Submitted, 2012 International Conference on Robotics and Automation (ICRA)
http://www.cds.caltech.edu/~murray/papers/chm12-icra.html

Fault detection and isolation from uninterpreted data in robotic sensorimotor cascades

Andrea Censi Magnus Håkansson Richard M. Murray

***Abstract*—One of the challenges in designing the next generation of robots operating in non-engineered environments is that there seems to be an infinite amount of causes that make the sensor data unreliable or actuators ineffective. In this paper, we discuss what faults are possible to detect using zero modeling effort: we start from uninterpreted streams of observations and commands, and without a prior knowledge of a model of the world. We show that in sensorimotor cascades it is possible to define static faults independently of a nominal model. We define an information-theoretic *usefulness* of a sensor reading and we show that it captures several kind of sensorimotor faults frequently encountered in practice. We particularize these ideas to the case of BDS/BGDS models, proposed in previous work as suitable candidates for describing generic sensorimotor cascades. We show several examples with camera and range-finder data, and we discuss a possible way to integrate these techniques in an existing robot software architecture.**

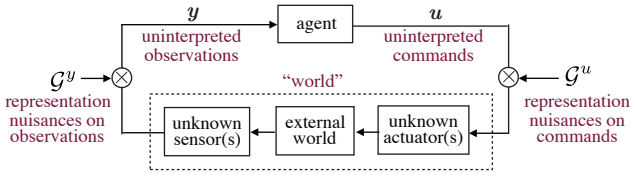


Figure 1. In this paper, we consider the problem of fault detection from a bootstrapping perspective: we start from streams of uninterpreted observations and commands, and no knowledge of a model of the "world" (the series of the robot actuators, the external world, and the robot sensors). We show that many faults occurring in robotic systems can be detected using simple classes of models (BDS/BGDS) that are versatile enough to represent different types of robotic sensors. The robustness of the approach is discussed with reference to "representation nuisances", static invertible transformation of the signals that change the representation but not the informative content of the data. This is a technical device that allows to explicitly state the limitations of the method by looking at the transformation to which it is not invariant.

I. Introduction

One of the challenges in designing the next generation of robots operating in non-engineered environments is that there seems to be an infinite amount of causes that make the sensor data unreliable or actuators ineffective, from hardware faults (the sensor physically stops working; the driver is confused), to software issues (one clock is skewed; the data is not synchronized), to configuration problems (Joe mounted a spoiler obstructing the camera on his new autonomous car), to even less predictable causes (a fly lands on the sensor). In principle, each of such problems can be solved using established techniques: the Bayesian framework gives us a way to model and identify all disturbances, if we are ready to devote design effort and computational resources to the endeavor. However, trying to anticipate all possible nuisances (e.g., including in the robot architecture a fly-detector continuously running on the sensor data) seems an untenable option.

In this paper, we consider the question of what sensor and actuator faults can be detected using zero modeling effort and relatively cheap inference. Our underlying philosophy is close to Sutton's *verification principle* [1]: the only way to build reliable systems is making them able to verify the quality of the models they are using. We start from the situation pictured in Fig. 1: we only have access to uninterpreted streams of commands $\boldsymbol{u}$ and observations $\boldsymbol{y}$. The "world" is the series of robot actuators, the external world, and the robot sensors. We do not assume to have a prior model of the world, apart from standard assumptions like causality. The theory we present is general for any robot, but the particular models we use for inference have additional assumptions on the kind of sensors and actuators (they are tailored to kinematic models and exteroceptive sensors).

A. Censi and R. Murray are with the Control & Dynamical Systems department, California Institute of Technology, Pasadena, CA. E-mail: {andrea, murray}@cds.caltech.edu. M. Håkansson is with the department of Automatic Control at Lundt University, Sweden. E-mail: et08mh1@student.lth.se

Temporary faults: In the established approach to fault detection and isolation (see, e.g., [2], [3]) one defines faults of sensors and actuators as deviation of their behavior from a nominal model, assumed to represent the "healthy" system, which can be given or identified from the data. In a robotic systems, these kind of faults capture several situations of interest: a fly lands of the sensor; a driver glitch returns all readings as 0; a robot drives over a bump, thus deviating from a planarity assumption. The established theory for fault detection mainly deals with linear systems, which cannot represent robotic sensorimotor cascades. Regarding this kind of faults, our contribution in this paper is showing that many of them can be detected using a black-box modeling of the world using BDS/BGDS models, introduced in previous work [4], [5] as a generic representation of robotic sensorimotor cascades.

Static faults in sensorimotor cascades: The other contribution of this paper is the definition of what we call *static faults* (as opposed to the others, which we call *temporary faults*). These are some examples of static faults: the readings of a range-finder are permanently occluded by other fixtures on the robot; the image given by an omnidirectional camera looking into a conic mirror includes the reflected camera and part of the robot. In these cases, the interested *sensels* (short for "sensory elements") are technically working, but the implicit assumption that the sensor is perceiving the environment is violated. Other examples of static faults we encountered include: a sensor driver setting the first few readings always to zero; a range reading that seems to oscillate for no apparent reason between a valid and invalid value; the last scan line of a wireless camera always received as white noise. These are all cases in which one must mark the interested readings as invalid before using the sensor data, and this is typically done in an *ad hoc* manner as part of a tuning process. In these cases, one does not have a nominal model of the system to use as a reference; therefore, the established approach to fault detection is not applicable.

Summary: Section II considers the problem of defining faults in a sensorimotor cascade when a nominal model is not available. From an information theoretic perspective, completely "faulty" sensels are those that do not provide information about the commands, and, symmetrically, completely "faulty" actuators are those that do not help predict the observations. Due to the ever present noise in the data, binary definitions of "faultiness" are not convenient; consequently, we define a continuous measure of sensel *usefulness* and actuator *effectiveness*. Section III recalls the definitions and properties of two classes of models called *bilinear dynamics systems* (BDS) and *bilinear gradient dynamics systems* (BGDS). Section IV shows that in the context of these models, sensels usefulness is readily computed from the correlation of observations and predictions based on the model. Section V shows the practical application of these notions for a robot equipped with camera sensors. Section VI discusses how sensel fault detection and isolation can be integrated in an existing software architecture. Section VII shows experiments with a practical implementation of these ideas, using a robot equipped with a range-finder sensor. Section VIII offers more discussion about the assumptions of the method and its invariance properties to representation nuisances. Finally, Section IX concludes the paper and offers directions for future work.

II. Static faults in sensorimotor cascades from a bootstrapping perspective

In this section, we discuss how to characterize faults in sensorimotor cascades from the black-box perspective of bootstrapping. We argue that static faults can be characterized from an information-theoretic perspective even without access to a nominal model of the system, and we define the *usefulness* of a sensel and the *effectiveness* of an actuator.

Static faults in sensorimotor cascades: We assume to start from uninterpreted stream of observations $\boldsymbol{y}_t$ and commands $\boldsymbol{u}_t$, and no knowledge of the model of the world (Fig. 1). In this context, we cannot define faults as deviation from a given model, because no model is known. However, we can find an alternative formal characterization of sensel faults, starting from an informal intuition: which sensels would we be fine in throwing away? The usual point of view is that observations are used to estimate *states*. If a sensel does not help in observing the world's state, then we can ignore it. However, in this formalization, we do not have access either to internal states of the world or the world model; what we have are just observations and commands.

