymbol{v}^j \boldsymbol{v}^i\} = \nabla_j \mathrm{P}^{sv} \mathrm{Q}^{ij}. \end{aligned}$$

The formula for the others is obtained similarly. ∎

Proposition 19. *For a mixing training distribution, the condition of Proposition 5 are satisfied for pure translation ($\mathcal{C} = \mathbb{R}^3$), and hence the simplified control (II.8) can be used.*

Proof: Proposition 5 requires us to verify that we can interchange the order of ∇ and $\mathbf{P}$ in (IV.11). Because the environment is mixing, we know from Proposition 13 that the covariance can be written as a function of the sensel distance: $\mathrm{P}^{sv} = f(d(s,v))$. Therefore, when $\mathbf{P}$ is used as a linear operator, it corresponds to smoothing with a radially

[7]Not all environments are monotone; see [12] for a counterexample.

symmetric function. Noting that in $\mathbb{R}^3$ gradient and radially symmetric smoothing commute concludes the proof. ∎

In conclusion, a field sampler is a sensor whose dynamics is precisely that of a BDS, and therefore, the analysis is quick and compact.

4.2 Analysis of camera: In general, the sensel space of a camera is $\mathcal{Y} = \mathbb{R}^3 \times \mathbb{S}^2$: each pixel captures the light arriving to a particular focus point (in $\mathbb{R}^3$) from a particular direction on the unit sphere ($\mathbb{S}^2$). For simplicity, we consider a central camera with only one focus point, so that the sensel space is just $\mathcal{Y} = \{\mathbf{0}\} \times \mathbb{S}^2$.

Proposition 20. *Let y^s, $s \in \mathbb{S}^2$, be the luminance signal captured by the camera. Let μ^s be the nearness, the inverse of the distance in direction s. Then the dynamics of $\boldsymbol{y}$ are*

$$\dot{y}^s = \mu^s \nabla_i y^s \boldsymbol{v}^i + (s \times \nabla y^s)_i \boldsymbol{\omega}^i. \quad (IV.13)$$

See [19] for a proof. From this it follows that a camera is a BDS only for pure rotation, or with the environment at infinity, because of the dependence on the hidden state μ.

However, we can show that a useful BDS approximation can be learned. What happens is that, when learning the observation dynamics, the hidden state μ^s gets filtered out and appears only as a multiplicative factor in the BDS tensor.

Proposition 21. *Assuming that nearness and luminance are independent in p_T, the learned tensors for a camera ar*e

$$\mathrm{T}^{svi} = \overline{\mu}^s \nabla_j \mathrm{P}^{sv} \mathrm{Q}^{ij}, \quad (IV.14)$$
$$\mathrm{T}^{sv(i+3)} = (s \times \nabla \mathrm{P}^{sv})_j \mathrm{Q}^{ij}.$$

Because the camera is not a BDS, we cannot use the stock results for convergence. However, the control law (II.7), when instantiated for the camera, has the same form as the one we studied in our previous work on bioplausible visual control [19]. Referring to those results, we can say that (II.7) locally converges. As for the convergence of the simplified control (II.8), we can prove the analogous of Proposition 19, this time for rotation rather than translation.

Proposition 22. *In a mixing environment, ignoring the conditions at the borders of the sensels area, the condition in Proposition 5 is satisfied for rotations ($\mathcal{C} = \mathrm{SO}(3)$), and hence the simplified control law (II.8) can be used.*

Proof: (sketch) The proof is based on interpreting the covariance operator as a convolution operator on the sphere,

proving that it is self-adjoint, and exploiting its commutation properties with rotation. ∎

4.3 Analysis for range-finder: Each reading of a range-finder measures the distance from a an origin point (in $\mathbb{R}^3$) to the closest obstacle in a certain direction (in $\mathbb{S}^2$). Like the camera, in principle, the sensel space for a range-finder is $\mathcal{Y} = \mathbb{R}^3 \times \mathbb{S}^2$, but for simplicity we consider the case where all rays have the same origin. We start by providing an expression for the observation dynamics—even though range-finders are popular sensors, we could not find this in the published literature.

Proposition 23. *Let y^s be the range reading (distance to the obstacle in direction s). Then the dynamics of a range-finder are*

$$\dot{y}^s = (\nabla_i \log y^s - s_i^*) \boldsymbol{v}^i + (s \times \nabla y^s)_i \, \boldsymbol{\omega}^i. \qquad \text{(IV.15)}$$

Note that the rotational part is exactly the same as the camera model (IV.13), because rotation has the same effect on range and luminance data. The "$-s_i^*$" term means that if the velocity $\boldsymbol{v}$ is in the direction on s, then the range decreases (the remaining nonlinear term $\nabla_i \log \sigma^s$ is less intuitive).

We can prove the following regarding the bootstrapping strategy.

Proposition 24. *If the training distribution is mixing (Definition 12) and the environment is monotone (Definition 15) the learned tensors for a range-finder ar*e

$$\begin{aligned} \mathrm{T}^{svi} &= \nabla_j \beta(\mathrm{P}^{sv}) \mathrm{Q}^{ij}, \qquad \text{(IV.16)} \\ \mathrm{T}^{sv(i+3)} &= (s \times \nabla \mathrm{P}^{sv})_j \, \mathrm{Q}^{ij}, \end{aligned}$$

where $\beta(\mathrm{P}^{sv})$ is an element-wise scalar function of P^{sv}.

