Submitted, 2010 Workshop on the Algorithmic Foundations of Robotics (WAFR)
http://www.cds.caltech.edu/~murray/papers/cm10-wafr.html

Bootstrapping bilinear models of robotic sensorimotor cascades

Andrea Censi and Richard M. Murray[1]

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

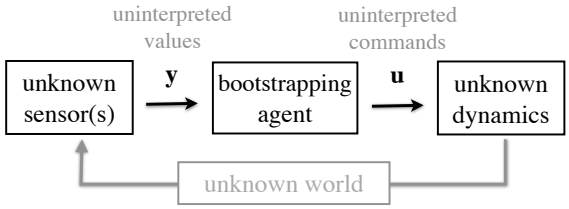


Fig. 1: The bootstrapping problem. Can an agent with no prior information about its sensors and actuators learn a model of its sensorimotor cascade, and to use it for useful tasks? Being able to solve this problem would allow us to realize truly zero-calibration plug-and-play robotics.

Control & Dynamical Systems, California Institute of Technology.
{andrea,murray}@cds.caltech.edu

1 Introduction

Two features of natural intelligence that we are far from emulating in artificial systems are its adaptiveness and generality. The human neocortex is highly uniform and its parts can be repurposed; for example, the visual cortex is repurposed to process tactile information in blind subjects [4]. Reproducing the same adaptability in artificial systems is a vast problem considered in various forms (and by various names) in several fields, such as AI, machine learning, robotics, and the specialized (and fairly distinct) fields of epigenetic and developmental robotics [11, 1].

A rather *extreme* yet *concrete* version of the problem has been put forward by Kuipers and colleagues in a long series of papers (see [9] and references therein). Suppose that an agent starts its life with no prior information about its sensors and actuators. It can read the sensors output as a sequence of values, but no semantics is associated to them. Likewise, it does not know how the unlabeled commands it can generate affect the world. The bootstrapping problem concerns creating a model for its sensorimotor cascade from scratch, and using it to achieve useful tasks.

This is a very stimulating way to see the general problem. On the one hand, it inspires dreams of future plug-and-play robotics: would it not be great, if you could just attach any sensor to your robot, and the robot would learn how to use it? On the other hand, it forces us to ask ourselves some deep questions: what are the basic assumptions we use when designing intelligent systems? What is the a priori knowledge we have to give them, and what knowledge is instead ancillary and can be learned autonomously? The bootstrapping problem also gives a concrete context for arguing over fuzzier topics, such as the nature of consciousness [8].

The most complete exemplification of Kuipers and colleagues' idea of a bootstrapping agent is the Spatial Semantic Hierarchy [7]. They show that it is possible to start from uninterpreted sensor data and build successive layers of representation up to a topological map of the environment. The crucial transition from continuous (sensor values) to symbolic representations depends on the definition of *trackers*, distinctive features in the sensory stream whose behavior is predicted by the agent's action.

Their work offers several opportunities for extension. One concern is that their results remain largely anecdotal, in the sense that they are illustrated by simulations/experiments, but not by proofs; this makes it hard to build on them. The other concern is that most of the results regard the case of range-finders: while they showed, in principle, that the same design pattern could be applied to different sensor modalities, it is unclear whether exactly the same algorithm could be run unchanged for different sensorimotor cascades. These two aspects (provability and generality) are our focus.

Throughout the paper we consider three classes of sensors: *cameras; range-finders*, devices sensing the distance to the closest obstacle; and *field samplers,* devices that sample a generic spatial field, such as the concentration

of a chemical substance. We call these three "canonical" sensors, in the sense that they are representative of many others. For example, the range-finder abstraction encompasses both sparse sonar-like sensors, as well as dense lidar-like sensors. The camera abstraction encompasses RGB and infrared cameras. The field-sampler is general enough to represent olfactory and temperature sensors. The idea is that, even if these three sensors do not capture all possible sensors, they cover enough ground such that, if we present an algorithm that can work for all of these, then it would be fair to think it is a "generic" algorithm. As for the dynamics, in the analysis we limit ourselves to fully actuated robots controlled in velocity (still, the agent does not know which command is which).

