Submitted, International Journal of Robotics Research (IJRR), Dec 2013
http://www.cds.caltech.edu/~murray/papers/cm13-ijrr.html

Bootstrapping bilinear models of simple Vehicles

Andrea Censi Richard M. Murray

***Abstract*—Learning and adaptivity will play a large role in robotics in the future, as robots move from structured to unstructured environments that cannot be fully predicted or understood by the designer. Two questions that are open: 1) in principle, how much it is possible to learn; and, 2) in practice, how much we should learn. The *bootstrapping* scenario describes the extremum case where agents need to learn "everything" from scratch, including a torque-to-pixels models for its robotic body. Systems with such capabilities will be advantaged in terms of being resilient to unforeseen changes and deviations from prior assumptions. This paper considers the bootstrapping problem for a subset of the set of all robots: the Vehicles, inspired by Braitenberg's work, are idealization of mobile robots equipped with a set of "canonical" exteroceptive sensors (camera; range-finder; field-sampler). Their *sensel*-level dynamics are derived and shown to be surprising close. We define the class of BDS models, which assume an instantaneous bilinear dynamics between observations and commands, and derive streaming-based bilinear strategies for them. We show in what sense the BDS dynamics approximates the set of Vehicles to guarantee success in the task of generalized servoing: driving the observations to a given goal snapshot. Simulations and experiments substantiate the theoretical results. This is the first instance of a bootstrapping agent that can learn the dynamics of a relatively large universe of systems, and use the models to solve well-defined tasks, with no parameter tuning or hand-designed features.**

I. Introduction

The discipline of robotics collects the heritage of fields such as mechanical engineering, computer science, and control theory, whose basic methodology is based on explicit modeling and systematic design procedures. However, as robots are equipped with richer sensors, inhabit populated unstructured environments, and perform more complex behaviors, explicit modeling and design is too costly, thus motivating the use of adaptive and learning systems. While it is clear that *some* learning is needed, two open questions are: 1) how much learning is possible, in principle, i.e. how little prior knowledge can we get away with; and 2) how much learning is desirable. The latter question is easier to answer, the answer being: it depends. Learning is a trade-off of performance (running cost) vs prior knowledge (design cost). The main challenge for learning in robotics is guaranteeing safety and performance [1], in contrast with other industrial applications of learning techniques (an "error" of a recommender system is only an inopportune suggestion).

The question of how much learning is possible requires more qualification. Biology gives us a proof of existence that, in some sense, "everything" can be learned in relatively large domains. Most cognitive-level processing in the brain happens in the neocortex, a six-layered sheet of uniform neurons, initially

A. Censi is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA. E-mail: censi@mit.edu. R M. Murray is with the Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA. E-mail: murray@cds.caltech.edu.

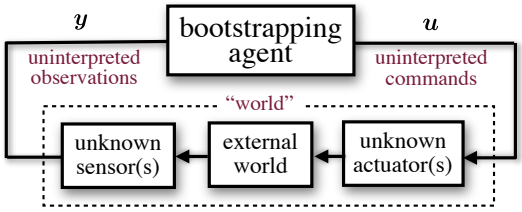


Figure 1. For a bootstrapping agent connected to an unknown body the "world" is the series of the unknown actuators, the external world, and the unknown sensors.

"blank" and eventually adapted during development to work with different sensor modalities (the visual cortex is repurposed to process tactile information in blind subjects [2]; completely new modalities can be learned as well [3]). Reproducing the same adaptability and generality in artificial systems will make us able to create more reliable robots.

A concrete formalization of the "learn everything" problem has been proposed by Kuipers and colleagues [4, 5]. In the *bootstrapping* scenario an agent starts its life embodied in a robotic body with no prior information about its sensors and actuators: the external *world* is the series of the unknown actuators, the external environment, and the unknown sensors (Fig. 1). The agent has access to a stream of uninterpreted observations and commands, with no associated semantics. The central problem of bootstrapping is obtaining a predictive model of the sensorimotor cascade and using it to perform useful tasks. Bootstrapping can be seen as an extreme form of system identification/calibration. Calibration techniques for robotics assume to know the type of sensors/actuators being calibrated and can only estimate a limited number of parameters in a known family of models. Can an agent learn to use *unknown* sensors and actuators? Can the *same* learning algorithm work for a range-finder and a camera? These are, at the moment, open questions. It would be extremely convenient if we could just attach any sensor to a robot, and the robot could learn how to use it, without tedious programming.

In research towards learning there is a trade-off of results *strength vs generality*: on one extremum, system identification has a complete, rigorous theory for learning with relatively small class of systems (e.g., *linear, time invariant* dynamical systems); on the other extremum, *deep learning* [6] provides generic tools with the ambition of learning *everything*, but essentially no formal results. One of cornerstones of learning theory is that learning "everything" is impossible, as the "free lunch" theorems show; therefore, it is important to properly define the assumptions of the methods. Our work formalizes the problem of bootstrapping for the Vehicles universe. We apply a formal control-theoretic approach giving precise statements about the conditions needed for successfully learning the dynamics of the Vehicles and succeeding at specific tasks.

A. Related work

The first rigorous formalization of learning is due to Vapnik [7]. In *supervised learning* the goal is to learn the relation between the two for the purpose of prediction. *Unsupervised learning* refers to techniques, such as manifold learning and autoencoders [6, Section 4.6], that learn compact representations of the data, often used as a preliminary step in supervised learning. In the *interactive* setting the agent interacts with the world; this introduces the problem of exploration and issues such as the exploration-vs-exploitation tradeoff. In the *embodied* setting the agent lives in a (robotic) body and it has to deal with high-dimensional data streams and an unobservable world too complex to be modeled exactly. There are four things that one might learn in the context of interactive agent-world settings: a) statistical regularities in the observations: what is the agent likely to experience in a particular setting; b) the input-output *dynamics* of the system: how the commands influence state and observations; c) *policies*: how to choose commands that obtain a desired behavior; d) "*values*": what are desired behaviors or states. There are thus manifold variations of the learning problem, depending on the prior knowledge assumed for each of these components.

a) Learning dynamics in computer science & control : System identification techniques [8] have been successfully deployed in industrial applications since a few decades. There is a complete theory for linear systems [9] as well as for some limited classes of nonlinear systems (e.g., linear+static nonlinearity [10], Volterra series [11]). Adaptive control [12] is concerned with what happens when closing the loop, such as guaranteeing stability/performance of a learned controller.

Most work in computer science focuses on more general models with fewer assumptions on the system dynamics. (Hidden) Markov Models and (Partially Observable) Markov Decisions Processes (MDPs) are discrete-time dynamical systems with a discrete state space that evolves according to arbitrary probabilistic transitions. Their dynamics can be learned with spectral methods [13, 14]. There are also alternative black-box-style models for dynamical systems, such as Predictive State Representations [15–17], in which the dynamics is represented by a series of *tests,* which are observable functions of the state. These methods can potentially approximate any dynamics, though they suffer from the curse of dimensionality, which makes them unsuitable to use for generic high-dimensional data streams.

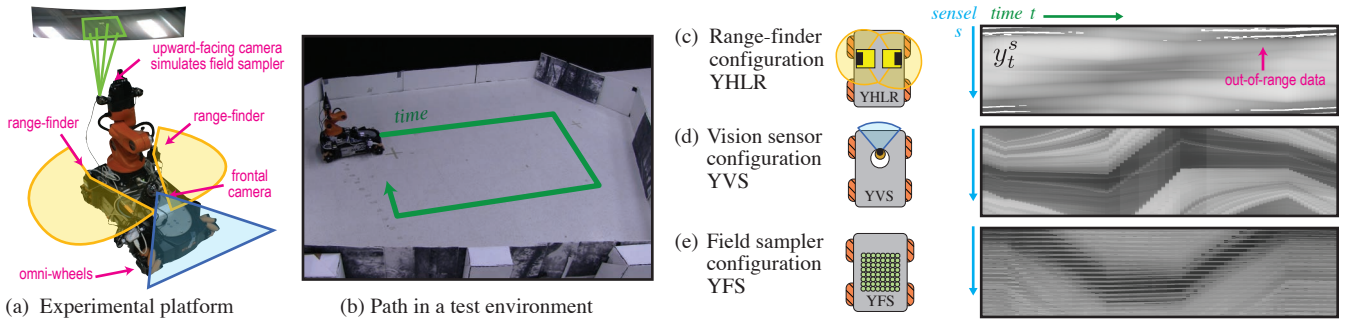
Our approach is in the spirit of system identification, in the sense that we strive to give formal results, if in ideal conditions (e.g. asymptotical results, local convergence); for some other aspects it is more inspired to computer science (e.g., the style of our definitions for constructing the set of Vehicles) or machine learning (e.g., simple algorithms on large datasets).

b) Deep belief networks: Deep belief networks (DBNs) have the same ambition of learning everything from scratch. In contrast to "classic" neural networks that are mostly understood as fitting a nonlinear function between input and output [18], the learned weights of a DBN [6, 19] encode a low-dimensional representation of a probability distribution generating the data. The computational structure of DBNs gives them some "universal approximator" property, but at this time it is not clear how the potential is limited by the approximations needed in practice (e.g., limiting the iteration of a nominally infinite MCMC process). Moreover, the investigation so far has been primarily empirical—tuning a DBN is still considered an art. While the primary domain in which DBNs are used are static images, DBNs have been used to describe dynamical data [20–22]. Extensions using three-way interactions have been used to represent motions and transformations at the pixel level [23–28]. Other extensions have been used to model the statistics of multimodal sources of data [29, 30], but not the *dynamics* of those data, as of interest here. There is no DBN-based work that considers the semantic of "observations" and "commands" and control synthesis, though it is certainly worth exploring.

c) Learning in robotics: Most recent work on learning in robotics leans towards the structured learning of policies to obtain specific behaviors, either in the unsupervised setting, for example according to the reinforcement learning framework [31, 32], or in the supervised setting, for example using apprenticeship learning [33]. In this paper we are concerned with learning *models* of the dynamics at the sensorimotor level; once we have models, policies are manually derived. In the long term, as complexity of tasks increase, it is inevitable that an agent *must* have a model of the world [34, 35].

Developmental robotics is interested specifically in the developmental point of view of an embodied agent, often informed by developmental stages observed in humans [36, 37]. For example, one of the issues is learning the "body schema" [38] for which the approach can be either parametric [39, 40] or nonparametric [41] using tools such as Gaussian processes [42, 43]. None of these techniques can deal with raw high-dimensional sensorimotor data.

Our work closely follows Kuipers' and colleagues' definition of the problem, while introducing a more rigorous mathematical formalization. Pierce and Kuipers [4] describe how an agent can build successively more abstract levels of representation for its sensors and actuators that can be used for prediction and various navigation tasks. The logical organization has been used in successive works, and also remains as a guideline in this thesis. This structure follows the Spatial Semantic Hierarchy [44–48], which describes how cognitive knowledge of space and motion can be organized four hierarchical ontology levels. At the *sensorimotor level* the agent learns the structure of the sensors; the topological or metric organization of *sensels* can be recovered by computing statistics of the sensory streams; several variations of the same idea have been developed [49–53]. Understanding the structure of the sensors is a necessary preliminary step before modeling sensorimotor interaction [54]. The agent must learn how actions influence the observations, and in particular what action have a reproducible effect. At the *control level*, the agent learns to perform simple spatial tasks, such as *homing* to a given position. Some of the developed control laws induce a discretization of the continuous sensorimotor experiences by mapping a set of initial states to a final state. This bootstraps a concept of "place", and the agent can create a topological map by mapping which control laws moves from one place to another [55]. Further work by Kuipers and colleagues focused on other ways to find a symbolization of the uninterpreted streams of data, through the abstraction of *objects* [56–58]. The ultimate goal is to have a generic bootstrapping architecture integrated with more traditional learning approaches that is able to go "from pixels to policies" [59].



(a) Experimental platform (b) Path in a test environment

Figure 2. Bootstrapping agents should be able to deal with any robotic body. Our experimental platform is designed to simulate the three "canonical sensors" described in this paper. An upward-facing camera simulates an ideal field-sampler (Definition 12); the intensity values from the central band of a front-facing camera simulates an ideal 1D vision sensor (Definition 18); finally, two Hokuyo sensors are used as representatives of ideal range finders (Definition 15).

B. Contribution and outline

This paper shows that it is possible to solve the bootstrapping problem for a relatively large subset of the robotics domain. We describe the first instance of a bootstrapping agent that can learn the dynamics of a relatively large universe of systems, and use the models to solve well-defined tasks, without manual tuning or features design, and with some theoretical guarantees.

Section II formally constructs the set of Vehicles, a subset of the "set of all robots", that are the idealization of mobile robots equipped with exteroceptive sensors. Section III describes the three "canonical" sensors that we use in the paper: field-sampler, range finder and the ideal vision sensor.

Section IV describes the family of BDS models, which assume that there is an instantaneous, *bilinear* relation between observations and commands. Section V describes a strategy for learning BDS models appropriate for a bootstrapping agent, based on computing Hebbian-style statistics.

Section VI describes in what sense the BDS family approximates the canonical exteroceptive sensor despite the fact that they do not fully capture the nonlinearities or the hidden states in the real dynamics. Section VII describes a variety of learning experiments, conducted both in simulation, where the idealized sensors are simulated exactly, as well as experiments using the real implementation of the idealized canonical sensors (Fig. 2).

Section VIII describes the problem of generalized servoing (driving the current observations to some goal observations) and how it can be solved for BDS models. Section IX proves that the servoing strategy for the BDS models works also for the dynamics of the Vehicles, under certain assumptions. Section X shows experiments for servoing and servoing-based navigation.

Finally, Section XI wraps up and describes future work.

Relation with previous work: The ideas of this paper first appeared in an ICRA paper [60], developed in several others [61–63] and eventually condensed in a dissertation [64].

Additional materials: Source code and datasets are available at http://purl.org/censi/2013/jbds. The page includes a video that shows further details about the experiments.

Notation: Our notation is standard, with a few exceptions. $\mathbb{R}_\circ^+$ are the positive reals, and $\mathbb{R}_\bullet^+ = \mathbb{R}_\circ^+ \cup \{0\}$. $\mathsf{Measures}(A)$ are all probability measures on a set A.

Given three sets $\mathcal{Y}$, $\mathcal{U}$, $\mathcal{X}$, let $\mathcal{D}(\mathcal{Y};\mathcal{X};\mathcal{U})$ be the set that contains all continuous-time control systems with observations $\boldsymbol{y}$ in $\mathcal{Y}$, state in $\boldsymbol{x} \in \mathcal{X}$, and commands $\boldsymbol{u} \in \mathcal{U}$. $\mathcal{D}(\mathcal{Y};\mathcal{U})$ is a shortcut for $\mathcal{D}(\mathcal{Y};\mathcal{Y};\mathcal{U})$. $\mathcal{D}(\mathcal{Y};\mathcal{X};\mathcal{U};\Delta)$ is the set of all *discrete-time* dynamical systems with sampling interval Δ.

II. Modeling the Vehicles universe

Consider the set of all robots Robots, which contains all possible systems obtained by pairing every combination of robotic actuators and robotic sensors. The goal of bootstrapping is to create agents that can work with any system in Robots. As a starting point, this paper considers a smaller set, the *Idealized Vehicles* universe, inspired by Braitenberg's work *Vehicles* [65]. The set Vehicles contains particular simple robots, which have the kinematics of simple vehicles and are equipped with any combination of a set of "canonical" exteroceptive sensors. An element of Vehicles is defined by describing its body, composed by a particular kinematics and sensors, and then placed in an "environment":

$$\mathsf{Robots} \ni \mathsf{vehicle} = \overbrace{\mathsf{sensors} + \mathsf{kinematics}}^{\text{vehicle body}} + \mathsf{environment}$$

It is perhaps unusual to put the environment as part of the "robot", yet this makes sense in the context of learning: can a robot learn it has a camera if it spends its life in the dark?

1) Vehicle kinematics: The kinematics of a rigid body in $\mathrm{SE}(3)$ can be written as a function of the position $\boldsymbol{t} \in \mathbb{R}^3$ and the attitude $\mathbf{R} \in \mathrm{SO}(3)$. The linear velocity $\boldsymbol{v} \in \mathbb{R}^3$ is a three-dimensional vector, and it is expressed in the body frame. The angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ is also a three-dimensional vector giving the instantaneous angular velocities around the three principal axes in the body frame. Using the *hat map*, the vector $\boldsymbol{\omega}$ is mapped to an antisymmetric matrix $\hat{\boldsymbol{\omega}} \in \mathsf{so}(3)$ such that $\boldsymbol{\omega} \times v = \hat{\boldsymbol{\omega}} v$. With this notation, the dynamics of a rigid body controlled in velocity are

$$\dot{\boldsymbol{t}} = \mathbf{R}\,\boldsymbol{v}, \qquad \dot{\mathbf{R}} = \mathbf{R}\,\hat{\boldsymbol{\omega}}. \tag{1}$$

Let $\mathcal{Q}$ be the *configuration space* in which the robot moves. We assume that $\mathcal{Q}$ is a subgroup of $\mathrm{SE}(3)$ with Lie algebra q, such as $\mathrm{SE}(2)$ (planar motion), or $\mathbb{R}^3$ (pure translations).

Definition 1. A *vehicle kinematics* is a left-invariant dynamical system defined on a subgroup $\mathcal{Q} \leq \mathrm{SE}(3)$ such that $\dot{\boldsymbol{q}} = \boldsymbol{q}\mathbf{A}(\boldsymbol{u})$ where $\mathbf{A} : \mathcal{U} \to \mathrm{q}$ maps commands to instantaneous velocities.

The control results given in this paper (34) are limited to the holonomic case, where $\mathbf{A}$ spans the tangent space q.

2) Maps and environments: The definition of map depends on the sensors. We use simple sensors so the map will be simple as well. For range-finders, we need to describe the geometry of the obstacles. For the camera, following a Lambertian model, we need the reflectance of the obstacles.

Definition 2. A *map* is a tuple $\mathsf{m} = \langle \mathcal{O}, \mathcal{T}, \mathcal{F} \rangle$ where $\mathcal{O} \subset \mathbb{R}^3$ is a compact subset of the space representing obstacles, $\mathcal{T} : \partial\mathcal{O} \to \mathbb{R}$ is the reflectance of the obstacles, and $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}$ is a spatial field. We assume that $\partial\mathcal{O}$, $\mathcal{T}$, and $\mathcal{F}$ are all smooth. The set of all maps is indicated as Maps.

This definition can be extended in trivial ways. If there are multiple field samplers sampling different fields, then the codomain o $\mathcal{F}$ should be $\mathbb{R} \times \cdots \times \mathbb{R}$. For RGB cameras, the codomain of $\mathcal{T}$ should be $[0, 1]^3$ rather than $\mathbb{R}$. The definition can also be restricted to the planar case by letting $\mathbb{R}^2$ instead of $\mathbb{R}^3$ in the definition, and using the planar euclidean group $\mathrm{SE}(2)$ instead of $\mathrm{SE}(3)$ whenever it appears in the following discussion. A relatively strong assumption is that the map is static, but the methods developed here are robust to slight deviations from this assumptions.

We assume that the agent experiences the world in *episodes*. At the beginning of each episode, the agent wakes up in the same body but in a possibly different map. We call *environment* the mechanism that generates these maps.

Definition 3. The *environment* is a probability distribution on Maps that generates the map seen at each episode.

