Bootstrapping Vehicles
a formal approach
to unsupervised sensorimotor learning
based on invariance

Acknowledgements:
Richard Murray
Stefano Soatto
Michael Dickinson
Andrew Straw
Shuo Han
Sawyer Fuller
Magnus Hakansson
Davide Scaramuzza
Scott Livingston

Andrea Censi
The progress of autonomous robotics

past

Kuka KR240

AGV

“autonomous” ground vehicles

present

2000

perception improvements

Mint floor cleaner

Kiva’s robotic warehouse

Mars rovers

future?

a robot butler in every house

Google car

Willow Garage’s PR2

“autonomous” ground vehicles

The progress of autonomous robotics
Challenges in building robots

- We still cannot match nature’s performance.
Challenges in controlling robots

- Robotics is too complex for a deductive approach.

- The complexity is often irreducible: there are no clean compositional properties.

- Robots act in an open, unstructured, changing, unpredictable world.
  - Robots must be able to learn and verify knowledge about themselves and the world.
• “Learning” and “adaptivity” can be formalized in different ways, at different scales.

- Passive/unembodied
- Ontogenic
- Phylogenetic
- Memetic

**Bootstrapping**

- A form of ontogenic learning.
- How can an embodied agent learn to use an unknown body?
• A robot is the interconnection of sensors and actuators, physically interacting with the external world.

• The “agent” is the software choosing the commands for the actuators on the basis of the observations from the sensors.
• In the **bootstrapping scenario**, the agent has no prior knowledge about its sensors, its actuators, and the external environment.
In the bootstrapping scenario, the agent has no prior knowledge about its sensors, its actuators, and the external environment.
Why working on bootstrapping

• **Intellectual** motivation
  • One of the main problems of AI is grounding symbols in sensorimotor experience.

• **Scientific** interest
  • Modeling “general purpose” cortex circuitry
  • Modeling mental developmental theories

• **Engineering** motivation
  able to learn models $\rightarrow$ less design effort
  able to verify models $\rightarrow$ safer and more resilient operation
Long term goals

- Detect and adapt to changes in sensor/action models.

- **Learn** to use any **new sensor/actuators** attached to the robot to improve performance.

(today: proofs of concept)
Goal: develop a systematic treatment of bootstrapping for robotics.

Approach:

- Strongly embodied, in the Vehicles universe.

- “Disciplined” analysis.

- Compositional synthesis.
1. First levels of bootstrapping
   *From bits to sensor geometry*

2. The Vehicles universe.
   *Sensel-level description of sensorimotor cascades.*

3. Learning robotic sensorimotor cascades
   *Focus: BDS models.*

4. Bootstrapping tasks
   *Examples: servoing, anomaly detection.*

5. Assumptions on representation and invariance properties.

6. Modular analysis tools
   *Use case: sensor calibration.*

7. Outlook
1. **First levels of bootstrapping**  
   *From bits to sensor geometry*

2. **The Vehicles universe.**  
   *Sensel-level description of sensorimotor cascades.*

3. **Learning robotic sensorimotor cascades**  
   *Focus: BDS models.*

4. **Bootstrapping tasks**  
   *Examples: servoing, anomaly detection.*

5. **Assumptions on representation and invariance properties.**

6. **Modular analysis tools**  
   *Use case: sensor calibration.*

7. **Outlook**

- **Assumptions** of the agent  
  - physical system  
  - format of the data  
  - representation/semantics

- **From bits to signals**

\[
y_t = \begin{array}{cccc}
\triangle & \square & \square & \circ \\
\square & \clubsuit & \clubsuit & \heartsuit \\
\square & \heartsuit & \heartsuit & \heartsuit \\
\diamondsuit & \spadesuit & \spadesuit & \spadesuit \\
\end{array}
\quad \rightarrow \quad y'_t = \begin{array}{cccc}
2 & 2 & 3 & 3 \\
2 & 4 & 4 & 5 & 4 \\
2 & 3 & 4 & 5 & 5 \\
6 & 6 & 7 & 7 & 6 \\
\end{array}
\]

- **From signals to sensor geometry reconstruction**
What are the agent’s assumptions?

- Assumptions about the **physical system**.
  