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

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

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

Our approach has been to study a model which is fairly general, yet simple enough to be learned and analyzed. In Section 2 we consider the abstract class of *bilinear dynamics sensors* and we design a two-phase bootstrapping strategy. During a first unsupervised learning phase, the agent learns a representation of its sensorimotor cascade using a simple Hebbian-learning based algorithm. In the second phase, the control action solving the servoing task is given as a function of the learned model. In Section 3 the algorithm is validated in simulation, for various configurations of the three sensors. Section 4 concerns actually proving that the algorithm works for the three canonical sensors. The analysis is long, but the method itself is simple, and can be expressed in few lines of code.

To respect the space constraints, the proofs and most of the simulations are reported in an extended version of this paper [3] to which we refer the interested reader. The extended version further develops the theory and shows an experimental application to real camera data.

2 Bootstrapping bilinear dynamics sensors

At the heart of bootstrapping, there is a problem of prediction: how do actions change the agent's view of the world? Specifically, how do the actions $\boldsymbol{u}$

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

Why using a bilinear model

In the most general case, a continuous-time dynamical system can be written as $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$; $\boldsymbol{y} = h(\boldsymbol{x})$, where $\boldsymbol{x}$ represents the hidden state. However, one seldom sees an explicit model of this kind for sensors such as cameras or range finders, because the function h should encode all information regarding the environment, and a closed form is impossible to write except in the simplest of environments. One alternative representation is focusing on the observations dynamics $\dot{\boldsymbol{y}} = g(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{x})$. In Section 4 we show that, for the three canonical sensors, this can actually be written in a closed form. In most cases, the function g depends on the underlying unobservable state $\boldsymbol{x}$. An agent that does not have access to the state $\boldsymbol{x}$ (and its dynamics) cannot learn such a model. This motivates us to look at approximating the observations dynamics by disregarding the dependence on the state, thus looking for models of the form $\dot{\boldsymbol{y}} = g(\boldsymbol{y}, \boldsymbol{u})$. Because the agent has access to $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, and $\boldsymbol{u}$, learning the map g from the data is a well-defined problem.

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

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

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

We can ignore some terms in (1) by using some assumptions regarding our specific context. For example, if we assume that $\boldsymbol{u}$ represents a "movement" or "velocity" command, in the sense that if $\boldsymbol{u}$ is 0, then the pose does not change, and $\boldsymbol{y}$ does not change as well ($\boldsymbol{u} = 0 \Rightarrow \dot{\boldsymbol{y}} = 0$), we can omit the terms $\boldsymbol{a}$, $A\boldsymbol{y}$ and $C(\boldsymbol{y}, \boldsymbol{y})$, and we are left with $\dot{\boldsymbol{y}} = B\boldsymbol{u} + D(\boldsymbol{y}, \boldsymbol{u}) + E(\boldsymbol{u}, \boldsymbol{u})$.

If we assume that $\boldsymbol{u}$ is a symmetric velocity commands, in the sense that applying $+\boldsymbol{u}$ gives the opposite effect of applying $-\boldsymbol{u}$, then we can get rid of the $E(\boldsymbol{u}, \boldsymbol{u})$ term as well. We are left with the model $\dot{\boldsymbol{y}} = B\boldsymbol{u} + D(\boldsymbol{y}, \boldsymbol{u})$,

where D is a bilinear operator. We can incorporate $B\boldsymbol{u}$ into the second term by assuming there is a trivial observation whose value is always 1. In conclusion, our *ansatz* for a generic robotic sensor is a bilinear model of the kind $\dot{\boldsymbol{y}} = D(\boldsymbol{y}, \boldsymbol{u})$.

Notation and formal definitions

We consider sensors whose output is composed of a set of sensory elements (*sensels*) that are physically related to one another. For example, the output of a camera is a function in the image space, and the output of a range-finder is a function in angle space. We call $\mathcal{Y}$ this *sensel space*. Real robots have discrete sensors with a finite number of sensels; but to make the derivation simpler we pretend that the sensel space $\mathcal{Y}$ is continuous. The values returned by the sensors lie in a certain *output space* $\mathcal{O}$. For a color camera, $\mathcal{O}$ would be the RGB space; for a range-finder, $\mathcal{O}$ would be $\mathbb{R}^+$ (distances). For simplicity, we will just assume $\mathcal{O}$ to be $\mathbb{R}$; everything can be extended to more complicated output spaces. At each time, the sensor returns the observations as a function from $\mathcal{Y}$ to $\mathbb{R}$; for technical reasons, we assume this function to be once differentiable. A formal signature for the observations $\boldsymbol{y}$ is $\boldsymbol{y} : \text{time} \to C^1(\mathcal{Y}; \mathbb{R})$.