3) Relative Exteroceptive Robot Sensors: The sensors we consider are composed of a set of sensory elements (*sensels*) that are physically related to one another and belonging to the *sensel space* $\mathcal{S}$, which is assumed to be a differentiable manifold. Real robots have discrete sensors with a finite number of sensels, but at first we pretend that the sensel space $\mathcal{S}$ is continuous. At each time, the sensor returns the observations as a function from $\mathcal{S}$ to an output space, here assumed for simplicity to be $\mathbb{R}$. Independently of the sensor, this function is called "image", and the set of all functions from $\mathcal{S}$ to $\mathbb{R}$ is written as $\mathsf{Im}(\mathcal{S})$. We write the observations as $\boldsymbol{y} = \{y^s\}_{s \in \mathcal{V}}$, where s is the sensel position ranging over the *viewport* $\mathcal{V}$, which is a connected compact subset of the sensel space $\mathcal{S}$.

Example 4. For a central camera, $\mathcal{S}$ is the visual sphere $\mathbb{S}^2$; the viewport $\mathcal{V}$ is a rectanguloid carved into $\mathbb{S}^2$; $s \in \mathcal{S}$ is a pixel's direction, and y^s is the measured luminance.

A relative exteroceptive sensor is characterized by two properties: relativity and exteroceptivity. To formally state these properties, we need to define an *action* of the configuration space $\mathcal{Q}$ on the sensel space $\mathcal{S}$ and on the space of maps Maps.

Definition 5. There is an action of the configuration space $\mathcal{Q}$ defined on the sensel space $\mathcal{S}$: for every $\boldsymbol{q} \in \mathcal{Q}$ and $s \in \mathcal{S}$, we can define the element $\boldsymbol{q} \cdot s \in \mathcal{S}$, and $\boldsymbol{q}_1 \cdot (\boldsymbol{q}_2 \cdot s) = (\boldsymbol{q}_1 \boldsymbol{q}_2) \cdot s$.

Example 6. For a pan-tilt-roll camera, $\mathcal{S} = \mathbb{S}^2$, $\mathcal{Q} = \mathrm{SO}(3)$, and $\boldsymbol{q} \cdot s$ corresponds to applying the rotation $\boldsymbol{q}$ to $s \in \mathbb{S}^2$.

Likewise, using a construction typical of stochastic geometry [66, 67], we define an action of the group $\mathrm{SE}(3)$ on Maps. Given an element $g \in \mathrm{SE}(3)$, the map $g \cdot \mathsf{m}$ corresponds to rototranslating the world according to the motion g.

Definition 7. Given $g \in \mathrm{SE}(3)$ and $\mathsf{m} \in \mathsf{Maps}$, then the rototranslated map $\tilde{\mathsf{m}} = g \cdot \mathsf{m} = \langle \tilde{\mathcal{O}}, \tilde{\mathcal{T}}, \tilde{\mathcal{F}} \rangle$ is given by $\tilde{\mathcal{O}} = g \cdot \mathcal{O}$, $\tilde{\mathcal{T}}(\boldsymbol{p}) = \mathcal{T}(g \cdot \boldsymbol{p})$, $\tilde{\mathcal{F}}(\boldsymbol{p}) = \mathcal{F}(g \cdot \boldsymbol{p})$.

Definition 8. A *relative exteroceptive sensor* is a map $\psi : \mathsf{Maps} \times \mathcal{Q} \times \mathcal{S} \to \mathbb{R}$ with the properties:

a) Relativity: For all $\boldsymbol{q}, g \in \mathcal{Q}$:

$$\psi(\mathsf{m}, \boldsymbol{q}, s) = \psi(g \cdot \mathsf{m}, g^{-1}\boldsymbol{q}, s). \tag{2}$$

This describes 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.

b) Exteroceptivity: For all $\boldsymbol{q}, g \in \mathcal{Q}$:

$$\psi(\mathsf{m}, \boldsymbol{q}, s) = \psi(\mathsf{m}, \boldsymbol{q}g^{-1}, g \cdot s). \tag{3}$$

This describes the fact that the robot is "carrying" the sensor: ultimately the output at sensel s depends only on $\boldsymbol{q} \cdot s$, therefore it is invariant if we apply g to s and multiply $\boldsymbol{q}$ by g^{-1}.

4) Vehicles and robots: We first describe the "vehicle body" suspended in a vacuum, and then, to obtain a "vehicle", we pair a body with an environment.

Definition 9. A *vehicle body* B is a tuple $B = \langle \mathcal{U}, \mathcal{Q}, K, \langle \mathcal{S}, \mathcal{V}, \psi, \boldsymbol{r} \rangle \rangle$, where $\mathcal{U}$ is the command space, $K \in \mathcal{D}(\mathcal{Q}; \mathcal{U})$ is the vehicle kinematics (Definition 1), $\mathcal{S}$ is the sensel space, $\mathcal{V} \subset \mathcal{S}$ is the viewport, the function $\psi : \mathsf{Maps} \times \mathcal{Q} \times \mathcal{S} \to \mathcal{O}$ describes an exteroceptive sensor (Definition 8), and $\boldsymbol{r} \in \mathcal{Q}$ is the relative pose of the sensor with respect to the vehicle's reference frame.

Definition 10. A *vehicle* is a tuple $\langle B, e \rangle$ where B is a vehicle body (Definition 9) and e is an *environment* (Definition 3).

A vehicle is a system in $\mathcal{D}(\mathcal{Y}; \mathsf{Maps}; \mathcal{U})$, where the observations space is $\mathcal{Y} = \mathsf{Im}(\mathcal{V})$. The observations of a vehicles are constructed as follows. At the start of *episode* k, one samples a map $\mathsf{m}_k \in \mathsf{Maps}$ from the environment. We index time-dependent quantities with both the episode k and the time t. The pose $\boldsymbol{q}_{k,t}$ is the output of the kinematics K from $\boldsymbol{q}_{k,0} = \mathsf{Id}$. The observations $\boldsymbol{y}_{k,t}$ for episode k and time t are a function from $\mathcal{V}$ to $\mathbb{R}$ such that $y^s_{k,t} = \psi(\mathsf{m}_k, \boldsymbol{r}\boldsymbol{q}_{k,t}, s)$.

5) Idealized vehicles: At last, we can define exactly the class of systems that are used for this paper. The set $\mathsf{Vehicles} \subset \mathsf{Robots}$ of *idealized vehicles* are defined by choosing a subset of vehicles and a subset of environments.

Definition 11 (Idealized vehicles)**.** The set $\mathsf{Vehicles}$ is created by assembling every combination of vehicle body B and environment e such that:

- The vehicle body's kinematics has commands that can be interpreted as kinematic velocities: the command space $\mathcal{U}$ is $\mathbb{R}^{n_{\mathcal{U}}}$ and the map $\mathbf{A}$ in Definition 1 is linear.
- The sensors are chosen from the three "canonical sensors" classes $\mathrm{RF}(\mathcal{S})$, $\mathrm{VS}(\mathcal{S})$, $\mathrm{FS}(\mathcal{S})$, to be defined later in Section II-3.
- The environment e satisfies certain observability properties, defined later in Definition 31.

The set $\mathsf{Vehicles}$ is larger than the typical class of systems considered in identification/machine learning, but still small with respect to the set of all robots Robots. Enlarging the set to include Robots is the broad goal of our future work.

III. Dynamics of Canonical Exteroceptive Sensors

This section derives the models for three "canonical" robot sensors: field-samplers, range finders, and cameras (Table I). For each sensor, we define the observations y_t^s, where t is time and $s \in \mathcal{S}$ is the sensel, as a function of the map and the robot pose. Depending on the sensor, the interpretation of y_t^s changes from distance reading (range-finder), luminance (camera), intensity (field sampler). We then derive the *sensor dynamics,* which is an expression for $\dot{y}_t^s$ as a function of the sensor velocities. The models derived here are ideal, as they are noiseless, continuous-time and continuous-space. Noise and spatiotemporal discretization are modeled as nuisances which are applied to the ideal system (Section IX-4).

Table I
Dynamics of Canonical Exteroceptive Sensors

field sampler	$s \in \mathbb{R}^3$	$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vj}^s y_t^v)\omega_t^i + (\nabla_{vi}^s y_t^v) v_t^i$
camera	$s \in \mathbb{S}^2$	$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vj}^s y_t^v)\omega_t^i + \mu_t^s (\nabla_{vi}^s y_t^v) v_t^i$
range-finder	$s \in \mathbb{S}^2$	$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vj}^s y_t^v)\omega_t^i + (\nabla_{vi}^s \log y_t^v - s_i^*) v_t^i$

These equations are valid for sensels far from discontinuities (Definition 16). Note that the dynamics of the three sensors is formally the same for rotations (albeit the gradient operator is on $\mathbb{R}^3$ in the first case and on $\mathbb{S}^2$ in the others).

1) Field-samplers: A field-sampler, indicated by the icon on the side, is a sensor that samples the spatial field $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}$ defined as part of the map (Definition 2) and is an idealization of several sensors [68, 69].

Definition 12. An *ideal field-sampler* is a relative exteroceptive sensor (Definition 8) with sensel space $\mathcal{S} \subset \mathbb{R}^3$ defined by

$$\psi(\langle \mathcal{O}, \mathcal{T}, \mathcal{F} \rangle, \langle \boldsymbol{t}, \mathbf{R} \rangle, s) = \mathcal{F}(\boldsymbol{t} + \mathbf{R}\, s).$$

The set of all field-samplers is $\mathrm{FS}(\mathcal{S}) \subset \mathcal{D}(\mathsf{Im}(\mathcal{S}); \mathsf{Maps}; \mathsf{se}(3))$.

The dynamics of a field sampler are particularly simple and can be written compactly using a tensor notation. The derivative $\dot{\boldsymbol{y}}$ depends on the spatial gradient. Given a function $\boldsymbol{y} \in \mathsf{Im}(\mathcal{S})$, the symbol $\nabla \boldsymbol{y}$ represents is the spatial gradient with respect to s. The gradient is a $(0, \dim(\mathcal{S}))$ tensor at each point of $\mathcal{S}$. The gradient being a linear operator, it can be written as a $(1,2)$ tensor field ∇_{vi}^s, so that $\nabla_{vi}^s y^v$ is the i-th component of the gradient at point s. This notation is valid both in the Euclidean case ($s \in \mathbb{R}^n$) and in the manifold case (e.g., $s \in \mathbb{S}^2$).

Proposition 13. *The dynamics of a field-sampler are bilinear in $\boldsymbol{y}$ and the sensor velocities $\boldsymbol{v}, \boldsymbol{\omega}$:*

$$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vj}^s y_t^v)\omega_t^i + (\nabla_{vi}^s y_t^v) v_t^i. \tag{4}$$

Proof: Fix a sensel $s \in \mathbb{R}^3$. Let $\boldsymbol{z} = \boldsymbol{t} + \mathbf{R}\, s$ so that $y(s) = \mathcal{F}(\boldsymbol{z})$. The values $\boldsymbol{z}$, $\boldsymbol{t}$, and $\mathbf{R}$ depend on time but we omit the time subscript for clarity. We can compute the derivative from $\dot{y}^s = \nabla_{vi}^s \mathcal{F}^v \dot{z}^i$ if we have an expression for $\dot{\boldsymbol{z}}^i$ and $\nabla_{vi}^s \mathcal{F}^v$. As for $\dot{\boldsymbol{z}}^i$, from (1) it follows that $\dot{\boldsymbol{z}}^i = R_k^i (v^k + \hat{\omega}_j^k s^j) = R_k^i (v^k + \hat{s}_j{}^k \omega^j)$. As for $\nabla_{vi}^s \mathcal{F}^v$, from $\mathcal{F}(\boldsymbol{z}) = \boldsymbol{y}(R_j^{*i}(z^j - t^j))$, where R_j^* is the transpose of the rotation matrix $\mathbf{R}$, it follows that $\nabla_{qi}^z \mathcal{F}^q = \nabla_{qi}^z y (R_z^{*i}(q^j - t^j)) = \nabla_{si}^z y^s \; \nabla_{qi}^s \; R_i^{*j}(q^j - t^j) = \nabla_{sj}^z y^s \; R_i^{*j}$. Putting all together, we obtain $\dot{y}^s = \nabla_{qi}^s \mathcal{F}^q \dot{z}^i = \nabla_{sb}^z \boldsymbol{y}^s \; R_i^{*j} \; R_k^i (v_j^b + \hat{s}^j \omega_j^b)$. Substituting $R_i^{*b} R_k^i = \mathrm{Id}_k^b$ and rearranging gives (4). ∎

2) Range finders: The readings of an ideal range finder measure the distance from a reference point (in $\mathbb{R}^3$) to the closest obstacle in a certain direction (in $\mathbb{S}^2$). (The yellow icon we use is a tribute to the Sick range-finder.) First, we define the "ray tracing" function, which depends on the obstacles $\mathcal{O} \subset \mathbb{R}^3$.

Definition 14 (Ray tracing function)**.** The *ray tracing function*

$$\sigma_{\mathcal{O}} : (\mathbb{R}^3 \setminus \mathcal{O}) \times \mathbb{S}^2 \to \mathbb{R}^+ \cup \infty$$

is the minimum value of α such that the ray of length α shot in direction s from point $\boldsymbol{p}$ hits the obstacle $\mathcal{O}$. Let $I_{\mathcal{O}} = \{\alpha \mid \boldsymbol{p} + \alpha s \in \mathcal{O}\} \subset \mathbb{R}$ be the values of α for which the ray hits $\mathcal{O}$. Then the ray tracing function can be written as

$$\sigma_{\mathcal{O}}(\boldsymbol{p}, s) = \begin{cases} \min\, I_{\mathcal{O}}(\boldsymbol{p}, s) & I_{\mathcal{O}}(\boldsymbol{p}, s) \neq \emptyset, \\ \infty & I_{\mathcal{O}}(\boldsymbol{p}, s) = \emptyset. \end{cases}$$

The following is the formal definition of a range finder as a relative sensor in the form required by Definition 8.

Definition 15. An *ideal range finder* is a relative exteroceptive sensor with sensel space $\mathcal{S} \subset \mathbb{S}^2$ defined by

$$\psi(\langle \mathcal{O}, \mathcal{T}, \mathcal{F} \rangle, \langle \boldsymbol{t}, \mathbf{R} \rangle, s) = \sigma_{\mathcal{O}}(\boldsymbol{t}, \mathbf{R}s).$$

The set of all range finders is $\mathrm{RF}(\mathcal{S}) \subset \mathcal{D}(\mathsf{Im}(\mathcal{S}); \mathsf{Maps}; \mathsf{se}(3))$.

The dynamics of a range finder are differentiable only where the ray tracing function itself is differentiable.

Definition 16 (Discontinuities)**.** The subset $\mathcal{S}_{\boldsymbol{p}}^{\mathrm{dif}}$ of $\mathcal{S}$ describes the points for which $\sigma_{\mathcal{O}}(\boldsymbol{p}, \cdot)$ is differentiable:

$$\mathcal{S}_{\boldsymbol{p}}^{\mathrm{dif}} = \{s \in \mathcal{S} \mid \sigma_{\mathcal{O}}(\boldsymbol{p}, \cdot) \text{ is differentiable at } s\}.$$

If the obstacles $\mathcal{O}$ are smooth, $\mathcal{S}_{\boldsymbol{p}} \setminus \mathcal{S}_{\boldsymbol{p}}^{\mathrm{dif}}$ are the occlusions.

Proposition 17. *The dynamics of an ideal range finder are defined for $\{s \mid \mathbf{R}s \in \mathcal{S}_{\boldsymbol{t}}^{dif}\}$ and given by*

$$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vi}^s y_t^v)\omega_t^i + (\nabla_{vi}^s \log y_t^v - s_i^*) v_t^i. \tag{5}$$

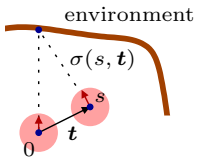
The "$-s_i^*$" term in (5) means that if the velocity $\boldsymbol{v}$ is in the direction on s, then the range decreases (the remaining nonlinear term $\nabla_i \log \sigma^s$ is less intuitive).

Proof: (This proof is due to Shuo Han) The dynamics for rotation (first term in (5)) is the same as the field-sampler; hence we are only concerned in proving the result for translation assuming zero rotation. Write $\sigma = \sigma(s, \boldsymbol{t})$ as a function of the direction s and the robot position $\boldsymbol{t} \in \mathbb{R}^3$. Then we have to prove that

$$\frac{\partial}{\partial \boldsymbol{t}} \sigma(s, \boldsymbol{t}) = \nabla \log \sigma(s, 0) - s^*. \tag{6}$$

Without loss of generality, we can assume we are computing the derivative at $\boldsymbol{t} = 0$. In a neighborhood of 0, it holds that

$$\|\boldsymbol{t} + \sigma(s)s\| = \sigma \left(\frac{\boldsymbol{t} + \sigma(s)s}{\|\boldsymbol{t} + \sigma(s)s\|}, 0 \right), \tag{7}$$



as can be seen by geometric inspection. The proof is based on the implicit function theorem applied to the relation (7). Define the function $n(v) : \mathbb{R}^3 \to \mathbb{R}^3$ as the vector v normalized by

its module: $n(v) \triangleq v/\|v\|$. Then (7) can be rewritten as the implicit function

$$F(\sigma, s, \boldsymbol{t}) = \|\boldsymbol{t} + \sigma(s)s\| - \sigma(n(\boldsymbol{t} + \sigma(s)s), 0) = 0.$$

The derivative $\partial\sigma(s,\boldsymbol{t})/\partial\boldsymbol{t}$ can be computed using the implicit function theorem applied to F, which states that $\partial\sigma/\partial\boldsymbol{t} = (\partial F/\partial\sigma)^{-1}\partial F/\partial\boldsymbol{t}$. Define $P : \mathbb{S}^2 \to \mathbb{R}^{3\times3}$ as the projector $P_v = I - vv^*$. The derivative of $n(v)$ is then $\partial n(v)/\partial v = P_v/\|v\|$.

We use the shortcut $x = \boldsymbol{t} + \sigma(s)s$, and $\sigma_0(s) = \sigma(s, 0)$.

$$\frac{\partial F}{\partial \boldsymbol{t}} = \frac{x^*}{\|x\|} - \nabla_u \sigma_0(u) P_u|_{u=n(x)} \frac{P_{n(x)}}{\|x\|}.$$

For $\partial F/\partial\sigma$ we simply obtain $\frac{\partial F}{\partial\sigma} = \frac{\partial F}{\partial p}s$. For $\frac{\partial\sigma}{\partial\boldsymbol{t}}$ we obtain:

$$\frac{\partial\sigma}{\partial\boldsymbol{t}} = -\frac{\partial F/\partial\boldsymbol{t}}{\partial F/\partial\sigma} = -\frac{x^* - \nabla_u\sigma_0(u)P_u|_{u=n(x)}P_{n(x)}}{x^*s - \nabla_u\sigma_0(u)P_u|_{u=n(x)}P_{n(x)}s}.$$

This expression is valid in a neighborhood of $\boldsymbol{t} = 0$. As $\boldsymbol{t} \to 0$, $x \to \sigma(s)s$, $\|x\| \to \sigma(s)$, and $n(x) \to s$. Substituting all of these, we obtain

$$\frac{\partial\sigma}{\partial\boldsymbol{t}} \to -\frac{\sigma(s)s^* - \nabla_s\sigma_0(s)P_s^2}{\sigma(s)s^*s - \nabla_s\sigma_0(s)P_s^2 s}.$$

Using the fact that $P(s)s = 0$ and $\nabla_s\sigma_0(s)P(s) = \nabla_s\sigma_0(s)$ (the gradient is tangent to s), we simplify it to

$$\frac{\partial\sigma}{\partial\boldsymbol{t}} = -\frac{\sigma(s)s^* - \nabla_s\sigma_0(s)}{\sigma(s)} = \frac{\nabla_s\sigma_0(s)}{\sigma(s)} - s^*,$$

from which (6) is easily obtained. ∎

3) Vision sensors: Vision sensors configurations can be quite exotic [70, 71]. We consider a *central* camera with one focus point, so that the sensel space is the visual sphere $\mathbb{S}^2$ (in general, the sensel space should be $\mathbb{R}^3 \times \mathbb{S}^2$). We also ignore all complications in the physics of light scattering. The following definition is applicable only if the surfaces have a Lambertian light model, far from occlusions, and considering a static world. See Soatto [72] for a formal definition that takes into account occlusions and shading. The output of an ideal vision sensor depends on the texture $\mathcal{T} : \partial\mathcal{O} \to \mathbb{R}$ defined on the border of the obstacles $\partial\mathcal{O}$. Given the ray tracing function $\sigma_\mathcal{O}$, the point sampled in direction s from point $\boldsymbol{p}$ is $\boldsymbol{p} + \sigma_\mathcal{O}(\boldsymbol{p}, s)\, s$ and its luminance is $\mathcal{T}(\boldsymbol{p} + \sigma_\mathcal{O}(\boldsymbol{p}, s)\, s)$.