  *causality*  
  *it is a robotic body in 4D space-time*  
  *observations are slowly changing*  
  *noise is “small” with respect to the signal*

- Assumptions about the **format** of observations and commands (u).

  
  \[ y_t \in \mathbb{R} \quad y_t \in \{1, 2, 3, 4, 5, 6, 7\} \quad u_t \in \mathbb{R}^m \]

  
  \[ y_t : S \rightarrow \mathbb{R} \quad y_t \in \{\triangle, \square, \bigcirc, \blacklozenge, \heartsuit, \diamondsuit, \spadesuit\} \quad u_t \in \{\text{start, stop}\}\]

- Assumptions about the **semantics / representation** of the data:

  \"1\" is \"close\" to \"2\"

  \"1\" is \"closer\" to \"2\" than it is to \"3\"

  (described by group actions, will be the last topic today)
What are the **assumptions** of an agent about the data representation?

- Physical system
- Sensels
- All equivalent representations
  - Gray code
- Shuffled bits
- Encrypted bits

Mathematical expressions:

\[ y : S^1 \rightarrow \mathbb{R} \]

\[ y \in \mathbb{R}^n \]

\[ y \in \{■, □\}^{nb} \]

\[ y \in \{■, □\}^{nb+\Delta} \]
First levels of a bootstrapping agent

- Interpreting bits as signals.
- Interpreting scrambled sensels as part of spatially coherent data.
- Inferring instantaneous action models.
- Inferring long-horizon action models.
- Inferring behaviorally relevant states.

Remixed from [Kuipers’10]
From bits to signals

format: \( y_t \in \{\bigtriangleup, \square, \bigcirc, \clubsuit, \heartsuit, \diamondsuit, \spadesuit\} \)  \hspace{1em} (uninterpreted symbols)

assumptions: “observations represents a one-dimensional numeric signal”
“observations change slowly”

observations
\( y_t = \bigtriangleup \square \square \bigcirc \bigcirc \bigcirc \diamondsuit \diamondsuit \diamondsuit \spadesuit \spadesuit \spadesuit \)

statistics
transition matrix

reconstructed order

symmetries:
reverse ordering

“feature”: transition matrix is band-diagonal

signal
\( y_t' = 1 \ 2 \ 2 \ 3 \ 3 \)
\( \hspace{1em} 2 \ 4 \ 4 \ 5 \ 4 \)
\( \hspace{1em} 2 \ 3 \ 4 \ 5 \ 5 \)
\( \hspace{1em} 6 \ 6 \ 7 \ 7 \ 6 \)
First levels of a bootstrapping agent

Remixed from [Kuipers’10]

- Interpreting bits as signals.
- Interpreting scrambled sensels as part of spatially coherent data.
- Inferring instantaneous action models.
- Inferring long-horizon action models.
- Inferring behaviorally relevant states.

...
• The intrinsic calibration describes each pixel’s direction on the visual sphere.

• Traditional calibration procedures use feature extraction and known patterns.

D. Scaramuzza’s arm (coauthor)
• Intuition: the statistics of the data encode the sensor geometry.

• PROP: Assuming random motion, the correlation between pixel values is an unknown function \( f \) of the pixel distance on the visual sphere.

\[
\text{corr}(y_i, y_j) = f(d(s_i, s_j))
\]
From signals to sensor geometry

format: \( y_t \in \mathbb{R}^n \) (scrambled sensels)

assumptions: observations are scrambled sensels of a 2D sensor (camera/range-finder)
sensor motion is random

observations scrambled sensels

statistics correlation matrix

solution reconstructed sensels position

reconstructed image

symmetries: isometries

“feature”: correlation is a function \( f \) of the pixel distance:
\[
\text{corr}(y_i, y_j) = f(d(s_i, s_j))
\]
First levels of a bootstrapping agent

- Interpreting bits as signals.
- Interpreting scrambled sensels as part of spatially coherent data.
- Inferring instantaneous action models.
- Inferring long-horizon action models.
- Inferring behaviorally relevant states.

...
1. First levels of bootstrapping
   *From bits to sensor geometry*

2. The Vehicles universe.
   *Sensel-level description of sensorimotor cascades.*

3. Learning robotic sensorimotor cascades
   *Focus: BDS models.*

4. Bootstrapping tasks
   *Examples: servoing, anomaly detection.*

5. Assumptions on representation and invariance properties.

6. Modular analysis tools
   *Use case: sensor calibration.*

7. Outlook

- **Scenario:** an extended version of Braitenberg’s *Vehicles* universe.