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

In this section, we consider the restricted class of *bilinear dynamics sensors,* defined as follows.

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

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

Bootstrapping and servoing with BDS

As explained in the introduction, the task we focus is servoing/homing (Problem 1). The solutions we study are two-part strategies, composed by a learning and a control phase. In the first learning phase, the agent builds a represen-

tation of its sensorimotor cascade. Then, the agent uses the representation it has built to solve the task during the action phase.

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

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

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

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

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

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

Remark 2. Here and in the following, we do not discuss the actual algorithm to compute expectations and covariances; we prefer to be concerned with issues at a higher level. Except when noted, all the computations can be realized with simple online algorithms, often with a neural implementation. For example, (5) is directly implementable using Hebbian learning.

We can prove the following about the tensor $\mathbf{T}$. It is meant to be a substitute for $\mathbf{M}$ in (2), but it also contains other components.

Lemma 3. *Let* $\mathbf{P}$*,* $\mathbf{Q}$ *be the covariance of* $\boldsymbol{y}$ *and* $\boldsymbol{u}$*. The learned* $\mathbf{T}$ *equals*

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

Remark 4. In the expression (6), we can observe a general pattern. Every quantity that the agent learns ultimately depends on three factors:

1. *The agent's sensorimotor cascade.* In this case, $\mathbf{M}$ represents that part.
2. *The environment statistics.* In this case, the covariance $\mathbf{P}$ represents the effect of the specific environment in which learning takes place. For example, in the case of a camera, the covariance $\mathbf{P}$ depends on the statistics of the environment texture, and it would change in different environments.
3. *The experience the agent had in such environment.* In this case, $\mathbf{Q}$ captures the kind of "training" the agent had in the environment. This tell us that more common actions can bias the learning.

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

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

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

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

Remark 6. It is a classic result [2] that, if a system such as (2) is nonholonomic, there exists no smooth controller that stabilizes $\boldsymbol{y}_\star$ asymptotically. In particular (7) is smooth in $\boldsymbol{y}$, therefore it cannot work in the nonholonomic case. Instead, the requirement that the operators $\{\mathrm{M}^s_{vi} y^v\}_{i=1}^k$ commute is a technical necessity for having a compact proof and can probably be relaxed.

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

Whitening the observations. If the observations had covariance equal to the identity, we could omit the term $\mathbf{P}^{-1}$. This suggests one way to proceed: find a transformation $\boldsymbol{z} = W\boldsymbol{y}$ such that $\boldsymbol{z}$ has unit covariance. In the signal processing community, the problem of finding a suitable transformation W is well known and it is called *whitening.* Numerous algorithms exist for whitening, most of them having a neural implementation [5].

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

Proposition 7. *Assume that* $\mathbf{P}$ *and* $\mathbf{M}$ *commute, in the sense that, defining* $\mathrm{P}^q_v = \mathrm{P}^{qr} m_{rv}$, *it holds that* $\mathrm{M}^{\cdot}_{qj} \mathrm{P}^q_{\cdot} = \mathrm{P}^{\cdot}_s \mathrm{M}^s_{\cdot j}$. *Then the control strategy*

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

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

Stated in this way, this property is quite mysterious. When we get to discussing the actual sensors, we will see that in some cases the covariance $\mathbf{P}$ acts as a smoothing operation, and that the tensor $\mathbf{M}$ is similar to a gradient operation, and we will be able to interpret the condition $\mathbf{MP} = \mathbf{PM}$ in more intuitive terms as "the gradient commutes with smoothing".

3 Simulations

Before moving to a theoretical analysis for the actual sensors, we show some simulations—making sure that something works before trying to prove it is usually a good idea. Simulations are precious because one can try several sensors, but of course cannot replace proofs or experiments. The extended

version of this paper reports experiments with a real robotic system and more simulations varying the intrinsic and extrinsic sensor configuration; here only one configuration per sensor is discussed.

Care should be given as to whether the environments are "varied" enough, otherwise the learned model is biased towards a particular environment. For example, we noticed that using a simple square environment produced different results than in a perfectly circular environment; introducing randomly positioned pillars/walls rectified the situation. See [3] for some examples of the random worlds that were used. As long as there is some variability, the results are largely independent of the details of how the randomness is introduced. These observations motivated the definition of the environment's *symmetry group* (Definition 13 on page 12). For simulating a camera sensor, one should choose a random texture for the surfaces: for example, sample a luminance value independently for each 20cm section of the surface, and smooth the result. Choosing structured inputs such as sinusoids introduces unwanted correlation. This motivated the definition of *monotone environment* (Definition 17 on page 13). For the field sampler sensor, we simulated a random distribution of point sources with quadratic decay. We choose to use an omnidirectional camera, a 180deg FOV range-finder, and a field-sampler whose sensels are set on a circle; all of them placed at the robot center. The method requires no intrinsic/extrinsic calibration: see [3] for more sensor configurations.