Definition 18. An *ideal vision sensor* is a relative exteroceptive sensor with sensel space $\mathcal{S} \subset \mathbb{S}^2$ defined by

$$\psi(\langle\mathcal{O}, \mathcal{T}, \mathcal{F}\rangle, \langle\boldsymbol{t}, \mathbf{R}\rangle, s) = \mathcal{T}(\boldsymbol{t} + \sigma_\mathcal{O}(\boldsymbol{t}, \mathbf{R}s)\mathbf{R}s).$$

The set of all vision sensors is $\mathsf{VS}(\mathcal{S}) \subset \mathcal{D}(\mathsf{Im}(\mathcal{S}); \mathsf{Maps}; \mathsf{se}(3))$.

The dynamics depends on the *inverse* of the distance to the obstacles, called *nearness* and indicated by $\mu = 1/\sigma$.

Proposition 19. *The dynamics of an ideal vision sensor are defined for* $\{s \mid \mathbf{R}s \in \mathcal{S}_{\boldsymbol{t}}^{dif}\}$ *and given by*

$$\dot{y}_t^s = (\hat{s}_i^j \nabla_{vi}^s y_t^v)\omega_t^i + (\mu_t^s \nabla_{vi}^s y^v)\, v_t^i. \tag{8}$$

Remark 20. (8) is the dynamics of the luminance signal seen by a *fixed* pixel $s \in \mathbb{S}^2$. It is *not* the dynamics of the position of a point feature that moves in the visual field, which is commonly used in other contexts, such as structure from motion.[1]

[1]Using our notation, those dynamics are $\dot{s} = s \times \boldsymbol{\omega} + \mu_t^s\,(1 - ss^*)\,\boldsymbol{v}$.

IV. Bilinear Dynamics Sensors

This section introduces the class of Bilinear Dynamics Sensors (BDS), which can be used as an instantaneous approximation of the dynamics the canonical sensors. The following sections will then describe learning and control algorithms.

1) Why using bilinear dynamics: One reason to use bilinear dynamics as a generic model is that the dynamics of the canonical sensors is approximately bilinear (Table I); indeed, the field-sampler dynamics is exactly bilinear. Ignoring those results, we can also justify the bilinear guess from first principles. Suppose we we want to model the observations dynamics $\dot{\boldsymbol{y}} = g(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{x})$, where g is, in general, nonlinear and dependent on an inaccessible hidden state $\boldsymbol{x}$. If we ignore this dependence, we are left with 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. A second-order linearization of g leads to the expression

$$\dot{\boldsymbol{y}} = \boldsymbol{a} + A\boldsymbol{y} + B\boldsymbol{u} + C(\boldsymbol{y}, \boldsymbol{y}) + D(\boldsymbol{y}, \boldsymbol{u}) + E(\boldsymbol{u}, \boldsymbol{u}). \tag{9}$$

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 $n_{\boldsymbol{y}}$ and $n_{\boldsymbol{u}}$, then C, D, E have dimensions, respectively, $n_{\boldsymbol{y}} \times n_{\boldsymbol{y}} \times n_{\boldsymbol{y}}$, $n_{\boldsymbol{y}} \times n_{\boldsymbol{y}} \times n_{\boldsymbol{u}}$, and $n_{\boldsymbol{y}} \times n_{\boldsymbol{u}} \times n_{\boldsymbol{u}}$. We can ignore some terms in (9) by using some semantic assumptions regarding the specific context. If $\boldsymbol{u}$ 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 $C(\boldsymbol{y}, \boldsymbol{y})$. 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. In conclusion, our *ansatz* to capture the dynamics of a generic robot sensor is $\dot{\boldsymbol{y}} = B\boldsymbol{u} + D(\boldsymbol{y}, \boldsymbol{u})$.

2) Discrete and continuous BDS *models:* We describe two types of bilinear dynamics. The BDS class (Definition 21) assumes that the observations are a vector of a real numbers, while the CBDS class (Definition 22) assumes that the observations are a function on a manifold.

Definition 21. A system is a *bilinear dynamics sensor* (BDS), if $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$, $\boldsymbol{u} \in \mathbb{R}^{n_{\boldsymbol{u}}}$ and there exist a $(1, 2)$ tensor $\mathbf{M}$ and a $(1, 1)$ tensor $\mathbf{N}$ such that

$$\dot{y}_t^s = \textstyle\sum_{i=1}^{n_u} \left(\sum_{v=1}^{n_y} \mathsf{M}_{vi}^s y_t^v + \mathsf{N}_i^s\right) u_t^i. \tag{10}$$

$\mathsf{BDS}(n; k) \subset \mathcal{D}(\mathbb{R}^n; \mathbb{R}^k)$ is the set of all such systems.

We will be using the Einstein conventions for tensor calculus, in which repeated up and down indices are summed over, so that the explicit "$\sum$" symbol can be omitted.

In the continuous-space case, the formalization is entirely similar—the only change is that the index s, rather than an integer, is a point in a manifold $\mathcal{S}$.

Definition 22. A system is a *Continuous-space BDS system* (CBDS) if $\boldsymbol{y}_t \in \mathsf{Im}(\mathcal{S})$, $\boldsymbol{u} \in \mathbb{R}^{n_{\boldsymbol{u}}}$, and there exists a $(1, 2)$ tensor *field* $\mathbf{M}$ and a $(1, 1)$ tensor *field* $\mathbf{N}$ such that

$$\dot{y}_t^s = \textstyle\sum_{i=1}^{n_u} \left(\int_{v\in\mathcal{S}} \mathsf{M}_{vi}^s y_t^v \,\mathrm{d}\mathcal{S} + \mathsf{N}_i^s\right) u_t^i. \tag{11}$$

$\mathsf{CBDS}(\mathcal{S}; k) \subset \mathcal{D}(\mathsf{Im}(\mathcal{S}); \mathbb{R}^k)$ is the set of all such systems.

Remark 23. The literature includes similar models, but not quite equivalent, for the finite-dimensional case. Elliot [73] calls *bilinear control system* a generalization of (10) which includes also a linear term $A\boldsymbol{y}$, and a particular case of (10) where $\mathsf{N}_i^s = 0$ is called *symmetric bilinear control system.*

3) Spatial sampling preserves bilinearity : The BDS model is defined for a discrete set of sensels, while the models for the canonical sensors were defined for continuous time and space. A natural question is whether the properties of the system (bilinearity) are preserved for discretely sampled systems, both in time and space. As for space discretization, is we assume that the continuous function y_t^s, $s \in \mathcal{S}$, is sampled at a dense enough set of points to allow precise reconstruction, then the sampled system has bilinear dynamics. If the sampling is uniform, sampling needs to be higher than at the Nyquist frequency. However, this is not a necessary condition; see Margolis [74, Chapter 3] for an elementary exposition on nonuniform sampling.

Proposition 24. *Suppose that $\boldsymbol{y}_t \in \mathsf{Im}(\mathcal{S})$ has bilinear dynamics described by a system in $\mathrm{CBDS}(\mathcal{S}; n_{\boldsymbol{u}})$. Let $\{s^1, \ldots, s^{n_{\boldsymbol{y}}}\} \in \mathcal{S}$ be a sufficient sampling sequence and $\tilde{\boldsymbol{y}}_t = \{y_t^{s^1}, \ldots, y_t^{s^{n_{\boldsymbol{y}}}}\} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ be the sampled signal. Then the dynamics of $\tilde{\boldsymbol{y}}_t$ is described by a system in $\mathrm{BDS}(n_{\boldsymbol{y}}; n_{\boldsymbol{u}})$.*

Proof: (Sketch) Because the sampling is sufficient, it is possible to reconstruct all values $\{y_t^s\}_{s\in\mathcal{S}}$ as a linear *combination of values of $\tilde{\boldsymbol{y}}_t$: there exist kernels $\varphi_i(s)$ such that $y_t^s = \sum_i \varphi_i(s)\tilde{y}_t^i$. From this the results follows easily.* ■

4) Temporal sampling preserves bilinearity to first order: Suppose the dynamics of $\boldsymbol{y}$ is bilinear: $\dot{\boldsymbol{y}} = \mathbf{M}\boldsymbol{u}\boldsymbol{y}$ (for simplicity, we ignore the affine part of the dynamics), and that the dynamics is discretized at intervals of length T. The value of $\boldsymbol{y}$ at the $(k+1)$-th instant is given by $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \int_{kT}^{kT+T} \mathbf{M}\boldsymbol{u}_t\boldsymbol{y}_t \,\mathrm{d}t$. Assuming that the commands take the constant value $\boldsymbol{u}_k$ in the interval $[kT, kT+T]$, the solution can be written using the matrix exponential as $\boldsymbol{y}_{k+1} = \mathsf{mexp}(T(\mathbf{M}\boldsymbol{u}_k))\boldsymbol{y}_k$. Using the series expansion of the matrix exponential, the first few terms are $\mathsf{mexp}(T(\mathbf{M}\boldsymbol{u}_k) = \mathbf{I} + T\mathbf{M}\boldsymbol{u}_k + o(T^2)$. The difference $\Delta\boldsymbol{y}_k = \frac{1}{T}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k)$ is then linear with respect to $\boldsymbol{u}$ only at the first order:

$$\Delta\boldsymbol{y}_k = \mathbf{M}\boldsymbol{u}_k\boldsymbol{y}_k + o(T^2).$$

V. Learning bilinear dynamics

This section presents a learning/identification algorithm for the class of BDS/CBDS tailored for bootstrapping: it can be written as a streaming algorithm and implemented using simple parallel operations (Algorithm 1).

1) A streaming algorithm for learning BDS*:* The agent is summarized as Algorithm 1. During the exploration phase, the commands $\boldsymbol{u}_t$ are sampled from any zero-mean distribution with a positive definite covariance. (Alternatively, the agent observes a stream of commands chosen by another entity.) As the agent observes the stream of data $\langle \boldsymbol{y}_t, \dot{\boldsymbol{y}}_t, \boldsymbol{u}_t \rangle$, it estimates the following statistics, described by (12)–(16): the mean of the observations and the $(2,0)$ covariance tensor $\mathbf{P}$; the $(2,0)$ covariance tensor $\mathbf{Q}$ for the commands; a $(3,0)$ tensor $\mathbf{T}$, and a $(2,0)$ tensor $\mathbf{U}$. For simplicity, the analysis is done in the continuous-time case (the agent observes $\dot{\boldsymbol{y}}$), and in the asymptotic regime, so that we assume that all sample averages converge to the expectations and the operators $\mathbb{E}$, cov are used for either. We use the following relation between the estimated $\mathbf{P}, \mathbf{Q}, \mathbf{T}, \mathbf{U}$ and the unknown model parameters $\mathbf{M}$, $\mathbf{N}$.

Algorithm 1 A streaming algorithm for learning BDS

- The agent generates commands $\boldsymbol{u}_t$ using a random sequence with positive-definite covariance matrix. (Alternatively, it observes commands chosen by another entity.)
- The agent observes the stream of data $\langle \boldsymbol{y}_t, \dot{\boldsymbol{y}}_t, \boldsymbol{u}_t \rangle$, and from these data it computes the following statistics:

$$\overline{y}^s = \mathbb{E}\{y_t^s\}, \tag{12}$$
$$\mathsf{P}^{sv} = \mathrm{cov}(y_t^s, y_t^v), \tag{13}$$
$$\mathsf{Q}^{ij} = \mathrm{cov}(u_t^i, u_t^j), \tag{14}$$
$$\mathsf{T}^{svi} = \mathbb{E}\{(y_t^s - \overline{y}^s)\,\dot{y}_t^v u_t^i\}, \tag{15}$$
$$\mathsf{U}^{vi} = \mathbb{E}\{\dot{y}_t^v u_t^i\}. \tag{16}$$

- The parameters for a BDS model are recovered using

$$\mathsf{M}_{xj}^v = \mathsf{T}^{svi}\mathsf{P}_{sx}^{-1}\mathsf{Q}_{ij}^{-1}, \tag{17}$$
$$\mathsf{N}_i^s = \mathsf{U}^{sj}\mathsf{Q}_{ij}^{-1} - \mathsf{M}_{vi}^s\overline{y}^v. \tag{18}$$

Lemma 25. *Let $\mathbf{P}$, $\mathbf{Q}$ be the covariance of $\boldsymbol{y}$ and $\boldsymbol{u}$. Then the tensors $\mathbf{T}$ and $\mathbf{U}$ tend asymptotically to*

$$\mathsf{T}^{svi} = \mathsf{M}_{qj}^s\mathsf{P}^{qv}\mathsf{Q}^{ij}, \tag{19}$$
$$\mathsf{U}^{sj} = (\mathsf{M}_{vi}^s\overline{y}^v + \mathsf{N}_i^s)\mathsf{Q}^{ij}. \tag{20}$$

Proof: The computation for $\mathbf{T}$ is as follows:

$$\begin{aligned}\mathsf{T}^{svi} &= \mathbb{E}\{\dot{y}_t^s\,(y_t^v - \overline{y}_t^v)\,u_t^i\}\\ &= \mathbb{E}\{((\mathsf{M}_{qj}^s y_t^q + \mathsf{N}_j^s)u_t^j)(y_t^v - \overline{y}^v)u_t^i\} \qquad (21)\\ &\qquad \text{(Independence of } \boldsymbol{u}, \boldsymbol{y}.)\\ &= \mathsf{M}_{qj}^s\mathbb{E}\left\{y_t^q\,(y_t^v - \overline{y}^v)\right\}\mathbb{E}\{u_t^i u_t^j\} = \mathsf{M}_{qj}^s\mathsf{P}^{qv}\mathsf{Q}^{ij} \qquad (22)\end{aligned}$$

The analogous computation for $\mathbf{U}$ is $\mathsf{U}^{sj} = \mathbb{E}\{\dot{y}_t^v\check{u}_t^j\} = \mathbb{E}\{(\mathsf{M}_{vi}^s y_t^v + \mathsf{N}_i^s)u_l^i\check{u}_t{}^j\} = (\mathsf{M}_{vi}^s\overline{y}^v + \mathsf{N}_i^s)\mathsf{Q}^{ij}$. ■

Following this result, $\mathbf{M}$ and $\mathbf{N}$ are obtained using (17)–(18).

A general pattern can be observed in (19). Every quantity that the agent learns ultimately depends on three factors:

1) *The agent's body dynamics.* In this case, the tensor $\mathbf{M}$.
2) *The environment statistics*. In this case, the covariance $\mathbf{P}$, which depends on the statistics of the environment (for a camera, the environment texture $\mathcal{T}$).
3) *The experience the agent had in such environment*. In this case, the tensor $\mathbf{Q}$ captures the kind of "training" the agent had in the environment.

If the agent wants to learn its body dynamics, the other two factors are nuisances to be removed (here, using (17)–(18)).

2) Singular case: We have assumed in (17)–(18) that the covariance is full rank. If it is not (e.g., a dead sensel, or two sensels giving the same), then the model is unobservable in a certain subspace corresponding to the kernel of $\mathbf{P}$ and those operations are to be projected in on the complement of the kernel.

3) Effect of additive noise: Suppose that instead of $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$ we observe $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t + \epsilon_t$ and $\tilde{\dot{\boldsymbol{y}}} = \dot{\boldsymbol{y}} + \nu_t$; and the commands seen by the robot are $\tilde{\boldsymbol{u}}_t = \boldsymbol{u}_t + \xi_t$, where ϵ, ξ, ν are zero-mean white *independent* stochastic processes. Then running Algorithm 1 with the perturbed values would give the same tensors $\mathbf{M}$, $\mathbf{N}$, because all noise terms cancel when computing expected values of mixed products, such as in $\mathbb{E}\{\dot{y}^s\,(y^v - \overline{y}^v)\,u^i\}$.

One particular case in which the noise is *not* independent depends on how the derivative $\dot{\boldsymbol{y}}$ is approximated. Suppose that $\{t_k\}_{k\in\mathbb{Z}}$ is the sequence of timestamps for the data available to the agent. If one uses a two-tap approximation to compute the derivative ($\dot{\boldsymbol{y}}_{t_k} \simeq (\boldsymbol{y}_{t_{k+1}} - \boldsymbol{y}_{t_{k-1}})/(t_{k+1} - t_{k-1})$) then the noise on $\dot{\boldsymbol{y}}_{t_k}$ is independent of the noise on $\boldsymbol{y}_{t_k}$. If, however, a simple Euler approximation is used, by computing $\dot{\boldsymbol{y}}_{t_k} \simeq (\boldsymbol{y}_{t_k} - \boldsymbol{y}_{t_{k-1}})/(t_k - t_{k-1})$, then the noise is dependent and the formulas given are not valid anymore.

4) Streaming implementation: These expectations can be computed recursively; for example, $\overline{y}^s_{k+1} = \frac{k}{k+1}\overline{y}^s_k + \frac{1}{k+1}y^s_k$.

The computation (15) is similar to three-way Hebbian learning between $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, and $\boldsymbol{u}$; the expectation of the product can be thought as an approximation of the frequency that the three signals are active together. There is also a computational similarity with deep learning methods using three-way interactions [23–26], as well as subspace approaches in the system identification literature for learning bilinear systems [75–77].

5) Being careful with indices: When implementing these operations, it is easy to get confused with the order of the indices.[2] Numerical packages usually offer some shortcuts to define complex operations on sets of tensors. For example, in Python's library *Numpy*, the operation (17) can be implemented using the user-friendly *einsum* function as follows:

```
M = einsum("svi,sx,ij->vxj",T,P_inv,Q_inv)
```

VI. BDS approximations of canonical sensors

This section describes what is the learned BDS approximation of the canonical robotic sensors, assuming certain constraints on the environment and the exploration strategy. The results are summarized in Table II.