- **Modeling** at the sensel level:
The dynamics of different sensors are more similar than expected.

\[
\begin{align*}
\dot{y}(s) &= (s \times \nabla_s y(s)) \cdot \omega + \nabla_s y(s) \cdot v \\
\dot{y}(s) &= (s \times \nabla_s y(s)) \cdot \omega + (\nabla_s \log y(s) - s) \cdot v \\
\dot{y}(s) &= (s \times \nabla_s y(s)) \cdot \omega + \mu(s) \nabla_s y(s) \cdot v
\end{align*}
\]
Braitenberg’s Vehicles

• Complex behavior arises from simple sensorimotor interaction.

\[ u = y_L - y_R \]

\[ u = y_R - y_L \]
• **Our universe** is an upgraded version of Braitenberg’s Vehicles.
• Simple enough for analytical results.
• Captures relevant phenomena of embodied intelligence.
Robotic actuators

manipulators

prismatic joint

revolute joint

wheeled dynamics

diff. drive

car-like

omnidirectional

derived

legged snakes

... future work

touch proprioception

... related or derived

compass

sonars

GPS

... "canonical" sensors

field sampler

range-finder

camera

temperature,

concentration,

...
The "world" $W$ (or "sensorimotor cascade") is the interconnection of sensors, actuators, and **external environment**.

```
"world" $W \in \times \times$
```

```
\begin{figure}
\centering
\begin{tikzpicture}
  \node [agent] (agent) {agent};
  \node [sensorimotor cascade] (sensorimotor) [below of=agent, xshift=-2cm] {
    \textbf{"world" or sensorimotor cascade}
  }
  \node [external environment] (environment) [below of=sensorimotor, xshift=-2cm] {
    \textbf{\textit{external}}
    \textbf{\textit{environments}}
    An important part, somewhat skipped for today.
  }
  \node [unknown sensor(s)] (unknown_sensors) [left of=environment, xshift=-2cm] {
    \textbf{unknown sensor(s)}
  }
  \node [unknown actuator(s)] (unknown_actuators) [right of=environment, xshift=-2cm] {
    \textbf{unknown actuator(s)}
  }

  \draw [->] (agent) -- node [above] {$y$} (sensorimotor);
  \draw [->] (sensorimotor) -- node [below] {$u$} (agent);
  \draw [->] (sensorimotor) -- (environment);
  \draw [->] (environment) -- (unknown_sensors);
  \draw [->] (environment) -- (unknown_actuators);
\end{tikzpicture}
\end{figure}
```
Strategy

• Work at the raw “sensel” level (sensory element)

\[ y(s) = \begin{cases} 
\text{luminance in direction } s & s \in S^2 \\
\text{distance in direction } s & s \in S^2 \\
\text{intensity at point } s & s \in \mathbb{R}^3 
\end{cases} \]

• We try to understand what is common among the dynamics of all robots.
• The dynamics of a camera are **bilinear** for 1D rotational motion.

\[
\begin{align*}
y(s, t) & : \text{luminance in direction } s \\
& \text{at time } t
\end{align*}
\]

**dynamics**
\[ \dot{\theta} = \omega \]

**sensor model**
\[ y(s, t) = m(\theta(t) + s) \]

**sensor dynamics**
\[ \dot{y}(s, t) = \nabla_s y(s, t) \omega(t) \]

- Linear in angular velocity
- Linear in \( y \) (\( \nabla \) is a linear operator)

\( m(\cdot) \) describes the environment texture

\( s \in S^1 \)
• The dynamics of range-finder and camera are the same for pure rotations.

\[ \dot{y}(s) = \nabla_s y(s) \omega \]
The dynamics of range-finder and camera are the same for pure rotations.

\[ \dot{y}(s) = \nabla_s y(s) \cdot \omega \]

**3D case**

\[ \dot{y}(s) = (s \times \nabla_s y(s)) \cdot \omega \]

\( s \in S^2 \quad \omega \in \mathbb{R}^3 \)

\( \times \) vector product in \( \mathbb{R}^3 \)

\( \cdot \) inner product in \( \mathbb{R}^3 \)

\( y(s): \text{luminance} \quad y(s): \text{distance} \)
• For translational motion, the range finder dynamics has a nonlinearity.

\[ \dot{y}(s) = \cdots + (\nabla_s \log y(s) - s) \cdot v \]

• The camera has a hidden state, which is the distance to the objects.

\[ \dot{y}(s) = \mu(s) \nabla_s y(s) \cdot v + \cdots \]