Figure 2 onward show the learned tensors. In all figures, blue means negative and red positive; each figure is normalized independently from the others. Subfigures *e-f* show the covariance ($\mathbf{P}$) and information ($\mathbf{P}^{-1}$) tensors of the simulated sensors. The covariance encodes information on the sensel topology. For the camera and range-finder, the covariance is sparse and local: only sensels that are very close to each other are correlated. The information matrix acts very similarly to a deconvolution operator. Subfigures *b-d* show the learned tensor $\mathbf{T}$. Subfigures *g-i* show the normalized tensor $\mathbf{TP}^{-1}$ used in the control law (7). For range-finder and camera, $\mathbf{T}$ is quite different, while $\mathbf{TP}^{-1}$ is similar; we will justify this theoretically.

Figures 5-7 show the convergence properties of the control law (7) and the simplified control law (8). We are interested in evaluating the radius of convergence. We sample numerous environments and goal configurations; then we compute the control law in a volume around the goal configuration. We show the results as "success maps": we slice the (x, y, θ) volume at the three planes (x, y), (x, θ), (θ, y), and we count the percentage of times the control law pointed the robot in the right direction. Light green means $> 99\%$; see the caption for the other values.

Being simulations, these results should be interpreted in a qualitative way. What they show is that the control law (7) and (8) actually work locally. It is unclear which one works best. The rest of the paper is dedicated to proving the results suggested by the simulations.

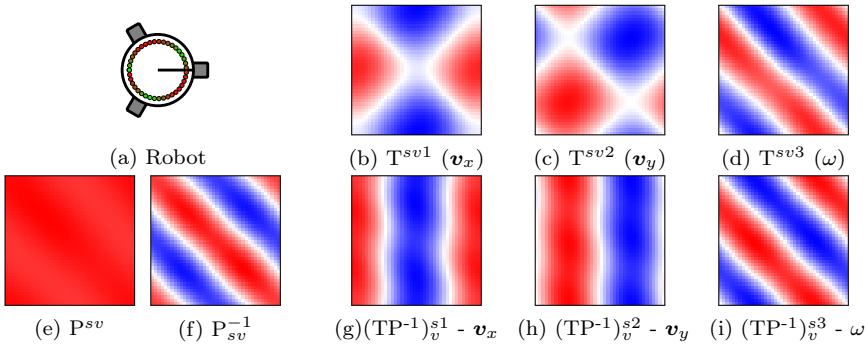


Fig. 2: Tensors learned for robot with a field sampler, with sensels placed on a 360deg ring. Each axis corresponds to the angle on the ring. Red means positive; blue negative; white zero. Note that the correlation is almost identically 1; this depends on the statistics of the field we simulated. Images best seen in color.

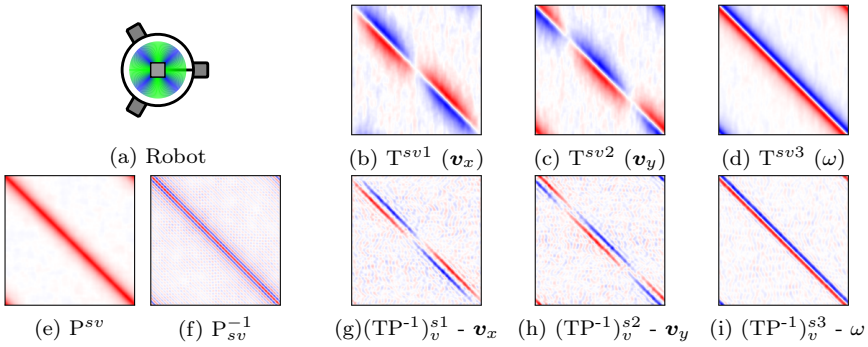


Fig. 3: Results for robot with omnidirectional camera. Note that all tensors are sparse and local (only nearby sensels interact).