We argue that we can use the commands as a proxy for the internal state, and that the usefulness of a sensel can be measure by the information it gives about the commands. This is illustrated in the diagram in Fig. 2, which shows an analogous statistical version of the classical control-theory concepts of unobservability and unreachability. The observations $\boldsymbol{y}$ are divided in two subsets $\boldsymbol{y}^a$ and $\boldsymbol{y}^b$. There is no causal link from the commands $\boldsymbol{u}$ and $\boldsymbol{y}^b$; therefore the knowledge of $\boldsymbol{y}^b$ does not give information about $\boldsymbol{u}$. This gives a characterization of faults which works even if $\boldsymbol{y}^b$ has very rich statistics compared to $\boldsymbol{y}^a$. For example, $\boldsymbol{y}^a$ could be the image of a camera connected to the robot, while $\boldsymbol{y}^b$ could be connected to a TV channel continuously looping reruns of "Friends". Even though statistical measures (such as entropy) of $\boldsymbol{y}^b$ would probably be higher than $\boldsymbol{y}^a$ (most robots do not live such interesting lives in a Manhattan loft), the knowledge of $\boldsymbol{y}^b$ does not give any information about $\boldsymbol{u}$. The same can be said for the commands $\boldsymbol{u}^b$: actuators that do not influence the observations are useless. This characterization does not cover all possible faults, only those which are evident when only the input-output properties of the sensorimotor cascade; however, this is the best we can do without having a model of the world, and it seems to fit well the case of robotic sensors: commands ultimately cause the robot to move, and sensels must give information about the robot motion.

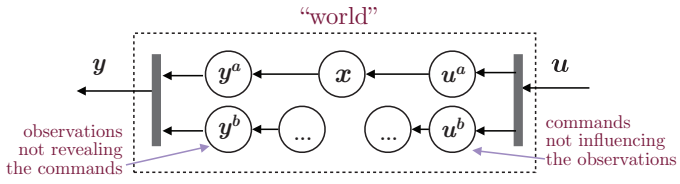


Figure 2. We give a characterization of "static faults" in sensorimotor cascades from an information-theory perspective. Completely faulty sensels ($\boldsymbol{y}^b$ in the figure) are those that do not provide any information about the commands applied to the system. Faulty actuators ($\boldsymbol{u}^b$ in the figure) are those whose commands do not help in predicting the future observations. To deal with noisy data, we use continuous versions of these conditions, called *sensel usefulness* (Definition 1) and *actuator effectiveness* (Definition 2).

Formal definition: We say that the i-th sensel of a sensorimotor cascade $(\boldsymbol{u}, \boldsymbol{y})$ is *useless* if $\boldsymbol{u}$ is conditionally independent on[1] y^i given the other sensels: $p(\boldsymbol{u}_{:t}|\boldsymbol{y}_{:t}) = p(\boldsymbol{u}_{:t}|\boldsymbol{y}^{-i}_{:t})$. However, this definition is not convenient, as "independence" is a binary property, and does not capture the reality that sensel quality degrades gracefully. One possibility is defining a continuous measure of "usefulness" as a distance between the distributions $p_1 = p(\boldsymbol{u}_{:t}|\boldsymbol{y}_{:t})$ and $p_2 = p(\boldsymbol{u}_{:t}|\boldsymbol{y}^{-i}_{:t})$. Unfortunately, to the best of our understanding, the only true well-defined distance between distributions is that induced by the Fisher information *metric,* which is difficult to compute, and it is really only definable for finite-dimensional family of distributions [6]. Therefore, we define usefulness of a sensel without committing to a specific distance, but for a generic divergence function.

Definition 1. The *$\mathcal{D}$-usefulness* of the i-th sensel for a given divergence $\mathcal{D}$, is defined as $\text{usefulness}^i_{\mathcal{D}} = \mathcal{D}(p_1, p_2)$, where $p_1 = p(\boldsymbol{u}_{:t}|\boldsymbol{y}_{:t})$ and $p_2 = p(\boldsymbol{u}_{:t}|\boldsymbol{y}^{-i}_{:t})$.

Actuator faults: The same ideas apply to actuators: if the knowing the commands given to an actuator does not help in predicting the observations, that actuator is not "effective".

Definition 2. The *$\mathcal{D}$-effectiveness* of the a-th actuator for a given divergence $\mathcal{D}$, is defined as $\mathcal{D}(p_1, p_2)$, where $p_1 = p(\boldsymbol{y}_{:t}|\boldsymbol{u}_{:t})$ and $p_2 = p(\boldsymbol{y}_{:t}|\boldsymbol{u}^{-a}_{:t})$.

This said, in this paper we are mainly concerned with sensels faults rather than actuator faults, so we will not have the occasion of using this definition.

III. BDS and BGDS models

In this section we recall two classes of models for representing sensorimotor cascades, introduced in [4], [5].

[1] We use subscripts to indicate time, and the notation "$:t$" means "up to and including time t". The superscript in y^i indicates the i-th element of the vector $\boldsymbol{y}$ (later on, if y is a continuous field on a manifold $\mathcal{S}$, we indicate it with y^s, $s \in \mathcal{S}$); and $\boldsymbol{y}^{-i}$ indicates the vector $\boldsymbol{y}$ with the i-th sensel removed.

Bilinear dynamics systems (BDS): In the BDS model, we assume that the observations and commands are vectors of real numbers: $\boldsymbol{y} \in \mathbb{R}^{n_\mathbf{y}}$, and $\boldsymbol{u} \in \mathbb{R}^{n_\mathbf{u}}$. The model is parametrized by a tensor M^i_{ja}, with three indices i, j, a, with $n_\mathbf{y} \times n_\mathbf{y} \times n_\mathbf{u}$ elements. The dynamics are described by the relation

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

where ϵ^i_t is assumed to be zero-mean gaussian noise. This model assumes implicitly that the commands $\boldsymbol{u}$ can be interpreted as kinematic velocities imposed on the system (if $\boldsymbol{u} = 0$ then $\dot{\boldsymbol{y}} = 0$). The "$\sum$" symbols are redundant because throughout the paper we respect the Einstein convention that repeated up and down indices are summed over.

Learning is done by computing three tensors N, P, Q:

$$N^{bik} = \mathbb{E}\{u^b \dot{y}^i y^k\},\ P^{jk} = \mathbb{E}\{y^j y^k\},\ Q^{ab} = \mathbb{E}\{u^a u^b\}. \tag{2}$$

Table I summarizes the meaning of the tensors and their dimensions. We take the symbol "$\mathbb{E}$" to mean the expectation over the learning data. One can show that $\mathbb{E}\{N^{bik}\} = \sum_j \sum_k Q^{ab} M^i_{ja} P^{jk}$, therefore an estimate of M can be computed as $\hat{M}^i_{ja} = \sum_j \sum_k N^{bik} (P^{-1}_{jk})(Q^{-1}_{ab})$. For estimating $\boldsymbol{u}$, note that, once $\boldsymbol{y}$ is fixed, (1) is linear in $\boldsymbol{u}$, therefore it can be recovered using linear least squares.