Table II
BDS approximations of canonical sensors

field sampler	$s \in \mathbb{R}^3$	$\dot{y}^s_t = (\hat{s}^j_i \nabla^s_{vi} y^v_t)\omega^i_t + (\nabla^s_{vi} y^v_t) v^i_t$
camera	$s \in \mathbb{S}^2$	$\dot{y}^s_t = (\hat{s}^j_i \nabla^s_{jv} y^v_t)\omega^i_t + \overline{\mu}(\nabla^s_{vi} y^v_t) v^i_t$
range-finder	$s \in \mathbb{S}^2$	$\dot{y}^s_t = (\hat{s}^j_i \nabla^s_{jv} y^v_t)\omega^i_t + \alpha\overline{\mu}(\nabla^s_{vi} y^v_t - s^*_i) v^i$

The symbol $\overline{\mu}$ denotes the average nearness (inverse of distance); α is a positive scalar.

1) Sufficient exploration: In identification it is often necessary to impose certain observability conditions on the data that make the system identifiable. The analogous concept in this context are the symmetries of the distribution over maps seen by the agent. Let the *training distribution* $p_{\mathrm{T}} \in \mathsf{Measures}(\mathsf{Maps} \times \mathcal{Q})$ be the distribution over maps/poses experienced by the agent during learning. This distribution depends on the environment e and the exploration strategy. If m_k is the map at episode k and $\boldsymbol{q}_{k,t}$ is the pose, the *subjective map,* which is the map from the point of view of the robot, is given by

$$\mathsf{m}'_{k,t} \triangleq \boldsymbol{q}^{-1}_{k,t} \cdot \mathsf{m}_k \in \mathsf{Maps}. \tag{23}$$

Its distribution $p_{\mathsf{m}'} \in \mathsf{Measures}(\mathsf{Maps})$ is found from p_{T} by marginalization. The *symmetries* of such distribution describe whether the agent has explored "enough".

Definition 26. $\mathrm{Sym}(p_{\mathsf{m}'})$ is the subgroup of $\mathcal{Q}$ to which the distribution $p_{\mathsf{m}'}$ is invariant: $g \in \mathrm{Sym}(p_{\mathsf{m}'}) \Leftrightarrow p(g\cdot\mathsf{m}) = p_{\mathsf{m}'}(\mathsf{m})$.

Example 27. A planar robot ($\mathcal{Q} = \mathrm{SE}(2)$) lives in a basement lab that is always kept extremely tidy: the map $\mathsf{m}_0 \in \mathsf{Maps}$ never changes across episodes. At the beginning of each episode the robot is turned on at a random pose, not necessarily uniformly distributed. The robot spins in place in a random direction until the batteries die. In this case, the group $\mathrm{Sym}(p_{m'})$ is the group of planar rotations $\mathsf{SO}(2)$, because of the motion.

Whether the symmetry is "symmetric enough" depends on the sensor. For example, $\mathrm{Sym}(p_{m'}) = \mathsf{SO}(2)$ is enough for making the statistics of a 2D range-finder uniform across the viewport $\mathcal{V} \subset \mathbb{S}^1$, but it would not be sufficient if the sensor was a camera for which $\mathcal{S} = \mathbb{S}^2$. What we need is a way to measure the viewport $\mathcal{V}$ using group actions.

Definition 28. Let $\mathrm{Sym}(\mathcal{V})$ be the minimal subgroup of $\mathcal{Q}$ such that there exists a $s_0 \in \mathcal{V}$ for which any $s \in \mathcal{V}$ can be written as $g_s \cdot s_0$ for some $g_s \in \mathrm{Sym}(\mathcal{V})$.

Example 29. For a planar range-finder with $\mathcal{V} \subset \mathbb{S}^1$, $\mathrm{Sym}(\mathcal{V}) = \mathrm{SO}(2)$. For a camera with $\mathcal{V} \subset \mathbb{S}^2$, $\mathrm{Sym}(\mathcal{V}) = \mathrm{SO}(3)$. For a field-sampler with $\mathcal{V} \subset \mathbb{R}^2$, $\mathrm{Sym}(\mathcal{V}) = \mathrm{SE}(2)$.

After these technical preliminaries, we can state elegantly a uniformity result.

Proposition 30. *If $\mathrm{Sym}(p_{m'}) \leq \mathrm{Sym}(\mathcal{V})$ the expected value of the observations of any relative exteroceptive sensor ψ is the same for all sensels s: $\mathbb{E}_{p_{\mathrm{T}}}\{\psi(\mathsf{m}, \boldsymbol{q}, s)\} = c$.*

Proof: The result depends on algebraic manipulation based on (2), (3), and the definition (23):

$$\begin{aligned}
& \mathbb{E}_{p_{\mathrm{T}}}\{\psi(\mathsf{m}, \boldsymbol{q}, s)\} \overset{(2)}{=} \mathbb{E}_{p_{\mathrm{T}}}\{\psi(\boldsymbol{q}^{-1} \cdot \mathsf{m}, \mathrm{Id}, s)\} \\
\overset{(23)}{=}\ & \mathbb{E}_{p(\mathsf{m}')}\{\psi(\mathsf{m}', \mathrm{Id}, s)\} \\
=\ & \quad (\exists g_s \in \mathrm{Sym}(\mathcal{V}) \text{ such that } s = g_s \cdot s_0) \\
& \mathbb{E}_{p(\mathsf{m}')}\{\psi(\mathsf{m}', \mathrm{Id}, g_s \cdot s_0)\} \\
\overset{(3)}{=}\ & \mathbb{E}_{p(\mathsf{m}')}\{\psi(\mathsf{m}', g_s^{-1}, s_0)\} \overset{(2)}{=} \mathbb{E}_{p(\mathsf{m}')}\{\psi(g_s^{-1} \cdot \mathsf{m}', \mathrm{Id}, s_0)\} \\
=\ & \quad (g_s^{-1} \in \mathrm{Sym}(\mathcal{V}) \subset \mathrm{Sym}(p_{m'})) \\
& \mathbb{E}_{p(\mathsf{m}')}\{\psi(\mathsf{m}', \mathrm{Id}, s_0)\},
\end{aligned}$$

which is independent of the sensel s. ∎

To make the following development simple, we group together all assumptions on the environment and training data.

Definition 31 (Common assumptions on environment and exploration)**.** We make the following assumptions on the environment and exploration:

1) $\mathsf{Q}^{ij} = \mathrm{cov}(u^i, u^j)$ is positive definite.
2) $\mathsf{P}^{sv} = \mathrm{cov}(y^s, y^v)$ is positive definite.
3) $\mathrm{Sym}(p_{m'}) \leq \mathrm{Sym}(\mathcal{V})$.
4) $\boldsymbol{u}_t$ was chosen independently of $\boldsymbol{y}_t$.

[2] The tensors $\mathbf{T}$ for the Vehicles are antisymmetric: if two indices are swapped, the robot moves in the opposite direction as expected…

5) The distance to the obstacles in direction s and the reflectance of the obstacle are independent.

The first three are essentially *excitability* conditions: for example, the second assures that a robot with a camera is not kept in the dark. Assumption 4 is very convenient in the derivations (one correlation less to take into account); it holds if the exploration uses random babbling, but not if the robot is guided along trajectories based on the observations. The last assumption is very mild; it is satisfied for example if the robot's exploration covers the whole environment, as eventually it will see any texture from every distance.

Field-samplers: The dynamics is exactly bilinear ($\mathrm{FS}(\mathcal{S}) \subset \mathrm{CBDS}(\mathcal{S}; \mathbb{R}^{n_{\boldsymbol{u}}})$) so there is no approximation.

Vision sensors: Their dynamics contains a hidden state, the nearness μ^s_t. The best approximant in $\mathrm{CBDS}(\mathcal{S}; \mathbb{R}^{n_{\boldsymbol{u}}})$ uses the uniform average nearness $\overline{\mu}$.

Proposition 32. *Assuming the conditions of Definition 31, the projection of an element of VS($\mathcal{S}$) in* $\mathrm{CBDS}(\mathcal{S}; \mathbb{R}^{n_{\boldsymbol{u}}})$*, in the sense of finding the best mean-square error approximant is*

$$\dot{y}^s_t = (\hat{s}^j_i \nabla^s_{vi} y^v_y)\omega^i_t + (\overline{\mu}\nabla^s_{vi} y^v_t)v^i_t,$$

Proof: We give the proof for the bilinear part (**M**) of the BDS dynamics ignoring the affine part (**N**) which is easily seen to be 0 in this case. The real dynamics of the camera is of the form $y^s_t = R^s_{iv}(t) y^v_t u^i_t$ where $R^s_{iv}(t) = \mu^s_t \nabla^s_{iv}$ contains the nearness as hidden state. We want to approximate this using the BDS dynamics $\dot{y}^s_t = M^s_{iv} y^v_t u^i_t$. The mean squared error between the two is

$$E(M) = \mathbb{E}_t\{\textstyle\int_x (\int_q \sum_j \left(M^x_{jq} - R^x_{jq}(t)\right) y^q_t u^j_t \,\mathrm{d}q)^2 \,\mathrm{d}x\}.$$

This is a convex function of M. The condition $\partial E/\partial M$ gives

$$\partial E/\partial M^s_{iv} = \mathbb{E}_t\{(\textstyle\int_q \sum_j (M^s_{jq} - R^s_{jq}(t)) y^q_t u^j_t \,\mathrm{d}q) y^v_t u^i_t\}.$$

If $\boldsymbol{y}$ and $\boldsymbol{u}$ are independent, letting $\mathbb{E}_t\{y^q_t y^v_t\} = P^{qv}$, $\mathbb{E}_t\{u^j_t u^i_t\} = Q^{ij}$, we get

$$\partial E/\partial M^s_{iv} = \textstyle\sum_j (\int_q M^s_{jq} P^{qv} - \mathbb{E}\{R^s_{jq}(t) y^q_t y^v_t\} \,\mathrm{d}q)\} Q^{ij}. \tag{24}$$

If $R^s_{jq}(t)$ is independent of $\boldsymbol{y}$, which is verified in the case of the camera because nearness and color are independent, then $\mathbb{E}\{R^s_{jq}(t) y^q_t y^v_t\} = \mathbb{E}\{R^s_{jq}(t)\} P^{qv} = \overline{R}^s_{jq} P^{qv}$, which implies

$$\partial E/\partial M^s_{iv} \quad = \quad \textstyle\sum_j (\int_q P^{vq} (M^s_{jq} - \overline{R}^s_{jq}) \,\mathrm{d}q Q^{ij}.$$

This is equivalent to $P^{vq}(M^s_{jq} - \overline{R}^s_{jq}) Q^{ij} = 0$ for all s, j, q, which, if $\mathbf{Q}, \mathbf{P} > 0$ implies $M^s_{jq} = \overline{R}^s_{jq}$. In the case of the camera dynamics for translation, we have $R^s_{jq}(t) = \mu^s_t \nabla^s_{jv}$, therefore $\overline{R}^s_{jq} = \overline{\mu}^s \nabla^s_{jv}$, and $\overline{\mu}^s = \overline{\mu}$ by Corollary 30. ∎

Range finders: Also in this case the instantaneous nearness is replaced by the average nearness.

Proposition 33. *Assuming the conditions of Definition 31, the projection of an element of VS($\mathcal{S}$) in* $\mathrm{CBDS}(\mathcal{S}; \mathbb{R}^{n_{\boldsymbol{u}}})$ *is*

$$\dot{y}^s_t = (\hat{s}^j_i \nabla^s_{vj} y^v_t)\omega^i_t + (\alpha\overline{\mu}\nabla^s_{vi} y^v_t - s^*_i)v^i_t$$

where α is a positive constant. (In our experiments α is close to 1 but this is not supported by theory.)

As in the previous proof, we focus on the dynamics of the translation part, which is $\dot{y}^s_t = (\nabla^s_{vi} \log y^v_t - s^*_i)\boldsymbol{v}^i_t$. Note that $\nabla^s_{vi} \log y^v_t = \frac{1}{y^s_t} \nabla^s_{vi} y^v_t$. This means that the range-finder is instantaneously approximated by BDS, with $R^s_{jq}(t) = \frac{1}{y^s_t} \nabla^s_{jv}$. Again, we encounter the nearness. We write $R^s_{jq}(t) = \mu^s_t \nabla^s_{jv}$. The proof is the same as for the camera, up until (24), where there is a slight complication. For both sensors, the term $R^s_{jq}(t)$ is the same. However, for the camera, the observations y are the luminance, which is independent of the nearness. Hence the expectation can be split as follows: $\mathbb{E}_t\{R^s_{jq}(t) y^q_t y^v_t\} = \mathbb{E}_t\{R^s_{jq}(t)\} \mathbb{E}_t\{y^q_t y^v_t\}$. However, for the range-finder, the observations y are the distance, which is the inverse of the nearness, so they are not independent. Rewrite everything with the distance: $y = \sigma, \mu = 1/\sigma$. Then, in general, the expectation cannot be split:

$$\mathbb{E}_t\left\{(1/\sigma^s_t)\nabla^s_{jv}\ \sigma^v_t \sigma^q_t\right\} \neq \mathbb{E}_t\left\{(1/\sigma^s_t)\right\} \nabla^s_{jv}\ \mathbb{E}\{\sigma^v_t \sigma^q_t\}.$$

However, for our scenario, it is true that, for some positive value α,

$$\mathbb{E}_t\left\{(1/\sigma^s_t)\nabla^s_{jv}\ \sigma^v_t \sigma^q_t\right\} = \alpha\overline{\mu}\nabla^s_{jv}\ \mathbb{E}\{\sigma^v_t \sigma^q_t\}. \tag{25}$$

for some positive value α. To see this, rewrite the expectation as $\mathbb{E}_t\left\{\nabla^s_{jv} \log \sigma^v_t \sigma^q_t\right\}$. The gradient is a linear operator that commutes with expectation. We then need to evaluate $\nabla^s_{jv} \mathbb{E}_t\left\{\log \sigma^v_t \sigma^q_t\right\}$. Based on Bussgang's theorem, the cross covariance of $\log\sigma$ and σ is a linear function of the autocovariance of σ; for some $\alpha > 0, \beta \in \mathbb{R}$,

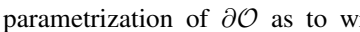
$$\mathbb{E}_t\left\{\log \sigma^v_t \sigma^q_t\right\} = \alpha\overline{\mu}\mathbb{E}_t\left\{\sigma^v_t \sigma^q_t\right\} + \beta 1^{vq}.$$

Here, 1^{vq} is the tensor with elements equal to 1. Taking into account that $\nabla^s_{jv} 1^{vq} = 0^{sq}_j$, we obtain (25).

VII. Learning experiments

This section shows examples of learning of BDS models for both simulations and real robots. Simulations implement exactly the sensor models described in the previous sections. Real experiments are used to check whether the results are robust with respect to unmodelled phenomena.

1) Simulations setup: There is an extremely large parameter space to explore: there is an infinite number of Vehicles to simulate. For the environment, not only we need to choose *a* map, but an entire probability distribution on maps. For the vehicles, there are many details in specifying the actuators and sensors (spatial discretization, levels of sensor noise, arrangement of pixels, etc.). Every small detail influences the results: for example, different environments change the learned **T** through the covariance matrix (Lemma 25), but the findings are qualitatively unchanged.

The vehicles inhabit procedurally generated maps. The obstacles $\mathcal{O}$ are a set of randomly generated circular obstacles, with density 0.15 circles/m^2 and radius distributed according to a uniform distribution on $[1\,\mathrm{m}, 3\,\mathrm{m}]$. (More complex shapes do not change the results.) The textures $\mathcal{T}$ are generated using "smoothed random checkerboards", which look like this:  . Let τ be a unidimensional parametrization of $\partial\mathcal{O}$ as to write the texture as a function $\mathcal{T}(\tau)$, which is equal to the convolution of a Gaussian kernel ($\sigma = 0.1\,m$) with a random process $t(\tau)$ that takes values in $\{0, 1\}$, It is defined as $t(\tau) = (d_i(\tau) \mod 2)$ where $d_i(\tau)$ counts the number of events in a Poisson process with intensity $\lambda = 10\,\mathrm{events/m}$. The field $\mathcal{F}(\boldsymbol{p})$ for the field-sampler is generated as a sum of randomly placed

sources: $\mathcal{F}(\boldsymbol{p}) = \sum_i \omega_i k(\|\boldsymbol{p} - \boldsymbol{p}_i\|)$), where $\boldsymbol{p}_i \in \mathbb{R}^2$ is randomly sampled with a density of 0.5 sources/m^2 and the kernel $k : \mathbb{R}^+_\circ \to \mathbb{R}^+_\circ$ is the function $k(d) = 1/(2\,d + 1)$.

The vehicles have the kinematics of a rigid body on $\mathrm{SE}(2)$ controlled in velocity. The maximum linear/angular velocity are 0.1 m/s and 0.1 rad/s. The commands are perturbed by additive Gaussian noise with $\sigma = 1\%$ of the range. Three *vehicles* equipped with different sensors were simulated:

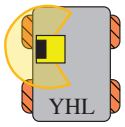
IDRF: This vehicle has an ideal range finder (FOV 360°, resolution 0.5 rays/°). For this and the other sensors, the frequency is 30 Hz and the noise is additive Gaussian with $\sigma = 1\%$ of the range.

IDCAM: This vehicle has an ideal vision sensor (FOV 360°; resolution 0.5 rays/°). Before sampling, the light field is smoothed with a Gaussian point-spread function with $\sigma = 1$.

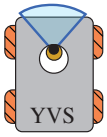
IDFS: This vehicle has an ideal field-sampler The sensels are placed in a regular 12×12 grid of radius 1 m.

The length of the simulated logs is ~1000 episodes for a total of ~800k total observations.

2) Experiments: The robotic platform we use is a Kuka *Youbot*. The 4 omnidirectional wheels allow to impose any velocity on $\mathsf{se}(2)$. Two *Hokuyo* URG-04LX range-finders [78] (FOV: ~270°; resolution: 2 rays/°; frequency 10 Hz) are mounted at the platform sides, approximately at pose $\langle 0.3\,\mathrm{m}, \pm 0.15\,\mathrm{m}, \pm 90° \rangle$ with respect to the platform center. A Logitech Orbit camera (FOV: ~60°; resolution: 320×240; frame rate: 10 Hz) is mounted at $\langle 0.02\,\mathrm{m}, 0\,\mathrm{m}, 0 \rangle$ with respect to the center of the base. Another camera (FOV: ~50°, resolution: 320×240; frame rate: 10 Hz) is mounted on the arm, fixed in such a way that the camera points upwards with axis orthogonal to the ceiling (Fig. 2a). The Kuka Youbot data was used in different *configurations* that emulate the three canonical sensors. Formally we can think of the Youbot as one point in the large set Robots, and these three configurations as three projections on the set Vehicles. For all configurations, the commands $\boldsymbol{u} = \langle v^1, v^2, \omega^3 \rangle$ are the kinematic velocities of the base.



YHL: The observations $\boldsymbol{y} \in [0,1]^{180}$ are the normalized ranges of the left Hokuyo. The field of view includes fixtures on the robot. The data corresponding to occlusions due to fixtures on the robot as well as out-of-range data (returned as "0" by the driver) are censored before learning. The YHLR configuration contains the concatenated readings from both range-finders subsampled to have 180 sensels in total.



YCF: A 1D vision sensor is simulated by taking the values of the pixels corresponding to a middle band (rows 100 to 140 in a 320×240 image) in the image returned by the Orbit camera. We use the luminance component as the signal: $y^s_t = 0.299\,r^s_t + 0.587\,g^s_t + 0.114\,b^s_t$ where $\langle r^s_t, g^s_t, b^s_t \rangle$ are the color components. (The analysis in Section III is valid for 2D sensors as well, but the BDS models cannot deal with full-frame 2D data because the complexity is $\mathcal{O}(n^2)$.)