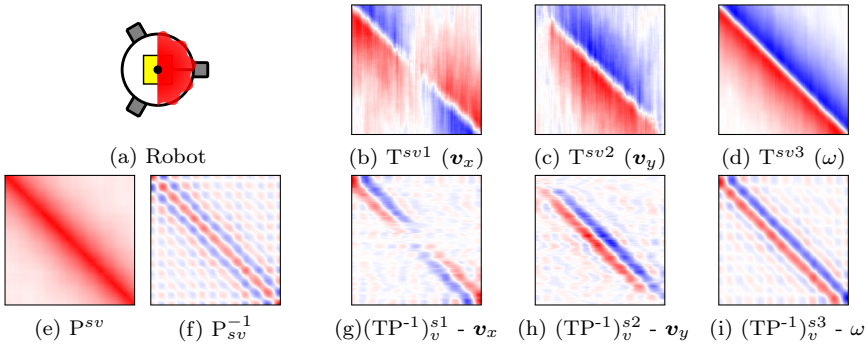
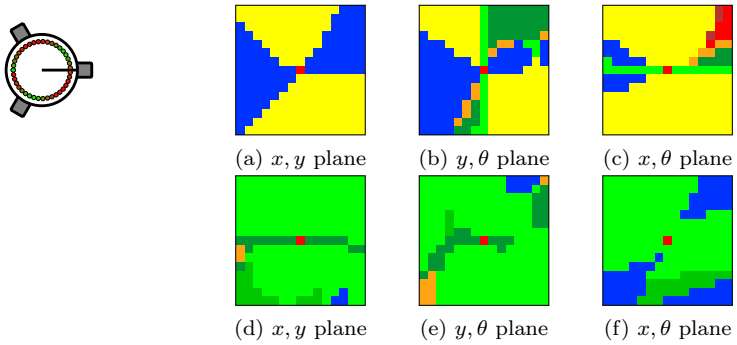


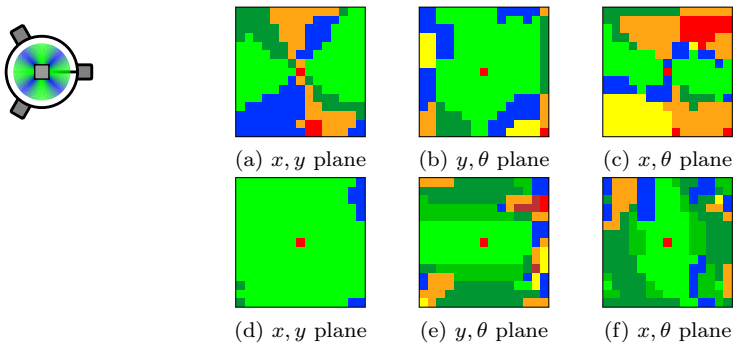
Fig. 4: Results for robot with range-finder (180deg FOV).



(a) x, y plane (b) y, θ plane (c) x, θ plane

(d) x, y plane (e) y, θ plane (f) x, θ plane

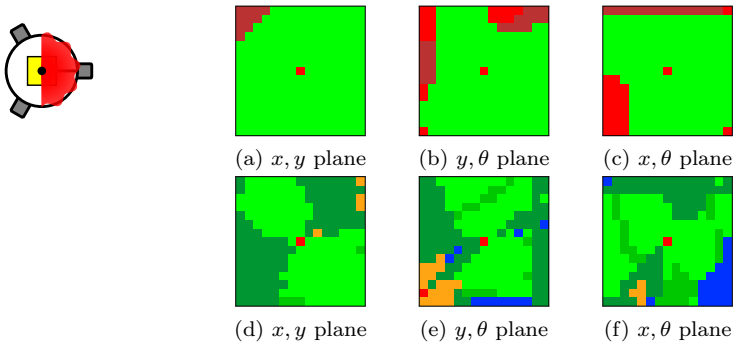
Fig. 5: Statistics of the convergence of control laws for robot with a field sampler, with sensels placed on a 360deg ring. The top figures *a-c* show the results for the control law (7); the bottom figures *d-f* show the analogous results for the simplified control law (8). The slices are 1m×1m×45deg around the goal. The figures show the percentage of times (over 200 trials with random environments) that the control law indicated a direction decreasing the error metric. Scale: 0% <25% >25% >50% >75% >95% 100%



(a) x, y plane (b) y, θ plane (c) x, θ plane

(d) x, y plane (e) y, θ plane (f) x, θ plane

Fig. 6: Results for robot with omnidirectional camera.