Bilinear gradient dynamics systems (BGDS): The BGDS models are a particularization of BDS models, in which it is assumed that sensels are characterized by their position in a "sensel space", and their dynamics only depend on nearby sensels. Formally, we assume that the observations $\boldsymbol{y}$ are a scalar function defined on a certain manifold $\mathcal{S}$. We write y^s_t, to indicate the value of the observations at a generic spatial location $s \in \mathcal{S}$. For example, in the case of a camera, s ranges over the pixels. The gradient $\nabla \boldsymbol{y}$ has $\dim(\mathcal{S})$ components. We indicate by $\nabla_d y^s_t$ the gradient of $\boldsymbol{y}$ in the d-th direction at location s at time t. The model is parametrized by two tensor fields B and G on $\mathcal{S}$: at each point $s \in \mathcal{S}$, $B^s_a \in \mathbb{R}^{n_\mathbf{u}}$ and $G^{ds}_a \in \mathbb{R}^{\dim(\mathcal{S}) \times n_\mathbf{u}}$. The dynamics are described by the relation

$$\dot{y}^s_t = \textstyle\sum_a \sum_d (G^{ds}_a \nabla_d y^s_t + B^s_a)\ u^a_t + \epsilon^s_t, \qquad \text{(BGDS)} \tag{3}$$

Table I
NOTATION USED FOR BDS AND BGDS MODELS.

(A) INDICES

indices	*domain*	*ranges over...*
a, b	$0, \ldots, n_\mathbf{u} - 1$	commands
i, j, k	$0, \ldots, n_\mathbf{y} - 1$	discrete observations
s	$\mathcal{S}$	continuous observations
d, e, f	$0, \ldots, \dim(\mathcal{S}) - 1$	gradient directions

(B) TENSORS USED FOR BDS MODELS

tensor	*shape*	*dimensions*	
M^i_{ja}	$(1, 2)$	$n_\mathbf{y} \times n_\mathbf{y} \times n_\mathbf{u}$	BDS dynamics.
N^{bik}	$(3, 0)$	$n_\mathbf{y} \times n_\mathbf{y} \times n_\mathbf{u}$	Proxy for M used during learning.
P^{jk}	$(2, 0)$	$n_\mathbf{y} \times n_\mathbf{y}$	Second moment of observations.
Q^{ab}	$(2, 0)$	$n_\mathbf{u} \times n_\mathbf{u}$	Second moment of commands.

(C) TENSORS USED FOR BGDS MODELS

tensor	*shape*	*dimensions*	
G^{sd}_a	$(2, 1)$	$\lvert\mathcal{S}\rvert \times \dim(\mathcal{S}) \times n_\mathbf{u}$	Gradient part of dynamics.
B^s_a	$(1, 1)$	$\lvert\mathcal{S}\rvert \times n_\mathbf{u}$	Affine part of dynamics.
H^{sb}_e	$(2, 1)$	$\lvert\mathcal{S}\rvert \times \dim(\mathcal{S}) \times n_\mathbf{u}$	Proxy for G during learning.
C^{sb}	$(2, 0)$	$\lvert\mathcal{S}\rvert \times n_\mathbf{u}$	Proxy for B during learning.
R^s_{de}	$(1, 2)$	$\lvert\mathcal{S}\rvert \times \dim(\mathcal{S}) \times \dim(\mathcal{S})$	Statistics of gradients.

with ϵ^s_t zero-mean gaussian noise. Learning is done by computing the tensors H, C, R, Q:

$$H^{bs}_e = \mathbb{E}\{u^b \dot{y}^s \nabla_e y^s\}, \tag{4}$$

$$C^{bs} = \mathbb{E}\{u^b \dot{y}^s\}, \tag{5}$$

$$R^s_{de} = \mathbb{E}\{\nabla_d y^s \nabla_e y^s\}, \tag{6}$$

$$Q^{ab} = \mathbb{E}\{u^a u^b\}. \tag{7}$$

Given these quantities, the tensors G and B can be recovered as $\hat{G}^{ds}_a = \sum_e \sum_b (R^{-1})^{sde} H^{bs}_e (Q^{-1}_{ab})$ and $\hat{B}^s_a = \sum_b C^{bs} (Q^{-1}_{ab})$. Here $(R^{-1})^{sde}$ denotes the point-wise inverse for the d, e indices for a fixed s. As in the previous case, for estimating $\boldsymbol{u}$, note that, once $\boldsymbol{y}$ is fixed, (1) is linear in $\boldsymbol{u}$, therefore it can be recovered using linear least squares.

Note that the BGDS model is more complicated and less general than a BDS model; however it is more efficient, because the computational cost for learning and prediction is linear in the number of sensels (size of the domain $\mathcal{S}$), while the cost for a BDS model is quadratic in the number of sensels.

IV. SENSEL USEFULNESS IN BDS/BGDS MODELS

This section shows that the usefulness of a sensel in BDS and BGDS models can be computed as the correlation between the observed sensel value derivative and the prediction given by the model. This result depends heavily on the fact that there is an instantaneous linear relation between $\dot{\boldsymbol{y}}$ and $\boldsymbol{u}$.

Proposition 3. *Assume that: a) the sensorimotor cascade is a BDS or BGDS (this model assumption is denoted as "$\mathcal{M}$"); b) the divergence d between two distributions used is the difference of the trace of the relative information matrices; c) the second moment of $\boldsymbol{u}$ is isotropic ($\mathbb{E}\{\boldsymbol{u}\boldsymbol{u}^*\} = U\mathbf{I}_{n_\mathbf{u}}$); then the usefulness of a sensel is a function $\gamma(\rho) = \frac{1}{U}\frac{\rho^2}{1-\rho^2}$ of only the correlation between the observed derivative $\dot{y}^i_t$ and the derivative $\hat{\dot{y}}^i$ predicted by the BDS/BGDS model:*

$$\textit{usefulness}^i_{d|\mathcal{M}} = \gamma(\mathsf{corr}(\hat{\dot{y}}^i, \dot{y}^i)). \tag{8}$$

The "$\mathcal{M}$" subscript in (8) is a reminder that this is only valid under the assumption of the system being BDS/BGDS.

Proof: We first note that BDS/BGDS models give an instantaneous relation among the derivative $\dot{\boldsymbol{y}}$ and the commands $\boldsymbol{u}_t$; because the commands at time t only influence the derivative at time t, we can write $p(\boldsymbol{u}_t | \boldsymbol{y}_{:t}, \mathcal{M})$ as $p(\boldsymbol{u}_t | \langle \boldsymbol{y}_t, \dot{\boldsymbol{y}}_t \rangle, \mathcal{M})$. Therefore, we can write the distance between $p_1 = p(\boldsymbol{u}_{:t} | \boldsymbol{y}_{:t}, \mathcal{M})$ and $p_2 = p(\boldsymbol{u}_{:t} | \boldsymbol{y}^{-i}_{:t}, \mathcal{M})$ as the time average between the distribution of $\boldsymbol{u}$ conditioned to the measurements of $\boldsymbol{y}$ and $\dot{\boldsymbol{y}}$ at time t:

$$d(p_1, p_2) = \mathbb{E}_t\{d(p(\boldsymbol{u}_t | \langle \boldsymbol{y}_t, \dot{\boldsymbol{y}}_t \rangle, \mathcal{M}), (p(\boldsymbol{u}_t | \langle \boldsymbol{y}^{-i}_t, \dot{\boldsymbol{y}}^{-i}_t \rangle, \mathcal{M})).$$

Next, we use the fact that the dynamics is linear in $\boldsymbol{u}$. We can write both (1) and (3) in the form:

$$\dot{y}^i_t = \left[\boldsymbol{f}^i_t\right]^* \boldsymbol{u}_t + \epsilon^i_t, \tag{9}$$

where $\boldsymbol{f}^i_t$ is a vector that depends on the current observations $\boldsymbol{y}_t$. (For the BGDS case, the index i becomes $s \in \mathcal{S}$ and the sums in the following become integrals over $\mathcal{S}$).