YFS: This configuration simulates a field sampler by using a camera pointed upwards parallel to the ceiling. Because the ceiling is always level, there is no perspective effect at play. Therefore, the value reported by each pixel can be understood as sampling a planar function. With the geometry of our environment, the area sampled is ~ 2 m × 2 m. The luminance signal is first smoothed with a Gaussian kernel ($\sigma = 0.1\,m$) and then sampled with a grid of 12×12 pixels. LDR: The other platform used is an *iRobot Landroid* (prototype). The Landroid has two tracks controlled in velocity and a Hokuyo on board approximately mounted at $\langle 5\mathrm{mm}, 0, 0 \rangle$ with respect to the center.

The logs were taken in a typical robotic lab environment (Fig. 3). In the real experiments, the Youbot started always from the same position at the center of an arena of area approximately 12 m × 12 m and ceiling height ~9 m. The laboratory clutter was periodically rearranged every few logs. The Landroid data was taken in nearby corridors as well. The datasets contain relatively rare deviations from ideal conditions, such as moving people, occasional slippage, hardware faults (out-of-battery faults), and bugs-related collisions with the environment. Learning is robust to such occasional nuisances because all statistics in Algorithm 1 are computed as expectations.

The logs were recorded in ROS format. All data from the Youbot's ROS computation graph were recorded; for saving space, images where compressed to JPG. The total size of the data for the Youbot is ~10 GB for a length of ~6 hours. For the Landroid, the data is ~1 hour.

3) Exploration strategy: We generate trajectories by sampling random commands (*motor babbling*); this satisfies the hypotheses of Corollary 30 and ensure uniform exploration.

The most obvious choice for the choice of commands would perhaps be to sample a random command at each time step. The resulting trajectory would be a Brownian motion in if the configuration space is Euclidean space, or an appropriate generalization if the configuration is a Lie group [79]. Brownian motion is not an efficient way to explore environments as it tends to revisit places. A better alternative is to use a Levy flight-type trajectory, which are assumed to describe the behavior of foraging animals [80]. We choose the value of $\boldsymbol{u}_t$ is a realization of a stochastic process, which depends on two probabilities: a measure $p_{\boldsymbol{u}} \in \mathsf{Measures}(\mathcal{U})$ from which commands are sampled, and a measure $p_\Delta \in \mathsf{Measures}(\mathbb{R}^+_\circ)$ from which switching intervals are sampled. At the beginning of the k-th interval at time $t^{(k)}$, one samples a command $\boldsymbol{u}^{(k)} \sim p_{\boldsymbol{u}}$ and an interval $\Delta^{(k)} \sim p_\Delta$. The commands are set to be $\boldsymbol{u}_t = \boldsymbol{u}^{(k)}$ for the interval $t \in [t^{(k)}, t^{(k)} + \Delta^{(k)}]$. Then at $t^{(k+1)} = t^{(k)} + \Delta^{(k)}$, the process is repeated.

We used for p_Δ a uniform distribution in $[1\,\mathrm{s}, 60\,\mathrm{s}]$. For $p_{\boldsymbol{u}}$ we used a discrete set of values, so that each component could assume the values $\{-0.1, 0, +0.1\}$ (m/s or rad/s). (The BDS model does not need the commands to belong to discrete sets; however, we planned to use the datasets for other methods that might have this assumption.)

Figure 3. Exploration via motor babbling using Levy-flight-type trajectories. Composite image of robot poses at times $t = 0, 1$ min, 10 min, 60 min.

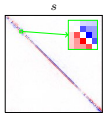
4) Safety supervisor: The bootstrapping agent does not have direct access to the robot velocities. Rather, the commands

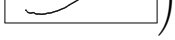
are filtered by a supervisor. The supervisor knows a priori the robotic platform and the map between uninterpreted commands $\boldsymbol{u}$ and the platform velocities. Based on the range-finder data, the supervisor judges a command unsafe if it would make the robot collide with an obstacles. An unsafe command is ignored and the platform is stopped until a safe command is received. In principle, we would want the agent to learn that certain actions are unsafe based on some "pain" feedback signal, but, with our current setup, neither the robot nor the lab would survive more than a handful of learning experiences.

5) Data visualization: Fig. 4 and successive show graphical representations of the learned tensors $\mathbf{P}$, $\mathbf{M}$, $\mathbf{N}$, $\mathbf{T}$, $\mathbf{U}$ for the various cases. The covariance tensors $\mathbf{P}$ are displayed as images (e.g. Fig. 4a). Positive entries are red (mnemonics: red = hot = positive), negative in blue, and white is 0. For each figure, the values are scaled so that the most positive values are pure red, so in general these figures only show the relative values and not the scale. When displaying the tensors the ordering of the sensels is important, though the agents are indifferent to a permutation of the sensels. The covariance shows whether the ordering of the sensels is correlated with the distance in the sensel space. For example, the covariance matrix of the range-finder (Fig. 5a) encodes the fact that the sensels are ordered in sensel space like they are in the vector $\boldsymbol{y}$, while the covariance of the field sampler reflects the fact that the sensels have a 2D grid disposition and are ordered by rows (Fig. 10a).

The tensors $\mathbf{M}$ and $\mathbf{T}$ are shown in 2D slices, each slice $\mathbf{M}_{\cdot i}$ and $\mathsf{T}^{\cdot\cdot i}$ corresponding to the i-th command. The colors use the same convention as the tensor $\mathbf{P}$. The tensors $\mathbf{N}$ and $\mathbf{U}$ are shown in slices corresponding to each command. Each slice is a one-dimensional function of s so it is plotted as a function of s. For example, the learned dynamics of the range-finder considering only the first command can be written as

$$\dot{y}^s_t = (\mathsf{M}^s_{v1} y^v + \mathsf{N}^s_1) u^1 = \left(\approx \right) y^v_t + \left(\right) u^1.$$





A. Findings

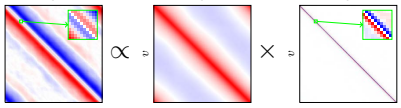
1) Uniform representation: The learned tensors are qualitatively similar for the ideal sensor in simulation (e.g., Fig. 4b) and the corresponding real sensor (e.g., Fig. 5b); this is a confirmation that the simple models introduced in Section III capture the dominant phenomena for real sensors even if some others phenomena were neglected (finite temporal and spatial sampling, value discretization).

2) Robustness: Sensels that do not follow the BDS model are simply ignored. For example, for the Landroid data, two antennas partially occlude the sensor, generating out-of-range or extremely noisy data. In the corresponding tensor $\mathbf{T}$ the entries corresponding to the occluded sensels have very small values (white stripes in Fig. 9b and Fig. 9c). By defining the concept of *usefulness* of a sensel [62] one can find quantitative criteria to find sensor faults without a reference model.

3) Sensitivity: These models exhibit sensitiveness to very subtle effects. For example, the covariance of the simulate camera (Fig. 7a) is a Toeplitz matrix (entries are constant for each diagonal), because its pixels are equidistant on the visual sphere and covariance is a function of the distance [53]. The covariance of the real camera (Fig. 8a) does not have this property: because of the perspective transform, pixels that are equidistant in image space are not exactly equidistant on the visual sphere, and the learned tensors are sensitive enough to pick up this effect..

4) Influence of environment and exploration statistics: The pictures confirm visually that the learned tensor $\mathbf{T}$ is equal to the product of $\mathbf{M}$ and $\mathbf{P}$ (Lemma 25). For the case of the range-finder (Fig. 5), the relation $\mathbf{T} = \mathbf{MPQ}$, assuming a diagonal $\mathbf{Q}$, can be visualized as follows:

$$\mathsf{T}^{sv3} \propto \mathsf{M}^s_{q3} \mathsf{P}^{qv} \quad \equiv \quad \approx \quad \propto \quad \approx \quad \times \quad \approx$$



The estimation step (17) in Algorithm 1 can be interpreted as removing the influence of the world statistics (covariance $\mathbf{P}$) and the exploration statistics (covariance $\mathbf{Q}$).

5) Composability: Data from multiple sensors can be used together just by concatenating the observations together. If the two sensors observations are uncorrelated, then there is nothing to gain. For example, luminance and distance are not predictive of each other, so if the data from a camera and a range-finder, the resulting tensors $\mathbf{T}$ is a block tensor where the the diagonal blocks are for the two original sensors, and the off-diagonal mixed blocks are zeros. If the data is correlated then there are off-diagonal blocks. For example, the field of view of the two range-finders used in our setup overlap for some readings, as can be seen from the covariance matrix (Fig. 6a). This overlap is automatically taken into account in the learned model as is shown by the off-diagonal patterns of the tensor $\mathbf{T}$ (Fig. 6b).

6) Invariance to reference frame: The dynamics of the canonical robotic tensors (summarized in Table I) were derived assuming that the commands are kinematic velocities relative to the sensor frame. If the sensors are off-centered, like in the case of the Hokuyo on the Youbot, the BDS approximation works just as well. The velocities of two reference frames which are linked together are related by a linear transform: if $g_r, g_s \in \mathsf{se}(3)$ are the velocity of the robot and the sensor, respectively, then there exists a matrix $\boldsymbol{A}$ such that $g = \boldsymbol{A} g_h$. This corresponds to a linear transform of the commands $\boldsymbol{u} \mapsto \boldsymbol{A}\boldsymbol{u}$. The BDS model family is invariant to linear transformations of the commands, therefore the model is able to represent just as well the sensor for any the reference frame. The resulting tensors are a linear combination of the tensors that would be obtained for canonical commands. For example, in the case of range-finders the slice of $\mathbf{N}$ corresponding to the angular velocity should be 0, as for the simulated vehicle (Fig. 4m), but in the case of the Youbot, N^s_2 is not identically zero, because a rotation of the platform also gives a slight translation of the sensor (Fig. 5m).

VIII. Servoing tasks

Models are only as good as the decisions they allow to make.

The task that we use in this paper is generalized servoing. Let $\boldsymbol{y}_\circ$ be some given "goal" observations. We define *servoing* as moving as to minimize $\|\boldsymbol{y} - \boldsymbol{y}_\circ\|$. This is a good task for bootstrapping because it can be stated for all sensor configurations; it is a basic skill for embodied agents; and it can be used as a building block to derive more complex behavior, such as navigation.

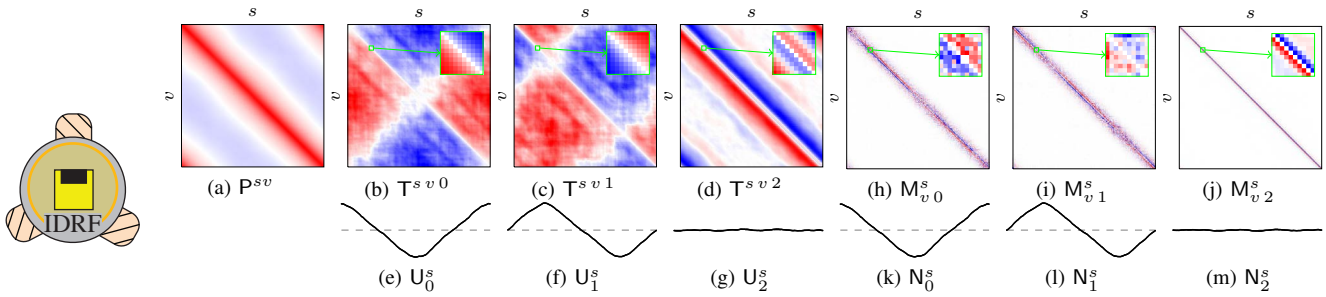


Figure 4. Learning results for the robot IDRF (ideal range-finder).

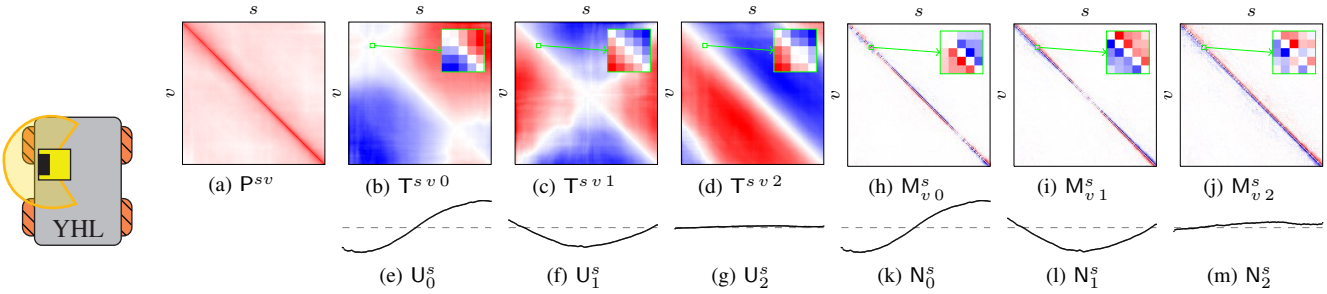


Figure 5. Learning results for the robot YHL (Youbout, left range-finder).

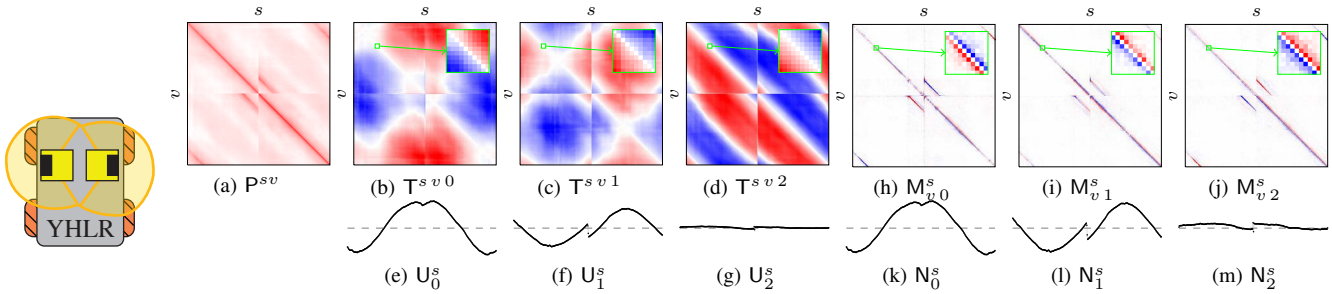


Figure 6. Learning results for the robot YHLR (Youbot, left + right range-finders)

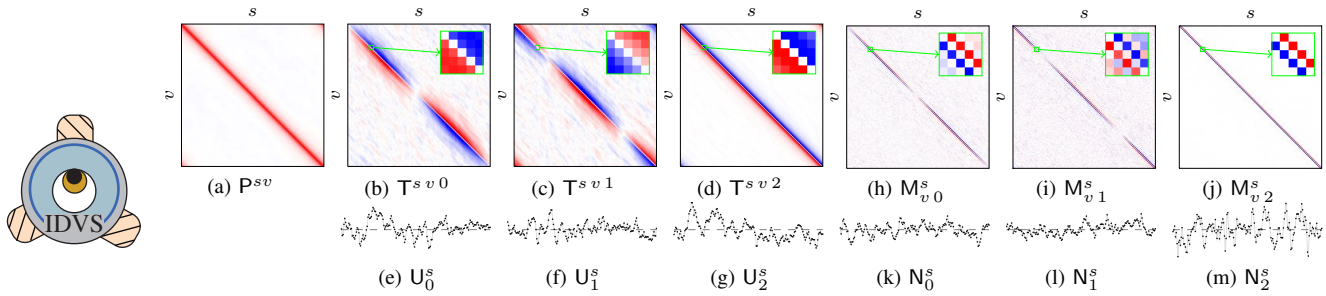


Figure 7. Learning results for the robot IDCAM (ideal camera) From the corners of the covariance in (a) we can see that this is a sensor with 360deg field of view. The expected value of the tensors **U** and **N** is 0; here the values are noisy estimates of 0.

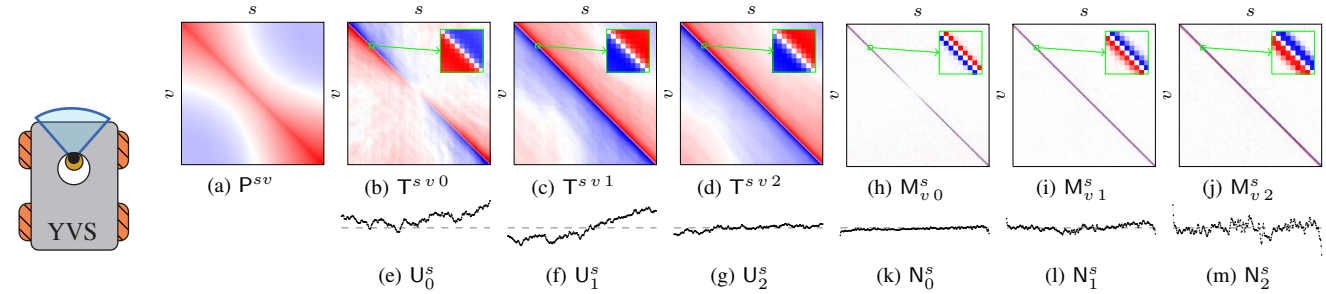


Figure 8. Learning results for the robot YCF (Youbot, frontal camera) From the hourglass shape of the covariance we can see that the sensels are not equispaced on the visual sphere (in fact, they are equispaced in image space).

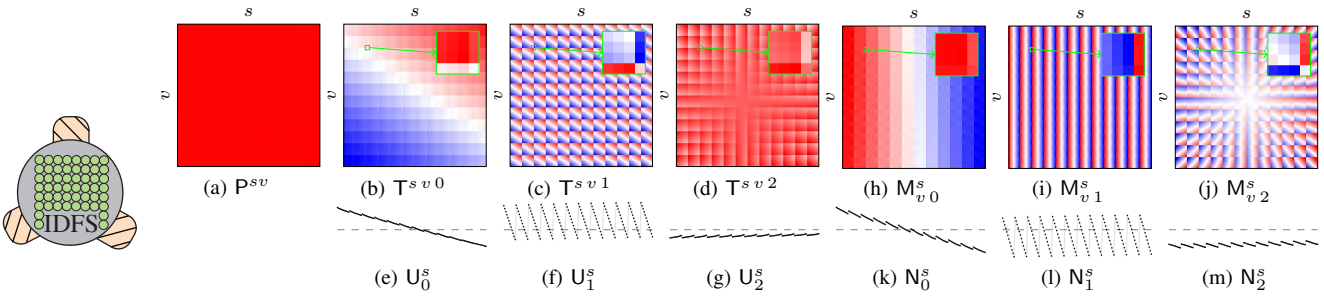


Figure 9. Learning results for the robot IDFS (ideal field sampler) The correlation between these sensels is almost 1 due to the fact that the sources are sparse in the environment.