(a) x, y plane (b) y, θ plane (c) x, θ plane

(d) x, y plane (e) y, θ plane (f) x, θ plane

Fig. 7: Results for robot with range-finder (180deg FOV). Note that the simplified control law (8) (top figures) has larger convergence radius.

4 Analysis for the three canonical sensors

In the previous section, we showed by simulation that bootstrapping works for the three canonical sensors. In this section, we consider the problem of justifying this theoretically. The main results are summarized in Table 1. In summary, the three canonical sensors have something in common at a certain level of abstraction, as can been seen by the fact that their bootstrapped tensors are formally very similar. Because the camera and range-finder are not precisely BDS, we have to provide separate proofs of convergence. We do not have a complete proof for the convergence in all cases.

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

Table 1: Summary of the results in this section. The last two columns indicates whether we have a formal proof that the general control laws for BDS work for the particular sensors. The cases for which we do not have a proof do seem to work in simulations. The tensor $\mathbf{P}$ is the covariance of $\boldsymbol{y}$; the tensor $\mathbf{Q}$ is the covariance of $\boldsymbol{u}$; $\overline{\mu}^s$ is the average nearness (inverse of distance) in direction s.

We start with a series of definitions and results common for all sensors. We assume the reader to be familiar with basic Lie group theory [12]. Using the standard notation, given a group $\mathcal{G}$ and two elements $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{G}$, we indicate their product as $\boldsymbol{ab} \in \mathcal{G}$, and $\boldsymbol{a}^{-1}$ is the inverse of $\boldsymbol{a}$.

Definition 8. Let $\mathcal{C}$ be the *configuration space* in which the robot moves. We assume that $\mathcal{C}$ is a subgroup of SE(3).

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

We assume that the underlying dynamics is a rigid body controlled in velocity. For $\mathcal{C} = \mathrm{SE}(3)$ the commands $\boldsymbol{u}$ are the linear and angular velocity $\boldsymbol{v}$, $\boldsymbol{\omega}$. Let $\boldsymbol{t} \in \mathbb{R}^3$, $\mathbf{R} \in \mathrm{SO}(3)$ be position and attitude, and $\hat{\cdot}$ be the hat map [12]. Then the dynamics are $\dot{\boldsymbol{t}} = \mathbf{R}\boldsymbol{v}$; $\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}}$.

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

Definition 9. Let $\mathcal{Y}$ be the *sensel space.* We assume that it is a metric space, and there is a left action of $\mathcal{C}$ on $\mathcal{Y}$. In particular, for every $\boldsymbol{q} \in \mathcal{C}$ and $s \in \mathcal{Y}$, we can define the element $\boldsymbol{q}s \in \mathcal{Y}$, and $\boldsymbol{q}_1(\boldsymbol{q}_2 s) = (\boldsymbol{q}_1\boldsymbol{q}_2)s$.

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

Finally, we have to give some structure to the world around the robot. "World" is everything needed to compute the sensor output, apart from the robot pose. For a range-finder, the world includes the 3D environment structure; for a camera, it includes the texture, reflectance, and illumination information as well. We use a construction typical of stochastic geometry [10]: we assume that the set of worlds $\mathcal{W}$ can be factorized into a "shape" and "pose" component, in the sense that, for each world, there are many others that share the same shape (including color, texture, etc.), but rototranslated. Therefore, we let $\mathcal{W} \equiv \mathcal{S} \times \mathrm{SE}(3)$, where $\mathcal{S}$ is called *shape space.* We write an element of $\mathcal{W}$ as a tuple $\langle \boldsymbol{s}, \boldsymbol{p} \rangle$, with $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{p} \in \mathrm{SE}(3)$.

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

Definition 10. Given a sensel space $\mathcal{Y}$, a pose space $\mathcal{C}$, and a shape-pose space $\mathcal{W} \equiv \mathcal{S} \times \mathrm{SE}(3)$, the map $y : \mathcal{W} \times \mathcal{C} \times \mathcal{Y} \to \mathcal{O}$ corresponds to a *relative sensor* if the following two properties hold for all $x \in \mathcal{C}$.

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

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

Remark 11. Property P1 corresponds to the fact that there is an intrinsic ambiguity in choosing the frame of reference. The world and the robot have both a pose with respect to some fixed coordinate frame, but the output of the sensor depend, of course, only of the *relative* pose $\boldsymbol{q}^{-1}\boldsymbol{p}$ (let $x = \boldsymbol{q}^{-1}$ in (9) to see this fact). Property P2 corresponds to the fact that the robot is "carrying" the sensor: ultimately the output at sensel s depends only on $\boldsymbol{q}s$, therefore it is invariant if we apply x to s and multiply $\boldsymbol{q}$ by x on the right.