Because (9) is linear in $\boldsymbol{u}_t$, an estimate of $\boldsymbol{u}_t$ can be obtained using linear least squares. The information matrix of the least-squares estimate of $\boldsymbol{u}_t$ is given by $\mathcal{I} = \mathrm{cov}(\boldsymbol{u} | \langle \boldsymbol{y}, \dot{\boldsymbol{y}} \rangle)^{-1} = \sum_i ([\boldsymbol{f}^i_t \boldsymbol{f}^{i^*}_t)/\mathsf{var}(\epsilon^i_t)$. Using the information matrix trace as

the distance between distributions, the increment in information given by the i-th sensel is $\text{Trace}(\boldsymbol{f}_t^i \boldsymbol{f}_t^{i*})/\text{var}(\epsilon_t^i) = \left\|\boldsymbol{f}_t^i\right\|^2/\text{var}(\epsilon_t^i)$: letting $\tilde{p}_1 = p(\boldsymbol{u}_t|\left\langle \boldsymbol{y}_t, \dot{\boldsymbol{y}}_t \right\rangle, \mathcal{M})$ and $\tilde{p}_2 = (p(\boldsymbol{u}_t|\left\langle \boldsymbol{y}_t^{-i}, \dot{\boldsymbol{y}}_t^{-i} \right\rangle, \mathcal{M})$, we have $d(\tilde{p}_1, \tilde{p}_2) = \frac{\left\|\boldsymbol{f}_t^i\right\|^2}{\text{var}(\epsilon_t^i)}$. This is valid for a particular time t and particular observations $\boldsymbol{y}_t$. Averaging over time, we obtain

$$\text{usefulness}_{d|\mathcal{M}}^i = \frac{\mathbb{E}\{\left\|\boldsymbol{f}_t^i\right\|^2\}}{\text{var}(\epsilon_t^i)}. \tag{10}$$

Not surprisingly, we find out that usefulness is expressed as a kind of signal-to-noise ratio. Let $\hat{\dot{y}}^i = \boldsymbol{f}_t^{i*} \boldsymbol{u}_t$ be the predicted change in y given the learned model and the current observations, and let $\dot{y}_t^i$ be the actual observations. It holds that $\dot{y}_t^i = \hat{\dot{y}}^i + \epsilon_t^i$, where ϵ_t^i is assumed to be independent of y. Let $\rho^i = \text{corr}(\dot{y}^i, \hat{\dot{y}}^i)$ be the correlation between observations and model prediction. For any three zero-mean random variables a, b, c such that $a = b + c$ and $\mathbb{E}\{ac\} = 0$, it holds that $\text{corr}(a, b) = (1 + \text{var}(c)/\text{var}(b))^{-1/2}$. For $\dot{y}^i, \hat{\dot{y}}^i, \epsilon^i$ we obtain $\rho^i = (1 + \text{var}(\epsilon^i)/\text{var}(\hat{\dot{y}}^i))^{-1/2}$. As a sanity check, note that as $\text{var}(\epsilon^i) \to 0$, $\rho^i \to 1$, and as $\text{var}(\epsilon^i) \to \infty$, $\rho^i \to 0$. An expression for $\text{var}(\hat{\dot{y}}^i)$ can be found easily assuming that the second moment of $\boldsymbol{u}$ is isotropic ($\mathbb{E}\{\boldsymbol{u}\boldsymbol{u}^*\} = U\mathbf{I}_{n_{\boldsymbol{u}}}$), as we have $\text{var}(\hat{\dot{y}}^i) = \text{var}(\boldsymbol{f}_t^{i*} \boldsymbol{u}_t) = U\mathbb{E}\{\left\|\boldsymbol{f}_t^i\right\|^2\}$. Substituting this in the previous expression for ρ^i, one gets $\rho^i = (1 + \text{var}(\epsilon^i)/U\mathbb{E}\{\left\|\boldsymbol{f}_t^i\right\|^2\})^{-1/2}$. From this, we obtain $U\frac{\mathbb{E}\{\left\|\boldsymbol{f}_t^i\right\|^2\}}{\text{var}(\epsilon^i)} = \frac{(\rho^i)^2}{1-(\rho^i)^2}$, which, together with (10) gives (8). Note that the right expression blows up for $\rho \to \pm 1$, but this corresponds to the case $\text{var}(\epsilon^i) \to 0$, in which also the left-hand side blows up. ■

V. Application to camera data with BGDS models

In this section we show an application of the theory to the case of a robot with camera sensors, using a BGDS model. Several kinds of static and temporary faults are identified with zero prior knowledge of the robot sensors and actuators.

Platform: The platform we use is an Evolution Robotics ER-1 (Fig. 3a). The robot has differential drive dynamics, and it is commanded in angular and linear velocity. There are two cheap USB web-cams mounted on board that give 320x240 RGB images at ~7.5 Hz; one is pointing forward, the other is pointing down towards the ground. Because of a mirror mounted on the robot, the camera frame includes part of the robot chassis, the ground, and the environment in front of the robot, reflected through the mirror (Fig. 3b).

Data: The robot is driven through a series of indoor office-like, lab-like, and an outdoor campus environment, at day and night (Fig. 3c). The terrain went from smooth indoor, to slippery, to bumpy. There were occasionally moving people around the robot. The linear and angular velocities were chosen among fixed values: for the angular velocity, $\omega \in \{-0.2, 0, +0.2\}$ rad/s, and for the linear velocity $v \in \{-0.3, 0, +0.3\}$ m/s. Having a discrete number of velocity commands values is not a requirement of the method, but it makes the subsequent analysis simpler. The total length of the logs is about 50 min (about 24,000 camera frames).