Because range-finders and cameras are equivalent under pure rotations, it is immediate to show convergence of (II.7) in that case. In particular, the equivalent of Proposition 22 holds. It is instead challenging to prove convergence of (II.7) for translation. The main reason is that $\mathbf{P}^{-1}$ does not cancel the term $\beta(\mathbf{P})$ in (IV.16), due to the nonlinearity of β.

V. CONCLUSIONS AND FUTURE WORK

This paper presents a contribution to the vast problem of bootstrapping, which consists in estimating and using models of a sensorimotor cascade, starting from uninterpreted commands and observations. We have shown that the abstraction of bilinear dynamics sensors (BDS) is general yet powerful enough to represent the main phenomena of a representative selection of robotics sensors (field samplers, cameras, and range-finders).

To the best of our knowledge, this is the first presentation of a bootstrapping agent that can provably learn to use a variety of sensors to solve the same cross-modality task. The algorithm is also simple, consisting only of a few lines of code; it is fast, being extremely parallelizable. With respect to the approach of Kuipers and his group, we focused on a more "continuous" rather than a "symbolic" solution, and this made it possible to actually prove strong results concerning the convergence of the learning process and the control law. The extended version

of this work contains further development of the theory and an example application to real world robots.

So far we have been focusing more on the sensors rather than actuators, as we only considered the case of fully-actuated velocity control. Previous work [19] suggests that control in velocity can be extended to control in forces/torques with relatively little effort. Instead, it seems more challenging learning a hidden state and its dynamics. This would be useful in the case of a camera, where the observation dynamics depends on the nearness. The other challenge is learning a more nonlinear model (perhaps by learning additional levels of Taylor approximations after the bilinear terms); this would be useful in the case of the range-finder, which is not a pure BDS for its nonlinearity.

REFERENCES

[1] Cohen and *et al.*, "Functional relevance of cross-modal plasticity in blind humans," *Nature*, vol. 389, no. 6647, pp. 180–183, 1997.
[2] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey," *Connection Science*, vol. 15, pp. 151–190, 2003.
[3] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: A survey," *IEEE Trans. on Autonomous Mental Development*, vol. 1, no. 1, pp. 12 –34, 2009.
[4] B. Kuipers, "An intellectual history of the Spatial Semantic Hierarchy," *Robotics and cognitive approaches to spatial mapping*, vol. 38, pp. 243–264, 2008.
[5] J. Stober, L. Fishgold, and B. Kuipers, "Learning the sensorimotor structure of the foveated retina," in *Proceedings of the Ninth International Conference on Epigenetic Robotics People*, 2009.
[6] J. Stober, L. Fishgold, and B. Kuipers, "Sensor map discovery for developing robots," in *AAAI Fall Symposia Series: Manifold Learning and Its Applications*, 2009.
[7] D. George and J. Hawkins, "Towards a mathematical theory of cortical micro-circuits," *PLoS Comput Biol*, vol. 5, p. e1000532, 10 2009.
[8] M. Hutter, *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Berlin: Springer, 2004.
[9] B. Kuipers, "The Spatial Semantic Hierarchy," *Artificial Intelligence*, vol. 119, no. 1-2, 2000.
[10] T. Lochmatter and A. Martinoli, "Theoretical analysis of three bio-inspired plume tracking algorithms," in *Proceedings of the IEEE Conference on Robotics and Automation*, 2009.
[11] J.-S. Gutmann, G. Brisson, E. Eade, P. Fong, and M. Munich, "Vector field slam," in *Proceedings of the IEEE Conference on Robotics and Automation*, 2010.
[12] A. Censi and R. M. Murray, "Bootstrapping multilinear models for robotic sensorimotor cascades," Tech. Rep. CaltechCDSTR:2010.003, California Institute of Technology, Pasadena, CA, 2010. Available at http://purl.org/censi/2010/boot .
[13] R. W. Brockett, "Asymptotic stability and feedback stabilization," in *Differential Geometric Control Theory* (R. S. M. R. W. Brockett and H. J. Sussmann, eds.), pp. 181–191, Boston: Birkhauser, 1983.
[14] S. C. Douglas and A. Cichocki, "Neural networks for blind decorrelation of signals," *IEEE Trans. on Signal Processing*, vol. 45, no. 11, pp. 2829 –2842, 1997.
[15] Ceriani and *et al.*, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Auton. Robots*, vol. 27, no. 4, pp. 353–371, 2009.
[16] E. Grossmann, J. A. Gaspar, and F. Orabona, "Discrete camera calibration from pixel streams," *Computer Vision and Image Understanding*, vol. 114, no. 2, pp. 198 – 209, 2010.
[17] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. Berlin: Springer-Verlag, 1999.
[18] H. Le and D. G. Kendall, "The Riemannian structure of Euclidean shape spaces: A novel environment for statistics," *Annals of Statistics*, vol. 21, no. 3, pp. 1225–1271, 1993.
[19] S. Han, A. Censi, A. D. Straw, and R. M. Murray, "A bio-plausible design for visual pose stabilization," Tech. Rep. CaltechCDSTR:2010.001, California Institute of Technology, 2010. (a reduced version to appear in IROS'10).