The bootstrapping strategy proposed in Section 2, specifically equations (3)-(5), is described by means of statistical operators such as expectations. To predict the outcome, we have to specify something about the probability distribution implied by the expectation. Firstly, we give it a name.

Definition 12. Let $p_{\mathrm{T}}(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q})$ be the *training* probability distribution of worlds/poses that the robot has experienced during its bootstrapping phase.

The analysis is based on the regularities of the probability distribution p_{T}, which we describe with the language of Lie groups.

Definition 13. Define the set *symmetry group* of p_{T} as the subgroup Sym of $\mathcal{C}$ such that, for all $x \in \mathrm{Sym}$, $p_{\mathrm{T}}(\langle \boldsymbol{s}, \boldsymbol{p} \rangle, \boldsymbol{q}) = p_{\mathrm{T}}(\langle \boldsymbol{s}, x\boldsymbol{p} \rangle, \boldsymbol{q})$.

Example. Assume we are using a planar robot ($\mathcal{C} = \mathrm{SE}(2)$), and we believe that at the end of bootstrapping, the robot experience did not privilege

one particular orientation. Then we would set Sym to be the set of planar rotations.

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

Definition 14. We call the training distribution *mixing* if, for any two couples of sensels (s_1, u_1), (s_2, u_2), for which $d(s_1, u_1) = d(s_2, u_2)$, there exists a $x \in \mathrm{Sym}$ such that $(s_2, u_2) = (xs_1, xu_1)$.

Proposition 15. *For a mixing training distribution, the expectation of any function of two sensels is only a function of their distance; for all functions ϕ, we can write $\mathbb{E}\{\phi(y^s, y^u)\}$ as $\varphi(d(s, u))$ for some other function φ.*

Corollary 16. For a relative sensor in the mixing case, the covariance of two sensels is a function of only their distance: $\mathrm{cov}(y^s, y^u) = f(d(u, s))$.

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

Definition 17. We call the environment *monotone* if the covariance of two sensels is a monotone function of the distance between the sensels.

Not all environments are monotone; see [3] for a counterexample.

4.1 Analysis of field sampler

We start with our definition of a field sampler.

Definition 18. Let_the sensels space be $\mathcal{Y} = \mathbb{R}^3$. The sensor $\boldsymbol{y}$ is a field sampler if there exists a field $\mathcal{H} : \mathbb{R}^3 \to \mathbb{R}$ such that $y^s = \mathcal{H}(\boldsymbol{t} + \mathbf{R}s)$, where $\boldsymbol{t} \in \mathbb{R}^3$ and $\mathbf{R} \in \mathrm{SO}(3)$ are the agent position and attitude.

We can show is that a field tensor is indeed a perfect BDS, and therefore all the relevant results from Section 2 apply.

Proposition 19. *A field sampler is a* BDS*, because its observations dynamics are bilinear in $\boldsymbol{y}$ and $\boldsymbol{u} = (\boldsymbol{v}, \boldsymbol{\omega})$:*

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

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

Proposition 20. *The bootstrapped tensors for a field sampler i*s

$$\mathrm{T}^{svi} = \nabla_j \mathrm{P}^{sv} \mathrm{Q}^{ij}, \tag{11}$$

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

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

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

The formula for the other components is obtained in the same way. □

Proposition 21. *For a mixing training distribution, the condition of Proposition 7 are satisfied for pure translation ($\mathcal{C} = \mathbb{R}^3$), and hence the simplified control* (8) *can be used.*

4.2 Analysis for central camera

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

Proposition 22. *Let y^s, $s \in \mathbb{S}^2$, be the luminance signal captured by the camera. Let μ^s be the nearness. The observations dynamics are*

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

See [6] for a proof. From this it follows that a camera is a BDS only for pure rotation, or with the environment at infinity, because of the dependence on the hidden state μ. When learning the observation dynamics, the hidden state μ^s gets filtered out and appears only as a multiplicative factor.

Proposition 23. *Assuming that nearness and luminance are independent in p_T, the learned tensors for a central camera are*

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

Because the camera is not a BDS, we cannot use the stock results for convergence. However, the control law (7), when instantiated for the camera, has the same form as the one we studied in our previous work on bioplausible visual control [6]. Referring to those results, we can say that (7) locally converges. As for the convergence of the simplified control (8), we can prove the analogous of Proposition 21, this time for rotation rather than translation.