Learning: The observations $\boldsymbol{y}$ are set to be the luminance values of the composite frame obtained by splicing together the two camera images (Fig. 3b). In this context, the domain $\mathcal{S}$ is the 480×320 frame rectangle; when writing y^s, s ranges over pixels. We do not assume to know the intrinsic calibration of the camera, which would be the map from $\mathcal{S}$ to the visual sphere $\mathbb{S}^2$ (i.e., the direction associated to each pixel). For the commands $\boldsymbol{u}$, we set $u^0 = \omega$, $u^1 = v$. Observations and commands are available only at discrete sampling times t_k, in general not evenly spaced. The signals are synchronized to obtain pairs $(\boldsymbol{u}_{t_k}, \boldsymbol{y}_{t_k})$ by using the frontal camera as the master signal, and choosing the closest sample in time for the other two signals.The learning algorithm (formulas (4)-(7)) needs to know $\dot{\boldsymbol{y}}$ and $\nabla \boldsymbol{y}$. To compute $\dot{\boldsymbol{y}}$, we use a finite-time difference, setting $\dot{\boldsymbol{y}}_{t_k} = (\boldsymbol{y}_{t_{k+1}} - \boldsymbol{y}_{t_{k-1}})/(t_{k+1} - t_{k-1})$. Before computing the gradients $\nabla \boldsymbol{y}$, we smooth the data using an isotropic Gaussian filter with $\sigma = 0.5$ pixels. The additional materials[2] include videos of the logs, as well as ROS [7] Bag files containing the post-processed synchronized streams of $\boldsymbol{y}$ and $\boldsymbol{u}$. The tensors H and R learned are shown in Fig. 4ab. From those, one recovers the tensor G (Fig. 4c). The tensors C, B are not shown. The interpretation of the BGDS tensors is not intuitive; in the case of a camera, they can be put in correspondence with average patterns of optic flow (but this is not quite right, as we never define optic flow). What is intuitive to see is that corresponding parts of the environment in the top and bottom frame are given the same intensity (as in G_1^{s0}). The sign is inverted if the image is mirrored along the direction of the gradient (as in G_0^{s1}).

Sensel usefulness: Fig. 4*d* shows the estimated sensel usefulness, which captures a variety of phenomena. The fixed parts of the image are correctly identified (chassis and wheel). Also the border between the two camera frames is identified as an anomaly. Note that these are the pixels that one would want to mark as invalid before processing the images. It is interesting to note that the distant parts of the environment (both in the front camera and in the mirrored image) are given a lower usefulness score than the rest of the image. This is probably due to the finite resolution of the sensor: for the outdoor environments, the robot sees almost no change due to translational motion, while the BGDS model predicts motion, as it does not model the distance to the objects (a hidden state). This is a case in which the model slightly fails in representing the dynamics of the sensor; still, the usefulness score for those areas is much larger than the score given to the actual faults.

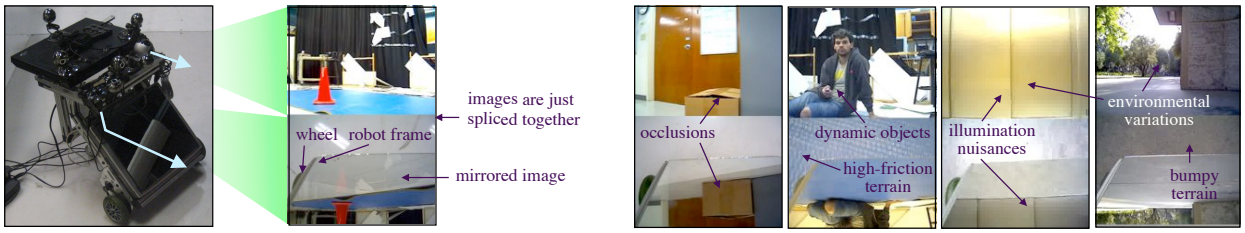
Temporary actuator faults: Fig. 5 shows the temporary actuator fault detection. We first learn the BGDS model from data, and then we compare the estimated command $\hat{\boldsymbol{u}}_t$, obtained via least squares from (3), with the recorded command $\boldsymbol{u}_t$. Subfigures *a,b,c* correspond to an indoor log, and *d,e,f* to an outdoor log. We had chosen these two logs to display because there is slipping in the indoor log, and bumpy terrain in the outdoor log; however, we discovered that most of the detected faults come from other reasons.

We note that the estimated commands are, in general, of the wrong scale. This is due to the fact that the variation of $\dot{\boldsymbol{y}}$ for a camera depends on the distance to the object. By fitting an averaged model, the BGDS models estimates a reduced command outdoor and higher command indoor. However, the sign is generally correct. Therefore, we use the error function

$$e(u^a, \hat{u}^a) = \begin{cases} \max\{0, -u^a \hat{u}^a\} & \text{if } |u^a| \neq 0, \\ |\hat{u}^a| & \text{if } |u^a| = 0, \end{cases} \tag{11}$$

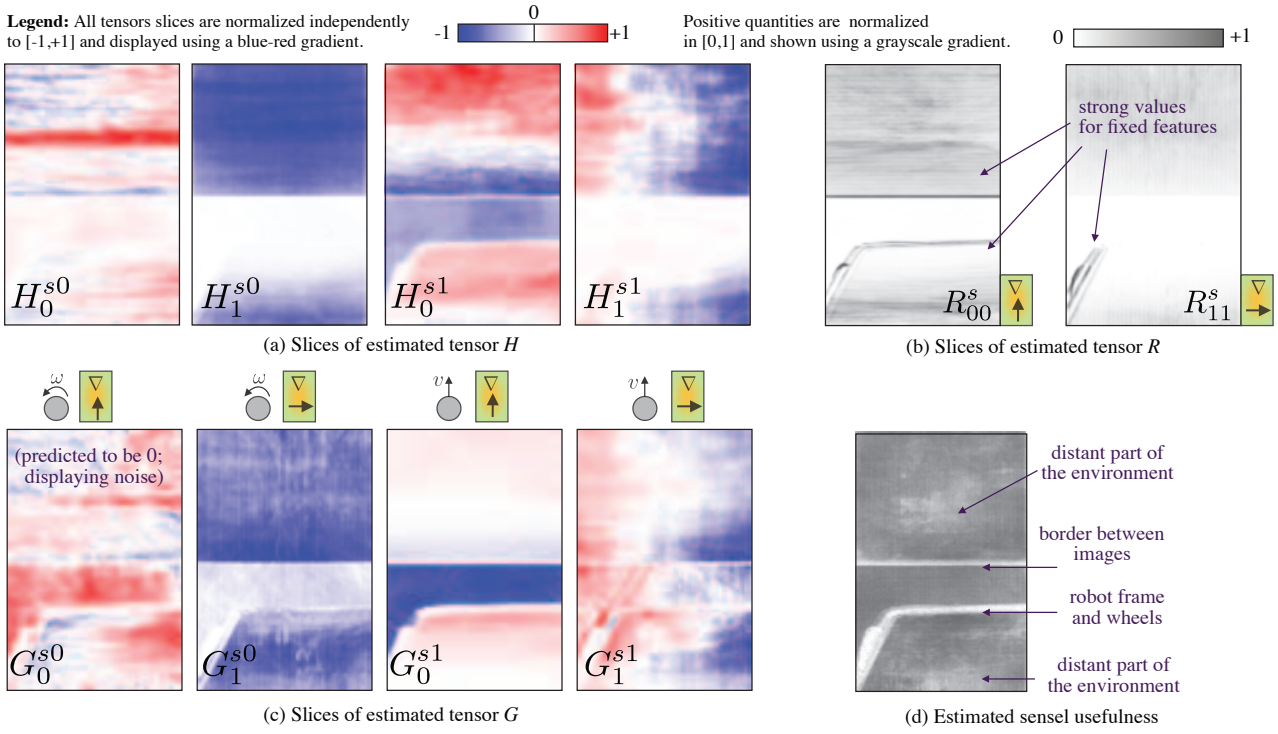
which detects a fault only if the signs of u^a and $\hat{u}^a$ disagree.