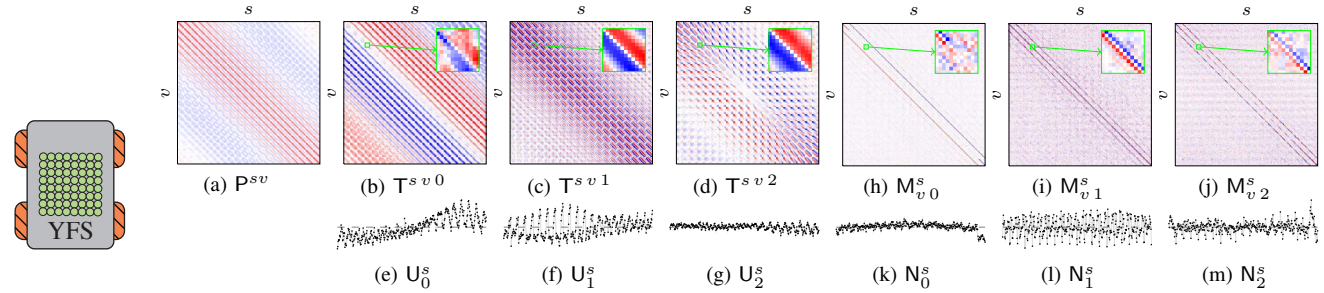


Figure 10. Learning results for the robot YCF (Youbot, upwards camera as field-sampler) The sensels are ordered by row...

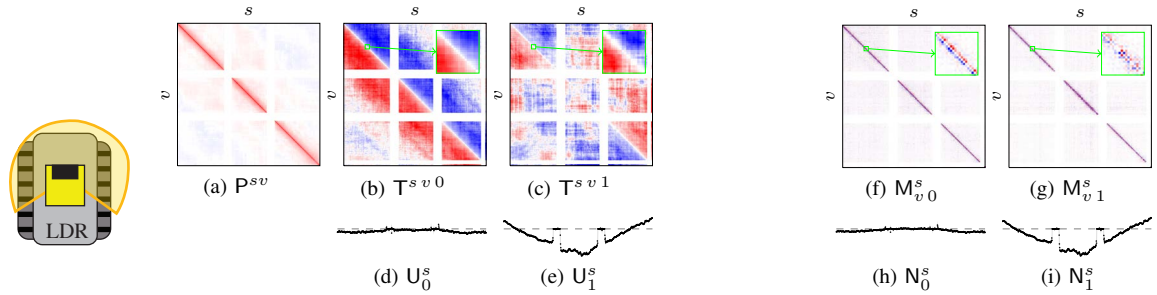


Figure 11. Learning results for the robot LDR (Landroid with Hokuyo)

The literature has many examples of the same problem, soled for one sensor modality and with a known model; see, e.g. [81], for a recent reference.

1) Gradient-descent based servoing for BDS models: A relatively simple to implement gradient-descent control law that minimizes $\|\boldsymbol{y}-\boldsymbol{y}_\circ\|$ can be derived for BDS models in the holonomic case and is given by (26) in the proposition below. It is computationally simple to implement because it is a bilinear function of the error $y^r - y^r_\circ$ and the current observations.

In general, there exists no *smooth* controller that stabilizes a goal state asymptotically for nonholonomic systems [82]. For the BDS dynamics (10), the system is holonomic if the linear operators $\{\mathsf{M}^s_{vi}y^v\}_{i=1}^{n_{\boldsymbol{u}}}$ form a nilpotent Lie algebra, which is taken as a necessary hypothesis for the following result.

Proposition 34. *Given the goal observations $\boldsymbol{y}_\circ$, a positive defined metric m_{rs} for $\boldsymbol{y}$ and r^{ij} for $\boldsymbol{u}$, the control law*

$$u^j = -\textstyle\sum_{r,s,v,i}\,(y^r - y^r_\circ)m_{rs}(\mathsf{M}^s_{vi}y^v + \mathsf{N}^s_i)r^{ij} \qquad (26)$$

corresponds to a descent direction of $\|\boldsymbol{y}-\boldsymbol{y}_\circ\|_m$.

Moreover, if the operators $\{\mathsf{M}^s_{vi}y^v\}_{i=1}^{n_{\boldsymbol{u}}}$ form a nilpotent Lie algebra, the point $\boldsymbol{y}=\boldsymbol{y}_\circ$ is asymptotically stable.

Proof: Define the error signal $e^r = y^r - y^r_\circ$ and write the error function as $V = \frac{1}{2}e^r m_{rs} e^s$. The derivative of V is

$$\begin{aligned}\dot V &= \dot e^r m_{rs} e^s = \dot y^r m_{rs} e^s = [(\mathsf{M}^r_{qj}y^q + \mathsf{N}^r_j)u^j]m_{rs}e^r \\ &= -e^t m_{ts}(\mathsf{M}^s_{vi}y^v + \mathsf{N}^s_i)r^{ij}(\mathsf{M}^r_{qj}y^q + \mathsf{N}^r_j)m_{rs}e^r.\end{aligned}$$

Defining a vector $g_i = e^t m_{ts}(\mathsf{M}^s_{vi}y^v + \mathsf{N}^s_i)$ we obtain $\dot V = -g_i r^{ij} g_j \le 0$, which shows that V never increases. If we prove that g is never 0 in a neighborhood of $\boldsymbol{y}_\circ$, then $\dot V < 0$, and V is then a Lyapunov function, making $\boldsymbol{y}_\circ$ asymptotically stable. In general, because the Lie algebra is nilpotent of order 0, any solution of (10) can be written in the form

$$\boldsymbol{y} = \exp(\mathsf{M}^s_{v1}b^1)\exp(\mathsf{M}^s_{v2}b^2)\cdots\exp(\mathsf{M}^s_{vn_{\boldsymbol{u}}}b^{n_{\boldsymbol{u}}})\boldsymbol{y}_\circ, \qquad (27)$$

where the $\{b^j\}_{j=1}^{n_{\boldsymbol{u}}}$ are the *Philip Hall coordinates*, and exp represents the exponential of a linear operator [83, Chapter 11, page 540]. Near $\boldsymbol{y}_\circ$ each term can be linearized as

$$\exp(\mathsf{M}^{\cdot}_{\cdot j}b_j) = \mathbf{I} + \mathsf{M}^{\cdot}_{\cdot j}b^j + o(\|\boldsymbol{b}\|^2).$$

The linearized version of (27) is then

$$y^s = y^s_\circ + \mathsf{M}^s_{vj}b^j y^v_\circ + o(\|\boldsymbol{b}\|^2).$$

From this we get $e^s = \mathsf{M}^s_{vj}y^v_\circ b^j + o(\|\boldsymbol{b}\|^2)$, which we substitute

in the definition of g_i to obtain

$$\begin{aligned} g_i &= y_\circ^v \mathsf{M}_{vi}^s m_{rs} e^r + o(\|\boldsymbol{b}\|^2) \\ &= y_\circ^v \mathsf{M}_{vi}^s m_{rs} (\mathsf{M}_{qj}^r y_\circ^q b^j) + o(\|\boldsymbol{b}\|^2). \end{aligned}$$

In a neighborhood of $\boldsymbol{y}_\circ$, but not precisely at $\boldsymbol{y}_\circ$, the vector $\boldsymbol{b}$ is nonzero; otherwise, from (27) we get $\boldsymbol{y} = \boldsymbol{y}_\circ$. Assuming that the $\mathsf{M}_{qj}^r y_\circ^q$ commute, we also have that $\mathsf{M}_{qj}^r y_\circ^q b^j \neq 0^r$. In fact, if they commute, we can write (27) as $\boldsymbol{y} = \exp(\mathsf{M}_{qj}^r y_\circ^q b^j)\boldsymbol{y}_\circ$ and the argument of the exponential must be different from 0. These two together imply that $g_i = y_\circ^v \mathsf{M}_{vi}^s m_{rs} (\mathsf{M}_{qj}^r y_\circ^q b^j) \neq 0$ near the origin. Here the abundance of indices is masquerading the simplicity of the assertion. Without indices: suppose there is a vector $\boldsymbol{v} \neq 0$, and a linear operator A such that $A\boldsymbol{v} \neq 0$; then $A^* A\boldsymbol{v} \neq 0$. In this case, $\boldsymbol{v} = \boldsymbol{b}$ and $A_i^v = y_\circ^v \mathsf{M}_{vi}^s$. ■

2) Servoing based on longer-horizon prediction: The gradient descent strategy is myopic: the agent might be trapped in local minima of the cost function. In our setup, this happens for the camera (YCF), because the environment texture has much higher frequency than the geometry: the change of the camera observations for a small motion is much larger than the change of observations for a range-finder.

One way to avoid myopia is to use receding-horizon control: we choose the command that minimize the dissimilarity between goal and prediction over a long time step Δ (gradient descent is the limit as $\Delta \to 0$). Let $\varphi(\boldsymbol{y}, \boldsymbol{u}, \Delta)$ be the predicted observations after an interval Δ for a constant command $\boldsymbol{u}$. We choose the command that achieves the minimum dissimilarity with the goal observations:

$$\boldsymbol{u}_t^\star = \min_{\boldsymbol{u} \in U} \|\varphi(\boldsymbol{y}_t, \boldsymbol{u}, \Delta) - \boldsymbol{y}_\circ\|. \tag{28}$$

In the case of the BDS models we can compute the prediction in closed form if the commands are kept constant. In fact, suppose that the commands are fixed to $\boldsymbol{u}_t = \overline{\boldsymbol{u}}$ for an interval $[t, t+\Delta]$. Then during that interval, the dynamics are $\dot{y}_t^s = (\mathsf{M}_{vi}^s y_t^v + \mathsf{N}_i^s)\overline{u}^i$. Rearranging we can write it as $\dot{y}_t^s = (\mathsf{M}_{vi}^s \overline{u}^i) y_t^v + (\mathsf{N}_i^s \overline{u}^i)$, which is of the form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ for $\boldsymbol{A} = \mathsf{M}_{vi}^s \overline{u}^i$ and $\boldsymbol{b} = \mathsf{N}_i^s \overline{u}^i$. This is a linear-time-invariant dynamics with fixed input, hence the solution is

$$\varphi(\boldsymbol{y}_t, \overline{\boldsymbol{u}}, \Delta) = \exp(\Delta \mathsf{M}_{vi}^s \overline{u}^i) y_t^v + (\int_0^\Delta \exp((\Delta - \tau)\mathsf{M}_{vi}^s \overline{u}^i)\, \mathrm{d}\tau) \mathsf{N}_i^s \overline{u}^i$$

For the camera, the second term is zero ($\mathbf{N} = 0$) and the matrix exponential can be interpreted as a diffeomorphism applied to the image. In this paper we are just going to iterate over all commands in a subset of the available commands. The more efficient solution of the problem formalized as graph search in observations space [84] is outside the scope of this paper.

IX. Robustness to model approximations

We have computed the approximations in a certain model class (BDS) for our vehicles. These approximations neglect aspects of the dynamics (e.g., occlusions) as well as hidden states. This section shows that those approximations are good enough for the task of servoing. The approach is to show that if the approximated dynamics are "close enough" to the real but unknown dynamics, then a control strategy that works for the learned system will work for the real system. Sufficient conditions for convergence are obtained as a function of the statistics of the environment.

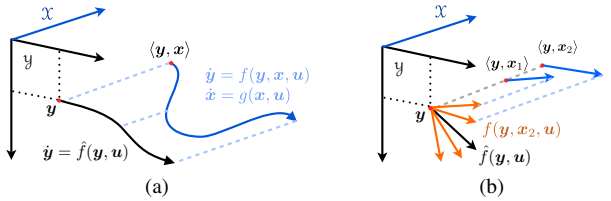


Figure 12. The learned dynamics $\dot{\hat{\boldsymbol{y}}} = \hat{f}(\boldsymbol{y}, \boldsymbol{u})$ is defined on the space of observations $\mathcal{Y}$ and neglects the dependence on the hidden state $\boldsymbol{x} \in \mathcal{X}$. The learned dynamics can be seen as the projection (in the maximum likelihood sense) of the true dynamics in the space $\mathcal{Y} \times \mathcal{X}$ to the observations space $\mathcal{Y}$.

1) A sufficient condition for convergence: We have considered instantaneous models of the form

$$\dot{\hat{\boldsymbol{y}}}_t = \hat{f}(\boldsymbol{y}_t, \boldsymbol{u}_t), \tag{29}$$

However, the real dynamics of $\boldsymbol{y}$ is different from $\hat{f}$, mostly because it depends on an unknown hidden state $\boldsymbol{x}$ that is not modeled, and has a dynamics of its own:

$$\begin{cases} \dot{\boldsymbol{y}}_t &= f(\boldsymbol{y}_t, \boldsymbol{x}_t, \boldsymbol{u}_t), \\ \dot{\boldsymbol{x}}_t &= g(\boldsymbol{x}_t, \boldsymbol{u}_t). \end{cases} \tag{30}$$

We look at the dynamics $\hat{f}$ as a projection on the space $\mathcal{Y}$ of the real dynamics f that exists in the space $\mathcal{Y} \times \mathcal{X}$ (Fig. 12a). When a point $\boldsymbol{y}$ is fixed, the real dynamics $f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{u})$ is different from the approximated dynamics $\hat{f}$ as the hidden state $\boldsymbol{x}$ varies (Fig. 12b). Intuitively, if the two dynamics are "close", then a control strategy for one should work for the other as well. "Closeness" is defined using the relative error between the norm of $\hat{f}$ and the norm of $f - \hat{f}$.

Definition 35. The dynamics $\hat{f}$ and f are a-close if there exists $a < 1$ such that, for all $\boldsymbol{x}$ and $\boldsymbol{u}$,

$$\|\hat{f}(\boldsymbol{y}, \boldsymbol{u}) - f(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{x})\| \leq a \|\hat{f}(\boldsymbol{y}, \boldsymbol{u})\|. \tag{31}$$

If this condition is satisfied, then the control law for the approximated system works also for the real system.

Proposition 36. *Let $\boldsymbol{u}_t^\star = \boldsymbol{u}^\star(\boldsymbol{y}_\circ, \boldsymbol{y}_t)$ be a control law makes the system $\dot{\boldsymbol{y}}_t = \hat{f}(\boldsymbol{y}_t, \boldsymbol{u}_t^\star)$ asymptotically stable in a neighborhood of $\boldsymbol{y}_\circ$. Assume that $\hat{f}$ and $\boldsymbol{u}^\star$ are smooth. If the bound (31) holds, then the same control law makes the system (30) asymptotically stable at $\boldsymbol{y}_\circ$.*

Proof: Close the loop and consider the dynamics of the autonomous system:

$$\dot{\boldsymbol{y}}_t = \hat{f}(\boldsymbol{y}_t, \boldsymbol{u}^\star(\boldsymbol{y}_\circ, \boldsymbol{y}_t)). \tag{32}$$

By assumption, $\boldsymbol{y}_\circ$ is asymptotically stable, so $\hat{f}(\boldsymbol{y}_\circ, \boldsymbol{u}^\star(\boldsymbol{y}_\circ, \boldsymbol{y}_\circ)) = 0$ and from (31) it follows that $\boldsymbol{y}_\circ$ is an equilibrium point also for the real dynamics.

Asymptotical stability can be proved using a Lyapunov argument. Because $\boldsymbol{y}$ is stable for the approximated dynamics, there exists a corresponding Lyapunov function (actually, many). We construct one which is convenient for our goals. Let N be a small compact neighborhood around $\boldsymbol{y}_\circ$. We construct a Lyapunov function for the set N [85]. Because $\boldsymbol{y}_\circ$ is asymptotically stable, for any point outside N, it takes a finite amount of time to arrive in N (while it could take infinite time

to get to the goal). Consider the function $V(\boldsymbol{y})$, defined as the time it takes for the solution of (32) to arrive in N:

$$V(\boldsymbol{y}) \triangleq \min_t \{\Phi_{\hat{f}}(\boldsymbol{y}; t) \in N\}. \tag{33}$$

Here, $\Phi_{\hat{f}}(\boldsymbol{y}; t)$ is the flow of the dynamics $\hat{f}$. This function is differentiable because the control law and the dynamics are smooth. Moreover, this function satisfies $\dot{V}(\boldsymbol{y}) = -1$, which means, geometrically, that the flow $\hat{f}(\boldsymbol{y}, \boldsymbol{u}^\star)$ is exactly perpendicular to the level sets of V:

$$\dot{V}(\boldsymbol{y}) = \frac{\partial V}{\partial \boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{u}^\star) = -1. \tag{34}$$

For any three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, the constraints $\langle \mathbf{a}, \mathbf{b} \rangle = -1$ and $\|\mathbf{b} - \mathbf{c}\| < \|\mathbf{c}\|$ implies that $\langle \mathbf{a}, \mathbf{c} \rangle < 0$. Applying this to the vectors $\mathbf{a} = \partial V / \partial \boldsymbol{y}$, $\mathbf{b} = \hat{f}(\boldsymbol{y}, \boldsymbol{u}^\star)$, $\mathbf{c} = f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{u}^\star)$ and using (34) and (31) one finds that $\dot{V}$ is negative along the trajectories of the real system: $\frac{\partial V}{\partial \boldsymbol{y}} f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{u}^\star) < 0$. This implies that V is a Lyapunov function for the set N with reference to the *real* dynamics. This implies that the system reaches the boundary of the set N in finite time. Because the size of N can be chosen arbitrarily small, the system is asymptotically stable. ∎

2) Vision sensor: convergence in spite of hidden states: The dynamics for pure rotations is purely bilinear for cameras and range-finders, while there is an approximation regarding the dynamics for translation (Table II). The worst case is pure translation ($\mathcal{Q} = \mathbb{R}^3$).

Proposition 37. *For pure translation, the control (26) applied to an element of VS($\mathcal{S}$) local convergence if $0 < \mu_t^s < 2\overline{\mu}$, which can be interpreted as: a) There is no object at infinity ($\mu_t^s > 0$); and b) There are no obstacles much closer than the average obstacle ($\mu_t^s < 2\overline{\mu}$).*

Proof: The camera dynamics for pure translation is $\dot{y}_t^s = \mu_t^s \nabla_{vi}^s y_t^v u_t^i$. The approximated model is $\hat{\dot{y}}_t^s = \overline{\mu} \nabla_{vi}^s y_t^v u_t^i$ where $\overline{\mu}$ is the average nearness. The difference between the two dynamics $\delta = \dot{y}_t^s - \hat{\dot{y}}_t^s$ has norm $\|\delta\|^2$ equal to $\int \| (\mu_t^s - \overline{\mu}) \nabla_{vi}^s y_t^v u_t^i \|^2 \, \mathrm{d}s$. By rewriting $(\mu^s - \overline{\mu})$ as $(\mu^s/\overline{\mu} - 1)\,\overline{\mu}$, we find the bound $\|\hat{f} - f\|^2 \leq \|\mu^s/\overline{\mu} - 1\|_\infty^2 \|\hat{f}\|^2$, which is in the form (31) with $a = \max_s (\mu_t^s/\overline{\mu} - 1)^2$. The condition $a < 1$ is satisfied if $0 < \mu_t^s < 2\overline{\mu}$. ∎

3) Range finder: convergence in spite of nonlinearities: We derive an analogous result for the range-finder.

Proposition 38. *For pure translation and a 360°viewport, applied to a range-finder a sufficient condition for local convergence of the control law is*

$$\|1 - \mu_t/\overline{\mu}\|_\infty \frac{\|\nabla \boldsymbol{y}\|_2}{\overline{\mu}^{-1} - \|\nabla \boldsymbol{y}\|_2} < 1,$$

which can be interpreted as: a) The environment is large enough (as $\mu \to 0$, the left-hand side goes to 0 as well); or b) The environment is close circular ($\|\nabla y\| \to 0$).