Proposition 24. *In a mixing environment, ignoring the conditions at the borders of the sensels area, the condition in Proposition 7 is satisfied for rotations ($\mathcal{C} = \mathrm{SO}(3)$), and hence the simplified control law* (8) *can be used.*

Proof. (sketch) The proof is based on interpreting the covariance operator as a convolution operator on the sphere, proving that it is self-adjoint, and exploiting its commutation properties with rotation. □

4.3 Analysis for range-finder

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

Proposition 25. *The observation dynamics for a range-finder are*

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

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

We can prove the following regarding the bootstrapping strategy.

Proposition 26. *If the training distribution is mixing (Definition 14) and monotone (Definition 17) the learned tensors for a range-finder are*

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

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

Because range-finders and cameras are equivalent under pure rotations, it is immediate to show convergence of (7) in that case. In particular, the equivalent of Proposition 24 holds. It is instead challenging to prove convergence of the control law (7) for translation. The main reason is that $\mathbf{P}^{-1}$ does not cancel the term $\beta(\mathbf{P})$ in (16), due to the nonlinearity of β.

5 Conclusions and future work

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

To the best of our knowledge, this is the first presentation of a bootstrapping agent that can provably learn to use a variety of sensors to solve the same cross-modality task. The algorithm is also simple, consisting only of a few lines of code; it is fast, being extremely parallelizable. With respect to the

approach of Kuipers and his group, we focused on a more "continuous" rather than a "symbolic" solution, and this made it possible to actually prove strong results concerning the convergence of the learning process and the control law. The extended version of this work contains further development of the theory and an example application to real world robots.

So far we have been focusing more on the sensors rather than actuators, as we only considered the case of fully-actuated velocity control. Previous work [6] suggests that control in velocity can be extended to control in forces/torques with relatively little effort. Instead, generalizing to the case of nonholonomic systems appears to be quite a challenge. More towards the horizon, there are two even greater challenges. The first is learning a hidden state and its dynamics. This would be useful in the case of a camera, where the observation dynamics depends on the nearness. The other challenge is learning a nonlinear model (perhaps by learning additional levels of Taylor approximations after the bilinear terms); this would be useful in the case of the range-finder, which is not a pure BDS for its nonlinearity.

References

1. M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. Cognitive developmental robotics: A survey. *IEEE Transactions on Autonomous Mental Development*, 1(1):12 –34, may 2009.
2. Roger W. Brockett. Asymptotic stability and feedback stabilization. In R. S. Millman R. W. Brockett and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhauser, Boston, 1983.
3. Andrea Censi and Richard M. Murray. Bootstrapping multilinear models for robotic sensorimotor cascades. Technical Report CaltechCDSTR:2010.003, California Institute of Technology, Pasadena, CA, 2010. Extended version of this article. Available at the author's webpage or at http://purl.org/censi/2010/boot .
4. Leonardo G. Cohen, Pablo Celnik, Alvaro Pascual-Leone, Brian Corwell, Lala Faiz, James Dambrosia, Manabu Honda, Norihiro Sadato, Christian Gerloff, M. Dolores Catalá, and Mark Hallett. Functional relevance of cross-modal plasticity in blind humans. *Nature*, 389(6647):180–183, 1997.
5. Scott C. Douglas and Andrzej Cichocki. Neural networks for blind decorrelation of signals. *IEEE Transactions on Signal Processing*, 45(11):2829 –2842, nov 1997.
6. Shuo Han, Andrea Censi, Andrew D. Straw, and Richard M. Murray. A bio-plausible design for visual pose stabilization. Technical Report CaltechCDSTR:2010.001, California Institute of Technology, 2010. (a reduced version to appear in IROS'10).
7. B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2), 2000.
8. Benjamin Kuipers. Drinking from the firehose of experience. *Artificial Intelligence in Medicine*, 44(2):155–170, 2008.
9. Benjamin Kuipers. An intellectual history of the Spatial Semantic Hierarchy. *Robotics and cognitive approaches to spatial mapping*, 38:243–264, 2008.
10. Huiling Le and David G. Kendall. The Riemannian structure of Euclidean shape spaces: A novel environment for statistics. *Annals of Statistics*, 21(3):1225–1271, 1993.
11. Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15:151–190, 2003.
12. Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, Berlin, 1999.