[2] Available at http://purl.org/censi/2011/fault



(a) Robotic platform (b) Composite frame (c) A sample of traversed environments

Figure 3. (a) For the first set of experiments, we use an Evolution Robotics ER1. There are two webcams mounted on the platform. One is pointing forward, and the other is pointing towards the ground. There is a mirror mounted on the robot, so that the bottom camera sees the ground, the front environment through the mirror, and part of the robot frame, including a wheel. (b) The input to the method (the array $\boldsymbol{y} = \{y^s\}_{s \in \mathcal{S}}$) is the grayscale corresponding to the composite frame. (c) The robot is driven through a variety of indoor and outdoor environments, containing dynamic elements and challenging terrains.



(a) Slices of estimated tensor H (b) Slices of estimated tensor R

(c) Slices of estimated tensor G (d) Estimated sensel usefulness

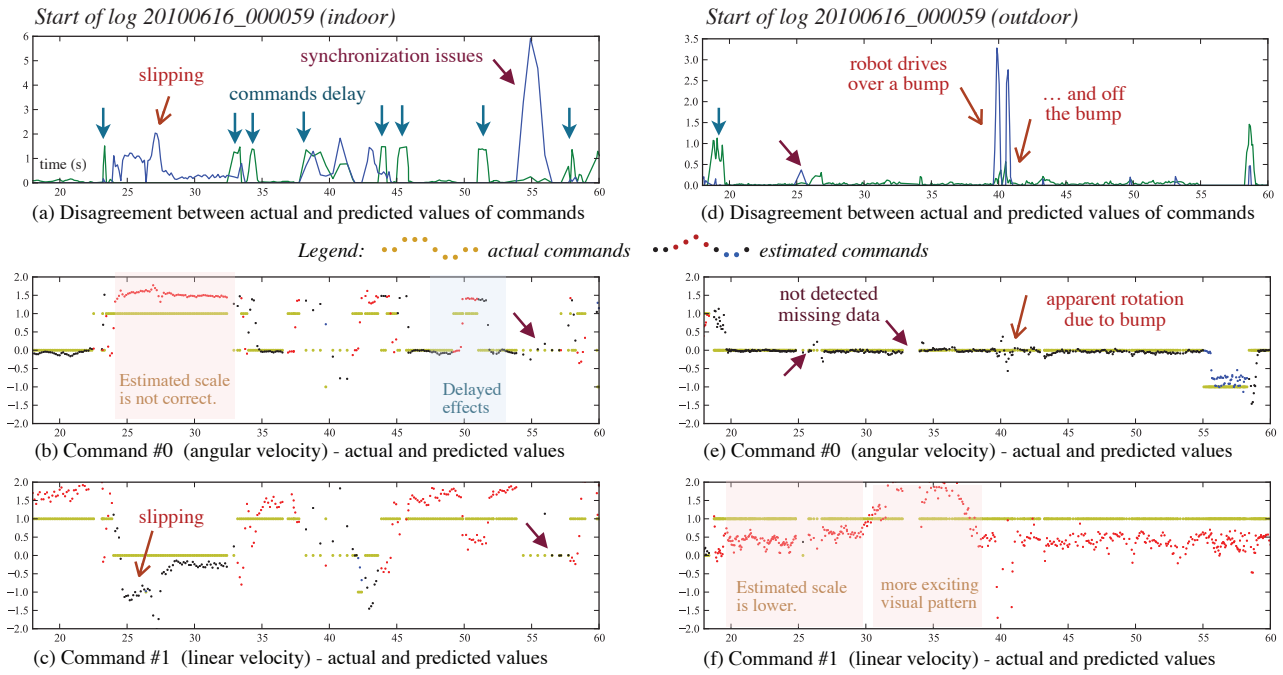
Figure 4. (a-b) A BGDS model (equation 3) is parametrized by two tensors G and B. We estimate the parameters by first learning two tensors H and R using Hebbian-like learning, by computing the expectation of certain quantities, given by (4)–(7). The tensor H_e^{sb} can be interpreted as the expectation of the product of the b-th command and the e-th component of the gradient of $\boldsymbol{y}$ at sensel s. Intuitively, these depend on the distributions of the commands and of the sensory statistics and must be normalized. For this normalization, one needs the second moment of $\boldsymbol{u}$ (equation (7)) and the tensor R, which gives the energy of the observed gradients (equation (6)), here shown in (b). Note that there is a strong energy associated to the edges of the fixed features. (c) This figure shows the slices of the resulting tensor G. Each slice is associated to a command and a gradient direction.

The diagrams in Fig. 5a-d show that slipping and bumpy terrain are correctly detected; however, most of the detections correspond to two unexpected effects. We discovered that, due to the architecture we used for the logging (various Python scripts on two laptops, communicating via Spread [8]), there is a large delay between the application of a command, and the effect of the command, due to the delay in passing the commands between computers and down to the various layers of drivers. Therefore, there is an inconsistency detected every time that commands are switched; these events are detected by our method as temporary actuator faults. Another quirkiness of the logs which is detected as a fault is the occasional reduced frequency of signals (caused by the non-realtime nature of the software). We do not show here the equivalent temporary *sensels* fault detection, which was already demonstrated in [4]; as expected, the strongest signal one gets is for dynamic objects that move independently of the robot.

VI. Integrating Faulty Sensels Detection in a Robot Software Architecture

We have described a fault detection method which is largely independent of the robotic sensor and tasks. Therefore, it is interesting to consider whether it can be just "plugged in" in an existing architecture, with minimal modification of the existing components. The architecture we are experimenting with is shown in Fig. 6. There are three components:

- The traditional controller is largely unchanged. For participating in the architecture, one only needs to add a "panic" signal, and having the data format support the semantics of an "invalid" sensels.
- A bootstrapping agent is included in a passive role: it reads observations and commands, and produces the



(a) Disagreement between actual and predicted values of commands

(d) Disagreement between actual and predicted values of commands

(b) Command #0 (angular velocity) - actual and predicted values

(e) Command #0 (angular velocity) - actual and predicted values

(c) Command #1 (linear velocity) - actual and predicted values

(f) Command #1 (linear velocity) - actual and predicted values

Figure 5. This figure illustrates the detection of temporary actuator faults. Figures *a,b,c* refer to an indoor run, and *d,e,f* refer to an outdoor run. Figures *b,c,e,f* show estimated and recorded commands. Due to the limitations of BGDS models, the scale of the estimated commands are incorrect. Therefore, for fault detection we use a measure of disagreement that ignores the scale of the signals, given by (11). The disagreement between predicted and recorded commands is shown in figures *a* and *d*. This disagreement signal detects a wide variety of actuator faults, both physical faults (wheel slipping, violations of planarity) as well as software faults (the delay between imposing a command and seeing the effects; and data synchronization/frequency issues).

usefulness measure. The model used internally by the agent is not important to the other components.

- A "glue" component is added, which marks as invalid the sensels with low usefulness.

The panic signal is useful to implements a simple adaptive mechanism which does not need fixed thresholds. The signal is given the semantics that the controller has detected inconsistencies in the observations.

The glue marks sensels as invalid if their usefulness is larger than a threshold α, which starts at 0: at the beginning, all sensels are marked as valid. When the panic signal is received, the glue component increases α by a fixed amount until it reaches a predefined maximum. This allows to discard as few sensels as needed to make the observations consistent.