Proof: The dynamics for a range finder for translation is $\dot{y}_t^s = (\mu_t^s \nabla_{vi}^s y_t^v - s_i^*)\, \boldsymbol{v}_t^i$, where μ is the nearness (and $\mu = 1/y$). The approximated BDS model is $\dot{y}_t^s = (\mu_t^s \nabla_{vi}^s y_t^v - s_i^*)\, \boldsymbol{v}_t^i$, hence the deviation between the two is$\delta_t^s = (\overline{\mu} - \mu_t^s) \nabla_{vi}^s y_t^v u^i$. The deviation becomes small as $\|\nabla y\| \to 0$ or as $|\mu_t^s - \overline{\mu}| \to 0$. Because y in this case is the distance, $\|\nabla y\| \to 0$ implies that the environment tends to a circular environment of any size, while $|\mu_t^s - \overline{\mu}| \to 0$ means that the environment should tend to circular of a particular radius $1/\overline{\mu}$. A lower bound on $\|\hat{f}\|_2^2$ can be found as follows:

$$\begin{aligned}
\|\hat{f}\|_2^2 &= \| (\overline{\mu} \nabla_{vi}^s y_t^v - s_i^*)\, u^i \|_2 = \| \overline{\mu} \nabla_{vi}^s y_t^v u^i - s_i^* u^i \|_2. \\
&\qquad (\|\boldsymbol{a} + \boldsymbol{b}\| \geq |\|\boldsymbol{a}\| - \|\boldsymbol{b}\||) \\
&\geq |\|s_i^* u^i\|_2 - \overline{\mu} \|\nabla_{vi}^s y_t^v u^i\||. \\
&\qquad \text{(for small } \nabla y \text{ we choose one side)} \\
&= \|s_i^* u^i\|_2 - \overline{\mu} \|\nabla_{vi}^s y_t^v u_t^i\| \\
&\qquad (\|\boldsymbol{A}\boldsymbol{b}\| \leq \|\boldsymbol{A}\| \|\boldsymbol{b}\|) \\
&\geq \|s_i^* u^i\|_2 - \overline{\mu} \|\nabla \boldsymbol{y}_t\| \|\boldsymbol{u}_t\|
\end{aligned}$$

An upper bound for $\|\delta\|$ is the following:

$$\|\delta_t^s\|_2 \leq \|\overline{\mu} - \mu_s^t\|_\infty \|\nabla_{vi}^s y_t^v u^i\|_2^2 \leq \|\overline{\mu} - \mu_s^t\|_\infty \|\nabla \boldsymbol{y}_t\|_2 \|\boldsymbol{u}_t\|_2.$$

Using these bounds, the ratio of $\|\delta_t^s\|_2^2$ and $\|\hat{f}\|_{2;S}^2$ can be bounded as

$$\frac{\|\delta f_t\|_2}{\|\hat{f}_t\|_{2;S}} \leq \|1 - \mu_s^t/\overline{\mu}\|_\infty^2 \frac{\|\nabla \boldsymbol{y}\|_2 \|\boldsymbol{u}\|_2}{\overline{\mu}^{-1} \|s_i^* u^i\|_2 - \|\nabla \boldsymbol{y}\|_2 \|\boldsymbol{u}\|_2}.$$

The quantity $\|s_i^* u^i\|_2$ depends on the shape of the sensor. If the viewport is 360°, then $\|s_i^* u^i\|_2 = \|\boldsymbol{u}\|_2$ which can be proved by evaluating the integral $\|s_i^* u^i\|_2^2 = \frac{1}{2\pi} \int_{\mathcal{S}} (s^* u)^2 \, \mathrm{d}s$. The bound is then of the form $\|\delta\| \leq a \|\hat{f}_t\|$ with $a = \|1 - \mu_s^t/\overline{\mu}\|_\infty \|\nabla \boldsymbol{y}\|_2 / (\overline{\mu}^{-1} - \|\nabla \boldsymbol{y}\|_2)$. ∎

4) Convergence for non-ideal vehicles: Taken together, the previous results show sufficient conditions under which the gradient-descent control law (26) realizes the servoing behavior for the set of idealized Vehicles (Definition 11). Here we discuss what happens when nuisances are introduced, such as spatiotemporal discretization and noise.

Regarding temporal discretization, let $\mathrm{Zoh}_\Delta : \mathcal{D}(\mathcal{Y}; \mathcal{X}; \mathcal{U}) \to \mathcal{D}(\mathcal{Y}; \mathcal{X}; \mathcal{U}; \Delta)$ be the *zero-order hold* operator that transforms a continuous-time system into a discrete time system with interval Δ. Regarding spatial sampling, given a set of $n_{\boldsymbol{y}}$ points $\{s^i\}_{i=1}^{n_{\boldsymbol{y}}} \subset \mathcal{S}$, we can map a system in $\mathcal{D}(\mathrm{Images}(\mathcal{S}); \mathcal{U})$ into one in $\mathcal{D}(\mathbb{R}^{n_{\boldsymbol{y}}}; \mathcal{U})$. Call $\mathrm{Sample}_{n_{\boldsymbol{y}}}$ the set of all such maps with $n_{\boldsymbol{y}}$ sampling points. Regarding noise, we can define the set Noise_n as the set of additive white gaussian disturbances.

Definition 39. The set of *reasonably perturbed vehicles* $\tilde{\mathsf{Vehicles}} \in \mathcal{D}(\mathbb{R}^{n_{\boldsymbol{y}}}; \mathsf{Maps}; \mathbb{R}^{n_{\boldsymbol{u}}}; \Delta)$ is created from Vehicles by adding spatial/temporal discretization and noise:

$$\begin{aligned}
\tilde{\mathsf{Vehicles}} \triangleq \big\{ & N_1 \circ S \circ \mathrm{Zoh}_\Delta(\boldsymbol{D}) \circ N_2 \mid \boldsymbol{D} \in \mathsf{Vehicles}, \\
& N_1 \in \mathrm{Noise}_{n_{\boldsymbol{y}}},\ N_2 \in \mathrm{Noise}_{n_{\boldsymbol{u}}}, S \in \mathrm{Sample}_{n_{\boldsymbol{y}}},\ \Delta > 0 \big\}.
\end{aligned}$$

Regarding temporal and spatial discretization, we can directly use the result of Proposition 36, as follows:

- As the spatial sampling tends to zero, the dynamics of the sampled system tends to the ideal system. Therefore, there exists a minimum sampling for which there is convergence.
- As the temporal sampling tends to zero, the dynamics of the ZOH system tends to the ideal system. Therefore, there exists a minimum sampling for which there is convergence.

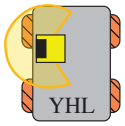
For the noise, instead, completely different results should be used: the concept of asymptotical stability does not apply, and must be replaced with appropriate stochastic equivalents. In our case, where we have proved stability with a Lyapunov argument, it is easy to adapt the statement to the case of

small additive noise on observations and commands and prove *positive recurrence* of a neighborhood of $\boldsymbol{y}_\circ$, meaning that the robot is attracted and remains around in a neighborhood of what was the asymptotical equilibrium [86].

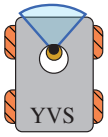
X. Servoing and navigation experiments

This section evaluates the convergence ratio and dynamic behavior of the control laws derived in the previous sections. The *same* agent is able to do the *same* task with *different* sensor configurations; but, obviously, with different performance based on the sensor. The results in terms of convergence are generally better than the results of the theory provide: while we proved *local* convergence, convergence regions appear to be quite robust and limited by hard factors (like the field of view of the sensor) rather than approximations in the learned model.

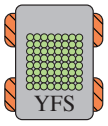
1) Evaluation of convergence basins: Fig. 13 shows the convergence basin of the control laws, for each of the sensor configurations, and three test environments, different from the training environment. The first two environments are convex with no occlusions (the first is shown in Fig. 2); the third has several occlusions. To create these figures, the robot was programmed to move in a grid with fixed heading. A circle indicates the goal position at which the goal observations $\boldsymbol{y}_\circ$ were taken. Fig. 13*a* and similar show the dissimilarity function $\|\boldsymbol{y}(\boldsymbol{q}) - \boldsymbol{y}_\circ\|$ as a function of the pose $\boldsymbol{q}$. More precise sensors, like the range-finder (panel *a*) have more peaked response than the others (panels *m*, *u*). Fig. 13*b* and similar show the velocities generated by the control law (26) in the plane; green arrows correspond to actions that lead to smaller dissimilarity.



YHL: The range-finder configuration has global convergence in the two convex environments (panels *b, d*), while the occlusions results in local minima (panel *f*). Using both range-finders one has better convergence and the local minima disappear (panel *l*): a bootstrapping agent does better with more data thrown at it, with no extra design effort required.



YCF: For this configuration the dissimilarity has many local minima (panels *m*, *ñ*). The gradient descent law(26) gets trapped in local minima (panels *n, o, q*). The prediction-based control (28), with $\mathcal{U}$ discretized in 32 values and planning horizon $\Delta = 2\,\mathrm{s}$ has larger convergence basin (panels *r, s, t*). These two controllers use the same learned models, but one is much more complex: the trade-off of complexity vs performance exists but is independent of the issues of learning.



YFS: The field sampler data has larger convergence basin (panel *x*), mostly limited by the sensor's FOV. The wrong directions generated near the goal (panel *v*) are due to sensor noise, which is dominant near the goal.

The figures do not show the convergence basin for rotations. For range-finder data (YHL), the basin size is ~40°, limited by local minima of the cost function. For the camera data (YCF) the convergence is lower, in the order of ~30°, and limited by the sensor's field of view (60°). For the field sampler (YFS), the convergence is as high as 80°, because the rotation does not make the scene exit the field of view.

The question of exactly how much data is needed for learning does not find a complete answer in this paper: all theoretical results are for asymptotic regime. To given an empirical investigation, Fig. 15 shows the convergence basins for configuration YHLR as a function of the size of the training data. The task is successful long before the mode model itself converges; i.e. far from the asymptotic regime.

2) Navigation experiments: The accompanying video shows several experiments of servoing. The behavior is in line with the expectations for a gradient-descent strategy, like tolerance to slight disturbances such as partial sensor occlusions and moving objects. Here, we comment on other experiments that use servoing as a building block to create the more complex behavior of *navigation*. Navigation in observations space is achieved by solving a repeated sequence of servoing problems on a map of observation snapshots. This task has been demonstrated in the past only for one sensor modality and with a known model [87, 88].

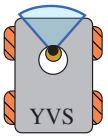
In an offline phase, the robot is driven along a closed trajectory. The trajectory is sampled at poses $\boldsymbol{q}_j$, $j \in \{1, \dots, N\}$, corresponding to the observations $\boldsymbol{m}_j$. The task is thus specified by the sequence of observations vectors: $\{\boldsymbol{m}_j\}_{j=1}^N$. Fig. 2 shows the approximate trajectory used for the experiments and the resulting waypoints map for the three configurations. The heading is kept fixed, but during execution the agent can also rotate. The distance between snapshots is ~0.1 m.

The navigation controller works by setting the goal observations $\boldsymbol{y}_\circ$ to be a few steps ahead in the map, so that the agent executes the navigation task by trying to converge to a goal which is constantly moving. More in detail, the navigation controller keeps as part of the state an index j_t indicating the current "position" in the map. At the beginning, the index j_{t_0} is set to be the closest map point in observations space: $j_{t_0} = \arg\min_j \|\boldsymbol{y}_{t_0} - \boldsymbol{m}_j\|$. During operation, the goal observations $\boldsymbol{y}_\circ$ are set to the value $\boldsymbol{y}_\circ = \boldsymbol{m}_g$, where $g = (j_t + d) \mod N$ is the index of the waypoint which is $d \geq 1$ steps ahead. If the observations are closer than a threshold c to the next observations in the sequence $\boldsymbol{m}_{j+1}$, the index j is incremented ($\mod N$). This ensures that the overall trajectory is smoothly executed, because the agent never converges: once it is close enough, the goal is advanced.

To make the results easier to compare across configurations, the commands generated were scaled linearly up to a maximum linear and angular velocity, respectively 0.1 m/s and 0.1 rad/s.

In the experiments we used $d = 3$. The convergence threshold c was slightly different for each configuration, to account for the fact that the different sensors configurations have different noise levels. Panels *b, f, j* show the observations used as waypoints. Panels *c, g ,k* show the observations y_t^s (horizontal axis: time t, vertical axis, sensel s). The red bar marks when the system was perturbed by occluding the sensors. Panels *d, h, l* show the dissimilarity $\|\boldsymbol{m}_j - \boldsymbol{y}_t\|$ between the observations $\boldsymbol{y}_t$ and all waypoints. The dissimilarity shows which part of the trajectory are more or less constrained.

YHLR: The trajectory for this configuration was the most repeateable (Fig. 14*a*) because of the sensor precision, as seen by the peaked dissimilarity. Note that the actual trajectory differs from the original square; the robot "cuts corners" because the goal is set d waypoints away.



YCF: The dissimilarity is much less peaked for the camera observations (panel *h*). In fact the pose is almost unobservable, especially on the sides of the trajectory. Therefore the trajectory

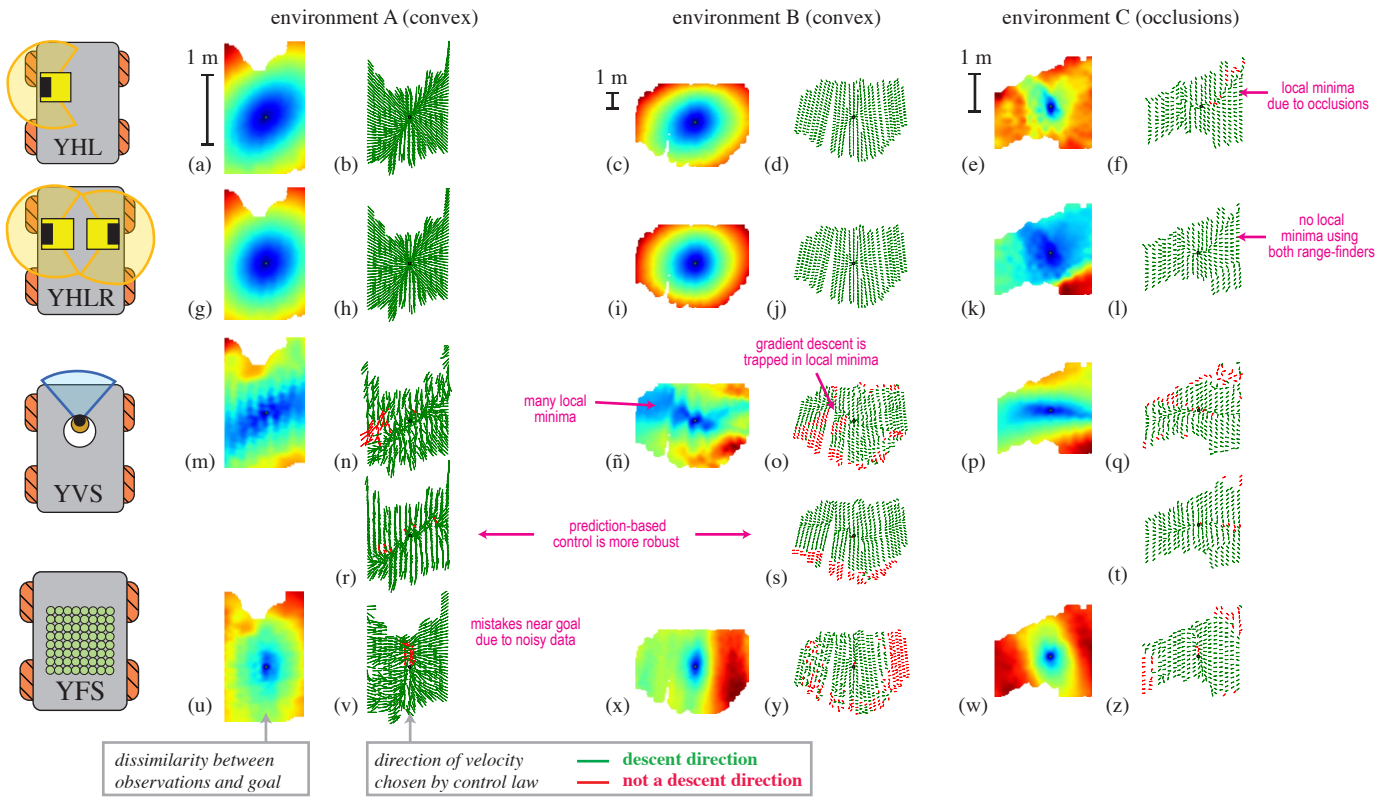


Figure 13. Attraction regions for the servoing task for different configurations and environments. The colormap in panels *a, c, e,* etc. show the dissimilarity $\|\boldsymbol{y} - \boldsymbol{y}_\circ\|$, which is different for each sensor. The arrows in panels *b, d, f,* etc. show the directions according to the control law (26), except for *r, s, t* which are for the prediction-based control law (28). Arrows are green if the corresponding motion minimizes the dissimilarity, and red otherwise.w

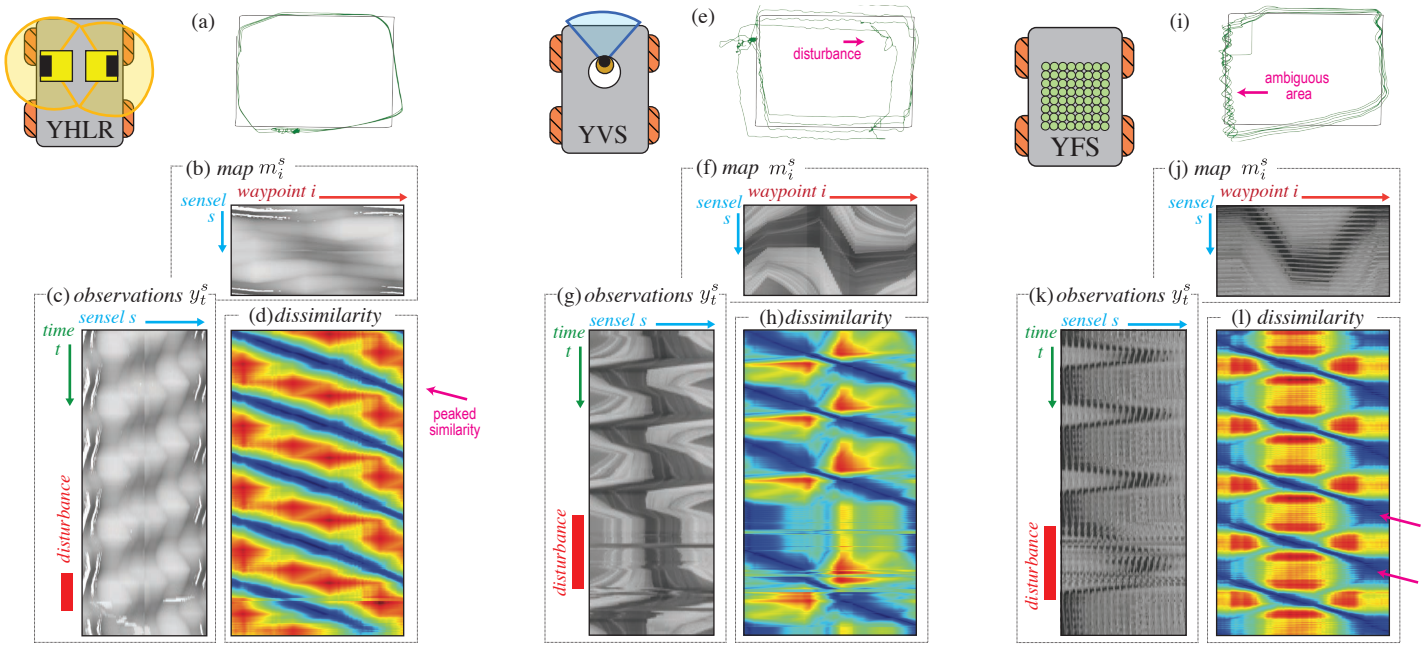


Figure 14. Navigation experiments based on repeated servoing tasks. The robot is taken along the rectangular trajectory (shown in Fig. 2); the data is recorded and used as waypoints (panels *b, f, j*). In a separate experiment with each configuration the agent follows the trajectory based on the raw observations. At some point in the experiments the observations have been perturbed (here indicated by red markers). The trajectories (panels *a, e, i*) do not follow exactly the original trajectory because 1) the goal is advanced along, so that the robot cuts corners; 2) The agent is controlling in observations space. The colormaps show the dissimilarity between the current observations and the map observations (panels *d, h, l*). Regions with relatively ambiguous observations (panel *l*) correspond to regions where the trajectory is not tracked well (panel *i*).