VII. Application to range-finder data

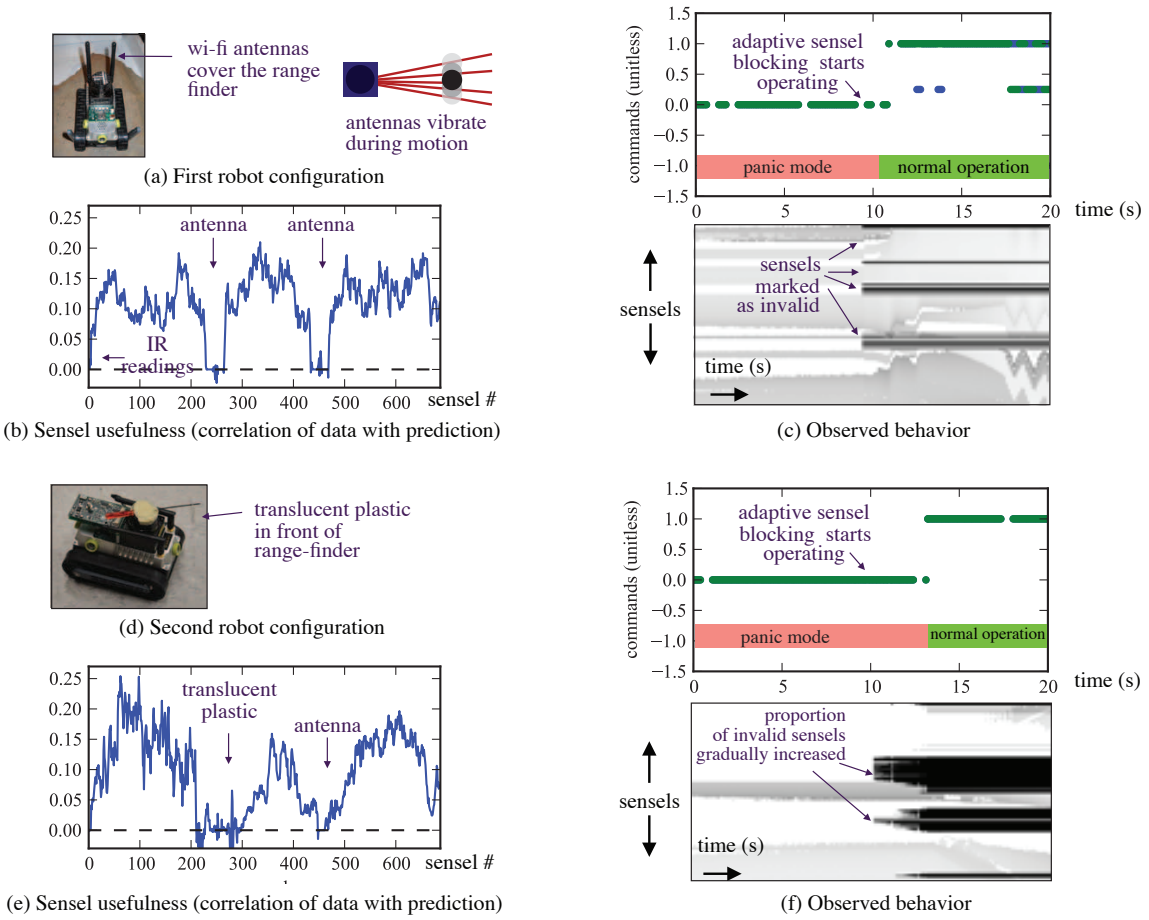
Platform: We use a small (~30cm) differential-drive robot called Landroid[3] with a Hokuyo range-finder on board [9], as well as 4 cameras, and 4 IR distance sensors pointed in the four orthogonal directions. We disrupted the sensor using two different configuration, shown in Fig. 7*a* and Fig. 7*d*. In the first configuration, the robot's WiFi antennas partially obstruct the sensor. Not all readings impacted end up with a fixed value because during motion the antennas vibrate; therefore, some sensels are reliable when the robot is still, and intermittently reliable during the motion. In the second configuration, we put some translucent tape in front of the sensor, spanning approximately 40° of the 270° field of view.

[3]This is a prototype produced by iRobot. See videos at http://www.irobot.com/gi/research/Advanced_Platforms/LANdroids_Robot

Data: The model is learned using about 15 minutes of data. During this period, the robot is programmed to move by randomly choosing all combinations in $\{-v_{\max}, 0, +v_{\max}\}$ for left and right track velocities. Each command is held for a random period, modeled as an exponential distribution with intensity 2s; the result is similar to a Levy flight. If the robot is about to collide, an automatic safety mechanism stops the logging and returns the robot to a safe position.

Learning details: To make things interesting, the bootstrapping agent is given a vector $\boldsymbol{y} \in \mathbb{R}^{691}$ containing both the 687 range-finder readings as well as the 4 IR-readings. All data is normalized in the $[0, 1]$ interval. The commands u^0, u^1 as the left and right track velocity. We model the sensorimotor cascade using a BDS, and we learn the model using (2). The resulting tensors are not shown, but are very much similar to those reported in [10]. Fig. 7*be* show the computed sensel usefulness. The occlusions are correctly identified, as well as the 4 IR readings (pointing in the four orthogonal directions, these are too sparse to be fitted by a BDS model).

Results: In these experiments, the traditional controller consists of a simple sensor-based exploration algorithm that takes the range-finder data as input to build a local map on which to plan. The algorithm sends out a panic signal if the range readings are inside a given threshold equal to the robot safety radius. Fig.7*cf* show the results of applying the adaptive sensel selection algorithm on this robot. When the episode starts, the bootstrapping agent already has a model of the sensorimotor cascade, based on which it computed a usefulness measure of the sensels. In the beginning, the traditional controller panics as there is no safe action to take. The adaptive algorithm removes the need of defining a fixed threshold for reliability. The "glue" starts operating around $t = 10$. It detects the panic signal from the exploration agent, and begins



(a) First robot configuration

(b) Sensel usefulness (correlation of data with prediction)

(c) Observed behavior

(d) Second robot configuration

(e) Sensel usefulness (correlation of data with prediction)

(f) Observed behavior

Figure 7. This figure illustrates the behavior of the architecture described in Fig. 6. In these experiments, the sensels $\boldsymbol{y}$ are the union of the range-finder sensels plus 4 IR sensors. (a) In the first configuration, the range-finder is obstructed by the robot's antennas. Moreover, the antennas vibrate during motion, so that the readings change in time. (b) The resulting sensor reliability curve (obtained by a BDS model) clearly shows the antennas and the IR sensors as static faults. (c) At the beginning of the run, the controller has no available action to take, because from its point of view the robot is already colliding with obstacles (the readings from the antennas appear to be *inside* the robot radius). The controller alerts the glue component in Fig. 6 of this impossibility with a panic signal. In response to the panic signal, the glue component raises the threshold (initially set to 0) for sensels to be consider reliable (this adaptive response removes the need of defining a fixed threshold). (d) Invalid sensels are displayed her with black lines. Once those sensels are marked as invalid, the controller gets out of the panic mode and starts moving. (e) In the second sets of experiments, a translucent piece of plastic is suspended in front of the sensor. In (g) one can see that more and more sensels are progressively removed, until all problematic sensels are removed, and the controller gets out of the panic mode and resumes normal operation.

increasingly marking unreliable sensels as invalid. In the first case (subfigure *c*), the problematic sensels are removed almost instantaneously; in the second case (subfigure *f*) one can notice the gradual increase in the fraction of invalid sensels removed. When all the invalid sensels are removed, the exploration algorithm returns in the normal operation mode and the robot starts exploring.