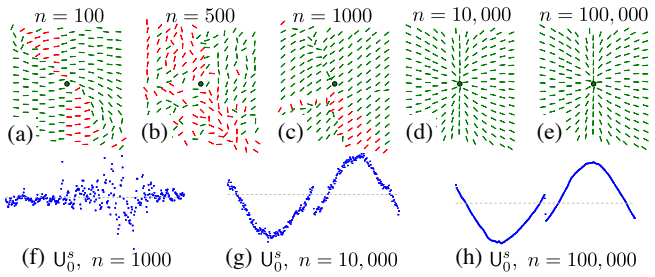
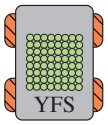


Figure 15. The task is successful before the model has converged. The top row show the velocities chosen by the gradient descent control law (same convention as in Fig. 13). The bottom row shows the learned tensor U_0^s.

is much less repeatable in pose space, especially following the perturbation of the system.



YFS: Also the field-sampler has large uncertainty towards the end of the trajectory (large blue bands in panel *k*). This unconstrained area causes the oscillations seen in the configuration space (panel *i*).

XI. Conclusions and future work

This paper considered the problem of designing bootstrapping agents that can be embodied in any robotic body. The analysis focused on the set of Vehicles, idealized models of mobile robots equipped with exteroceptive sensors, inspired by the work of Braitenberg. We derived the dynamics of three "canonical" exteroceptive robotic sensors (field-sampler, range-finders, and camera) and showed that they are more similar than expected. We described the class of BDS systems, whose bilinear dynamics can be learned easily using Hebbian-like algorithms. Moreover, their dynamics approximates the dynamics of the Vehicles well enough to perform simple spatial tasks such as servoing and navigation in observations space. We presented theoretical analysis of the learning algorithm as well as theoretical guarantees for the task. The properties of these agents were validated with simulations and experiments. It is the first time that the same learning agent, with no additional tuning or hand-designed features, is shown to work with multiple systems belonging to a reasonable large family (the Vehicles), and able to perform a task (in this case, servoing / navigation in observations space), with theoretical guarantees both on what approximated models are learned, and the performance of a task. We regard these results as evidence that formal / control-theoretical approaches is possible for learning in robotics and other large domains. We now look at some open issues.

Covering the set of all robots: The first concern is to enlarge the set of dynamical systems considered from Vehicles to a larger subset of Robots. This implies dealing with articulated bodies, which include kinematic hidden states that cannot be dismissed, and second-order dynamics, which imply controllers with internal states. There are other sensors of interest not captured by the three discussed here both proprioceptive (e.g., IMUs) and exteroceptive (e.g., touch sensors). The challenge is expanding the target subset while keeping the models general. This is a common trend in machine learning, and the solution is thought to be modularity and hierarchical representations [6].

Assumptions on training data: We made several limiting technical assumptions that we observe not needed in practice and that we aim to relax. For example, using the motor babbling exploration makes commands and observations independent, an assumption used in the proofs. Yet, further experiments, not reported here, show that the models are learned just as well using non-random and stereotypical trajectories.

Non-asymptotic results: All analysis is for the asymptotic regime (the sample average is the expectation), but better bounds are possible, because the agent is successful in the task even when the model has not converged yet (Fig. 15).

Invariance to representations: A rather important technical issue that we have not had the space to treat is invariance to *representation nuisances* acting on the data. It is easy to see that a bootstrapping agent using BDS models would be able to tolerate a linear transformation $y \mapsto Ay$ of the observations. But what would happen if the data was processed by an invertible nonlinear transformation of the type $y \mapsto f(y)$ with f invertible? This transformation does not change the observability or controllability of the system yet with a particularly evil choice of f the performance of the agent will be abysmal. Invariance to transformations can be interpreted as previous knowledge or assumptions about the system [64, Chapter 2]: the agents described here not only assumes that the robot belongs to a class of physical systems (the Vehicles) but also that the data is represented in a particular way. Different agents have different assumptions quantified by the group of transformations that they tolerated. A partial order of agents can be defined according to the groups to which they are invariant. One way to achieve invariance is by creating *canonization* operators [64, Part 3], in fact, it has been argued that canonization is the dominant aspect of the brain computation for problems such as pattern recognition [89].

References

[1] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. "Provably safe and robust learning-based model predictive control". In: *Automatica* (2013). DOI: 10.1016/j.automatica.2013.02.003.

[2] Cohen *et al.* "Functional relevance of cross-modal plasticity in blind humans". In: *Nature* 389.6647 (1997). DOI: 10.1038/38278.

[3] E. E. Thomson, R. Carra, and M. A. L. Nicolelis. "Perceiving invisible light through a somatosensory cortical prosthesis". In: *Nature Communications* 4 (2013). ISSN: 2041-1733. DOI: 10.1038/ncomms2497.

[4] 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.

[5] B. Kuipers. "Drinking from the firehose of experience". In: *Artificial Intelligence in Medicine* 44.2 (2008).

[6] Y. Bengio. "Learning Deep Architectures for AI". In: *Foundations and Trends in Machine Learning* (2009). DOI: 10.1561/2200000006.

[7] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[8] L. Ljung. *System Identification: Theory for the User*. 2nd ed. Prentice Hall, 1999. ISBN: 0136566952.

[9] T. Katayama. *Subspace methods for system identification*. Springer, 1999.

[10] F. Giri and E.-W. Bai. *Block Oriented Nonlinear System Identification*. Springer, 2010.

[11] M. O. Franz and B. Schölkopf. "A unifying view of Wiener and Volterra theory and polynomial kernel regression". In: *Neural Computation* 18.12 (12 2006). ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.12.3097.

[12] P. Ioannou. *Adaptive Control Tutorial (Advances in Design and Control)*. SIAM, 2006. ISBN: 0898716152.

[13] S. Siddiqi, B. Boots, and G. J. Gordon. "Reduced-Rank Hidden Markov Models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*. 2010.

[14] D. Hsu, S. M. Kakade, and T. Zhang. "A spectral algorithm for learning Hidden Markov Models". In: *Journal of Computer and System Sciences* 78.5 (2012). ISSN: 0022-0000. DOI: 10.1016/j.jcss.2011.12.025.

[15] M. L. Littman, R. S. Sutton, and S. Singh. "Predictive Representations of State". In: *Adv. in Neural Information Processing Systems*. MIT Press, 2001.

[16] S. Singh and M. R. James. "Predictive state representations: A new theory for modeling dynamical systems". In: *Int. Conf. on Uncertainty in Artificial Intelligence*. 2004. URL: http://portal.acm.org/citation.cfm?id=1036905.

[17] B. Boots and G. J. Gordon. "Predictive State Temporal Difference Learning". In: *Adv. in Neural Information Processing Systems*. 2011. URL: http://arxiv.org/abs/1011.0041.

[18] C. M. Bishop. *Neural Network for Pattern Recognition*. Oxford University Press, 1995.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7 (2006). DOI: 10.1162/neco.2006.18.7.1527.

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

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

[22] G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton. "Dynamical binary latent variable models for 3D human pose tracking". In: *Conf. on Computer Vision and Pattern Recognition*. 2010. DOI: 10.1109/CVPR.2010.5540157.

[23] R Memisevic and G Hinton. "Unsupervised learning of image transformations". In: *Conf. on Computer Vision and Pattern Recognition*. 2007.

[24] 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.

[25] M. Ranzato and G. Hinton. "Modeling pixel means and covariances using factorized third-order boltzmann machines". In: *Conf. on Computer Vision and Pattern Recognition*. 2010. DOI: 10.1109/CVPR.2010.5539962.

[26] H Larochelle and G Hinton. "Learning to combine foveal glimpses with a third-order Boltzmann machine". In: *Adv. in Neural Information Processing Systems*. Vol. 1. 2010.

[27] K. Yu, Y. Lin, and J. Lafferty. "Learning image representations from the pixel level via hierarchical sparse coding". In: *Conf. on Computer Vision and Pattern Recognition*. 2011.

[28] B. Hutchinson, L. Deng, and D. Yu. "Tensor Deep Stacking Networks". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35.8 (2013). DOI: 10.1109/TPAMI.2012.268.

[29] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. "Multimodal Deep Learning". In: *International Conference on Machine Learning (ICML)*. Bellevue, USA, 2011.

[30] N. Srivastava and R. Salakhutdinov. "Multimodal learning with deep Boltzmann machines". In: *Adv. in Neural Information Processing Systems*. 2012. URL: http://books.nips.cc/papers/files/nips25/NIPS2012_1105.pdf.

[31] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. ISBN: 0262193981.

[32] J Kober, J. Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey". In: *Int. J. of Robotics Research* (2013). DOI: 10.1177/0278364913495721.

[33] J. Kolter and P. Abbeel. "Hierarchical apprenticeship learning with application to quadruped locomotion". In: *Adv. in Neural Information Processing Systems*. 2008.

[34] R. C. Conant and W Ross Ashby. "Every good regulator of a system must be a model of that system ". In: *International journal of systems science* 1.2 (1970). DOI: 10.1080/00207727008920220.

[35] B. A. Francis and W. M. Wonham. "The internal model principle of control theory". In: *Automatica* 12.5 (1976). DOI: 10.1016/0005-1098(76)90006-6.

[36] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. "Developmental robotics: a survey". In: *Connection Science* 15 (2003). DOI: 10.1080/0954009031000165511O.

[37] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. "Cognitive Developmental Robotics: A Survey". In: *IEEE Trans. on Autonomous Mental Development* 1.1 (2009). DOI: 10.1109/TAMD.2009.2021702.

[38] M. Hoffmann, H. Marques, A. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer. "Body Schema in Robotics: A Review". In: *IEEE Transactions on Autonomous Mental Development* 2.4 (2010). ISSN: 1943-0604. DOI: 10.1109/TAMD.2010.2086454.

[39] M. Hersch, E. Sauser, and A. Billard. "Online learning of the body schema". In: *International Journal of Humanoid Robotics* 5.02 (2008). DOI: 10.1142/S0219843608001376.

[40] R Martinez-Cantin, M Lopes, and L Montesano. "Body schema acquisition through active learning". In: *Int. Conf. on Robotics and Automation*. 2010.

[41] J. Sturm, C. Plagemann, and W. Burgard. "Body schema learning for robotic manipulators from visual self-perception". In: *Journal of Physiology-Paris* 103.3 (2009). DOI: 10.1016/j.jphysparis.2009.08.005.

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

[43] J. Ko and D. Fox. "Learning GP-BayesFilters via Gaussian process latent variable models". English. In: *Autonomous Robots* 30.1 (2010). DOI: 10.1007/s10514-010-9213-0.

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

[45] B. Kuipers. "The Spatial Semantic Hierarchy". In: *Artificial Intelligence* 119.1–2 (2000).

[46] B. J. Kuipers and T. S. Levitt. "Navigation and mapping in large-scale space". In: *AI MAGAZINE* 9 (1988).

[47] B. Kuipers and Y.-T. Byun. "A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations". In: *Journal of Robotics and Autonomous Systems* 8 (1991).

[48] E. Remolina and B. Kuipers. "Towards a general theory of topological maps". In: *Artif. Intell.* 152.1 (2004). DOI: 10.1016/S0004-3702(03)00114-0.

[49] J. Stober, L. Fishgold, and B. Kuipers. "Sensor Map Discovery for Developing Robots". In: *AAAI Fall Symposium on Manifold Learning and Its Applications*. 2009. URL: http://www.cs.utexas.edu/~stober/pdf/stoberFSS09.pdf.

[50] J. Modayil. "Discovering sensor space: Constructing spatial embeddings that explain sensor correlations". In: *Int. Conf. on Development and Learning*. 2010. DOI: 10.1109/DEVLRN.2010.557885.

[51] 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.

[52] E. Grossmann, J. A. Gaspar, and F. Orabona. "Discrete camera calibration from pixel streams". In: *Computer Vision and Image Understanding* 114.2 (2010). ISSN: 1077-3142. DOI: 10.1016/j.cviu.2009.03.009.

[53] A. Censi and D. Scaramuzza. "Calibration by correlation using metric embedding from non-metric similarities". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013). DOI: 10.1109/TPAMI.2013.34.

[54] J. Stober, L. Fishgold, and B. Kuipers. "Learning the Sensorimotor Structure of the Foveated Retina". In: *Int. Conf. on Epigenetic Robotics*. 2009. URL: http://www.eecs.umich.edu/~kuipers/papers/Stober-epirob-09.pdf.

[55] J. Provost and B. Kuipers. "Self-organizing distinctive state abstraction using options". In: *Int. Conf. on Epigenetic Robotics*. 2007. URL: http://www.eecs.umich.edu/~kuipers/research/pubs/Provost-epirob-07.html.

[56] B. Kuipers, P. Beeson, J. Modayil, and J. Provost. "Bootstrap learning of foundational representations". In: *Connection Science* 18.2 (2006). DOI: 10.1080/09540090600768484.

[57] J. Modayil and B. Kuipers. "The initial development of object knowledge by a learning robot". In: *Robotics and Autonomous Systems* 56.11 (2008). ISSN: 09218890. DOI: 10.1016/j.robot.2008.08.004.

[58] C. Xu. "Towards the Object Semantic Hierarchy". In: *Int. Conf. on Development and Learning*. 2010. DOI: 10.1109/DEVLRN.2010.5578869.

[59] J. Stober and B. Kuipers. "From pixels to policies: A bootstrapping agent". In: *Int. Conf. on Development and Learning*. 2008. DOI: 10.1109/DEVLRN.2008.4640813.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] A. Censi. *Bootstrapping Vehicles: A Formal Approach to Unsupervised Sensorimotor Learning Based on Invariance*. Tech. rep. California Institute of Technology, 2012. URL: http://purl.org/censi/2012/phd.

[65] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1984. ISBN: 0262521121.

[66] H. Le and D. G. Kendall. "The Riemannian Structure of Euclidean Shape Spaces: A Novel Environment for Statistics". In: *Annals of Statistics* 21.3 (1993).

[67] P. W. Michor and D. Mumford. "Riemannian geometries on spaces of plane curves". In: *Journal of the European Mathematics Society* 8 (2006).

[68] T. Lochmatter and A. Martinoli. "Theoretical analysis of three bio-inspired plume tracking algorithms". In: *Int. Conf. on Robotics and Automation*. Kobe, Japan, 2009. ISBN: 978-1-4244-2788-8.

[69] J.-S. Gutmann, G. Brisson, E. Eade, P. Fong, and M. Munich. "Vector field SLAM". In: *Int. Conf. on Robotics and Automation*. 2010. DOI: 10.1109/ROBOT.2010.5509509.

[70] J. Neumann, C. Fermüller, and J. Aloimonos. "Polydioptric camera design and 3D motion estimation". In: *Conf. on Computer Vision and Pattern Recognition*. Vol. 2. IEEE. IEEE, 2003. ISBN: 0-7695-1900-8. DOI: 10.1109/CVPR.2003.1211483.

[71] M. Grossberg and S. Nayar. "The Raxel Imaging Model and Ray-Based Calibration". In: *Int. J. of Computer Vision* 61.2 (2005). DOI: 10.1023/B:VISI.0000043754.56350.10.

[72] S. Soatto. "Steps Towards a Theory of Visual Information: Active Perception, Signal-to-Symbol Conversion and the Interplay Between Sensing and Control". In: *CoRR* abs/1110.2053 (2011). URL: http://arxiv.org/abs/1110.2053.

[73] D. L. Elliott. *Bilinear control systems: matrices in action*. Springer, 2009. DOI: 10.1023/b101451.

[74] E. Margolis. "Reconstruction of periodic bandlimited signals from nonuniform samples". MA thesis. Technion, 2004. URL: http://webee.technion.ac.il/people/YoninaEldar/Download/main.pdf.

[75] W. Favoreel, B. De Moor, and P. Van Overschee. "Subspace identification of bilinear systems subject to white inputs". In: *IEEE Trans. on Automatic Control* 44.6 (1999). ISSN: 0018-9286. DOI: 10.1109/9.769370.

[76] V. Verdult and M. Verhaegen. "Kernel methods for subspace identification of multivariable {LPV} and bilinear systems". In: *Automatica* 41.9 (2005). ISSN: 0005-1098. DOI: 10.1016/j.automatica.2005.03.027.

[77] J.-W. van Wingerden and M. Verhaegen. "Subspace identification of Bilinear and LPV systems for open- and closed-loop data". In: *Automatica* 45.2 (2009). ISSN: 0005-1098. DOI: 10.1016/j.automatica.2008.08.015.

[78] L. Kneip, F. T. G. Caprari, and R. Siegwart. "Characterization of the compact Hokuyo URG-04LX 2D laser range scanner". In: *Int. Conf. on Robotics and Automation*. Kobe, Japan, 2009. DOI: 10.1109/ROBOT.2009.5152579.

[79] G. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2009. ISBN: 9780817648022.

[80] S. Benhamou. "How many animals really do the Lévy walk? Comment". In: *Ecology* 89.8 (2008). URL: http://www.jstor.org/stable/10.2307/27650759.

[81] G. Caron, E. Marchand, and E. Mouaddib. "Photometric visual servoing for omnidirectional cameras". English. In: *Autonomous Robots* 35.2-3 (2013). ISSN: 0929-5593. DOI: 10.1007/s10514-013-9342-3.

[82] R. W. Brockett. "Asymptotic Stability and Feedback Stabilization". In: *Differential Geometric Control Theory*. Ed. by R. W. Brockett, R. S. Millman, and H. J. Sussmann. Boston: Birkhauser, 1983.

[83] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Berlin: Springer-Verlag, 1999.

[84] 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.

[85] Y. Lin, E. Sontag, and Y. Wang. "A Smooth Converse Lyapunov Theorem for Robust Stability". In: *SIAM Journal on Control and Optimization* 34.1 (1996). DOI: 10.1137/S0363012993259981.

[86] S. Foss and T. Konstantopoulos. "An overview of some stochastic stability methods". In: *Journal of the Operations Research Society of Japan* 47.4 (2004).

[87] C. Pradalier and P. Bessiere. "Perceptual navigation around a sensorimotor trajectory". In: *Int. Conf. on Robotics and Automation*. Vol. 4. IEEE, 2004.

[88] D. Fontanelli, A. Danesi, F. A. W. Belo, P. Salaris, and A. Bicchi. "Visual Servoing in the Large". In: 28.6 (2009). ISSN: 0278-3649. DOI: 10.1177/0278364908097660.

[89] T. Poggio. "The computational magic of the ventral stream". In: *Nature Precedings* (2011). URL: http://hdl.handle.net/10101/npre.2011.6117.2.