VIII. Discussion

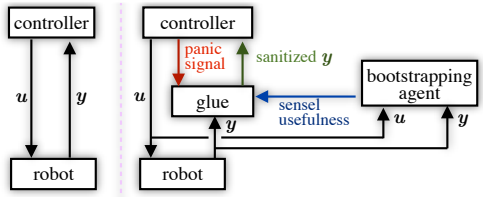
When does this work—the evident assumptions: The method described in this paper works only under certain assumptions, which we recall here. Firstly, there is the assumption that the data comes from a robotic sensorimotor cascade. This assumption allows the information-theoretic characterization of faults described in Section II. Then, one has to assume that this theoretical notion of faults corresponds to the real phenomena of interest. We have shown that this is the case for many faults (dead sensels, de-synchronized data, sensor occlusions, etc.), but there certainly are others that would not be detected with this criterion. The definition of faults we use is independent of the class of models used; however, in practice, one has to commit to a class of models, and assume that it is expressive enough to represent the physical systems of interest. In this paper we used BDS/BGDS models. Albeit they do *not* fully capture the dynamics of cameras (because of the hidden state of nearness) and range-finders (because of an unmodeled nonlinearity [5]), the approximation seems to be good enough to detect many of the common faults. Their main limitation is that they implicitly assume that the commands $\boldsymbol{u}$ are kinematic commands (velocities imposed to a rigid body), so they would fail if $\boldsymbol{u}$ represented torques/forces or was a nonlinear function of kinematic commands.

...and the hidden assumptions: In a recent paper [11], we argued that, while the goal of bootstrapping is to design agents that have no prior assumptions on the model, it is actually very hard to do so, as most methods have more or less hidden assumptions about the world. One way to highlight the hidden assumptions of a method is to consider its sensibility to "representation nuisances" applied to the observations and commands; these are represented by $\mathcal{G}^Y$ and $\mathcal{G}^U$ in Fig. 1. These nuisances are fixed, invertible transformations of the

signals; they change the representation but not the informative content. Therefore, one expects that the result of the method (in this case, the detected faults) be invariant to such changes. If this is not the case, it means that there are hidden assumptions in the method, or that there are certain unstated biases.

In [4] we discussed the invariance of BDS models to linear transformations of observations and commands, and the invariance of BGDS models to reparametrization of the domain $\mathcal{S}$. However, it turns out that it is actually very hard to maintain these invariance properties in practice, as they can be broken by seemingly innocuous operations. For example, in the experiments in Section V, the images are smoothed with a gaussian isotropic kernel of $\sigma = 0.5$ pixels prior to computing the gradients. This operation is standard in computer vision, but it hides two arbitrary choices. Firstly, the filter contains an arbitrary scale. This means that if the image was scaled up to twice the resolution, the effective scale of the filter would be halved. In some sense, choosing an arbitrary constant biases the agent towards features of a certain scale. Even worse is the effect of an arbitrary diffeomorphism: the gaussian filter is *isotropic* only for a certain parametrization and it is not invariant even to simple linear transformations such as shear. As another example, for computing the derivative $\dot{\boldsymbol{y}}$, we used the usual finite-difference approximation, another operation above suspicion. However, note that this is not invariant to the time step used. This means that, if we had the same observations, but coming at twice the frequency, we could have obtained different results, even though the sampling frequency has nothing to do with the faulty status of the sensor/actuator (using a higher-order filter would not help either).

The difficulty of maintaining the invariance properties is what keeps us sticking to simple models (BDS and BGDS), for which the invariance properties are easy to understand, rather than to look for more expressive models or more sophisticated learning techniques. For example, the elements of the various tensors are estimated independently from one another, even though it is reasonable to expect that the variation be smooth in s, except at edges, which could be achieved by regularization techniques (see, e.g., [12]). We noticed the similarities of the BGDS models applied to camera data to "optic flow fields"; several improvements are possible to make estimation more robust (see, e.g. [13]). Markov random fields [14] have been shown to learn very high-order statistics of raw data, and such techniques could help in fault detection. However, all these techniques come with a large amount of complexity



(a) Traditional architecture (b) Proposed architecture

Figure 6. We consider an architecture (right), in which an extra layer is put between a traditional controller and robot. A bootstrapping agent builds an internal model of the sensorimotor cascade, and produces a "sensels usefulness" signal. The traditional controller is modified to include a "panic" signal, which signals to a "glue" component when no action can be taken based on current observations. The glue marks sensels as invalid in inverse order of usefulness until the panic signals lasts. This minimal architecture change allows the robot to deal with sensor obstruction and various failures modes, by progressively marking sensels as invalid, without committing to predefined thresholds.

and parameters, while we saw that, from a bootstrapping perspective, it is hard to justify in an absolute sense even simple operations such as gaussian smoothing.

IX. Conclusions and Future Work

Designing agents that are more "aware" of the quality of their sensor data and the effectiveness of their actuators seems to be one of the keys for a more widespread adoption of robots in non-engineered, natural environments. A zero-prior-information approach, such as the one described in this paper, seems an attractive option for those cases in which one expects to not be able to model every possible disturbance that might appear during the system operation.

As for future work, we mention the ever-lasting problem of studying models that are able to capture a larger class of systems (for example robots with dynamics more complicated than pure kinematics implied by BDS/BGDS models; see [15] for preliminary work) as well as with being invariant to a larger class of nuisances (e.g., nonlinear scaling of sensel values). More on the engineering side, it is interesting to think of ways to give an active role of the fault detection mechanism (e.g., in case of doubts on the fault status, take control of the platform to do some self-diagnostic maneuver).

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

References

[1] A. Stoytchev, "Some Basic Principles of Developmental Robotics," *IEEE Trans. on Autonomous Mental Development*, vol. 1, no. 2, 2009. DOI.

[2] S. Simani, C. Fantuzzi, and R. J. Patton, *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Advances in Industrial Control, Springer, 2002.

[3] S. X. Ding, *Model-based fault diagnosis techniques*. Springer, 2008.

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

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

[6] S. Amari and H. Nagaoka, *Methods of information geometry*, vol. 191. Oxford University Press, 2000.

[7] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[8] Y. Amir, C. Danilov, M. M. Amir, J. Schultz, and J. Stanton, "The Spread toolkit: Architecture and performance," Tech. Rep. CNDS-2004-1, Johns Hopkins University, Center for Networking and Distributed Systems, Baltimore, MD, USA, 2004. (link).

[9] 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.

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

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

[12] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Simul*, vol. 4, 2005.

[13] R. Roberts, C. Potthast, and F. Dellaert, "Learning general optical flow subspaces for egomotion estimation and detection of motion anomalies," in *Conf. on Computer Vision and Pattern Recognition*, 2009. DOI.

[14] A. Saxena, S. Chung, and A. Ng, "3-d depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, 2008. 10.1007/s11263-007-0071-y. (link).

[15] A. Censi and R. M. Murray, "Learning diffeomorphism models of robotic sensorimotor cascades," 2011. Technical report. (link).