“nearness”: inverse of the distance to the obstacles

• Near occlusions, the dynamics are discontinuous.
• Even in 6 degrees of freedom, for all three sensors, the dynamics are “mostly” \textbf{bilinear}, up to hidden states and nonlinearities.

\[
\dot{y}(s) = (s \times \nabla_s y(s)) \cdot \omega + \nabla_s y(s) \cdot v \\
\text{field-sampler} \quad \text{exactly bilinear}
\]

\[
\dot{y}(s) = (s \times \nabla_s y(s)) \cdot \omega + (\nabla_s \log y(s) - s) \cdot v \\
\text{range-finder} \quad \text{bilinear, up to a nonlinearity}
\]

\[
\dot{y}(s) = (s \times \nabla_s y(s)) \cdot \omega + \mu(s) \nabla_s y(s) \cdot v \\
\text{camera} \quad \text{bilinear, up to a hidden state}
\]
1. **First levels of bootstrapping**  
*From bits to sensor geometry*

2. **The Vehicles universe.**  
*Sensel-level description of sensorimotor cascades.*

3. **Learning robotic sensorimotor cascades**  
*Focus: BDS models.*

4. **Bootstrapping tasks**  
*Examples: servoing, anomaly detection.*

5. **Assumptions on representation and invariance properties.**

6. **Modular analysis tools**  
*Use case: sensor calibration.*

7. **Outlook**

---

- **Simple models for learning robotic sensorimotor cascades.**  
  *Focus: BDS models*

  \[ \hat{y}^s = \sum_i \sum_v M_{iv}^s y^v u^i \]

- **Extensions and challenges**
First levels of a bootstrapping agent

- Interpreting bits as signals.
- Interpreting scrambled sensels as part of coherent data.
- Inferring instantaneous action models.
- Inferring long-horizon action models.
- Inferring behaviorally relevant states.

...
What we will see

Vehicles simulations

robots  logged data

What the agent sees

“uninterpreted” observations.

“uninterpreted” commands

Abstractions built with ROS at the software level
System identification philosophy

- Find a class \( C \) of dynamical systems that **subsumes** or **approximates** the systems of interest.

  - “learning” = maximum likelihood in the class \( C \) (projection, if \( W \) lies outside)
  
  - “policies” = manually designed control laws, assuming the world \( W \in C \).
Different classes of models offer different tradeoffs of representation power and efficiency.

- BDS
  
  Models a bilinear relation between $\dot{y}$, $y$, and $u$.
  
  [ICRA’11]

- BGDS
  
  Models bilinear flows of the observations space. 
  
  more efficient
  
  [IROS’11]

- DDS
  
  Models diffeomorphisms of the observations space. 
  
  non-affine kinematics long-horizon predictions
  
  [ICRA’12a]
The BDS model assumes a generic bilinear relation among $y$, $u$, and $\dot{y}$.

$$\dot{y}^s = \sum_i \sum_v M_{iv}^s y^v u^i \quad y \in \mathbb{R}^n \quad u \in \mathbb{R}^k$$

The $k \times n \times n$ tensor $M$ consists of one $n \times n$ matrix for each command.

Learning the one-dimensional dynamics

- In this case, the $M$ matrix approximates the gradient operator.

**real dynamics:** $\dot{y} = \nabla_s y \omega$

**postulated model:** $\dot{y} = My \omega$

**learned parameters:**

$$\dot{y} = \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} y \omega$$
The BDS model assumes a generic bilinear relation among $y$, $u$, and $\dot{y}$.

$$\dot{y}^s = \sum_i \sum_v M_{iv}^s y^v u^i \quad y \in \mathbb{R}^n \quad u \in \mathbb{R}^k$$

The $k \times n \times n$ tensor $M$ consists of one $n \times n$ matrix for each command.

**Estimation of $M$**

$M$ is estimated with streaming algorithms: $\dot{M} = f(M, y, u)$

$u = \omega$

Total: ~4 hours of log data
Bilinear Dynamics Sensors (BDS)

- The BDS model assumes a generic bilinear relation among $y$, $u$, and $\dot{y}$.

$$\dot{y}^s = \sum_i \sum_v M_{iv}^s y^v u^i \quad y \in \mathbb{R}^n \quad u \in \mathbb{R}^k$$

The $k \times n \times n$ tensor $M$ consists of one $n \times n$ matrix for each command.

**Estimation of $M$**

- Robust to some failure of the assumptions.

*Example: the antennas obstruct part of the range finder readings.*

Those readings do not contribute to the model.

$iRobot Landroid$

$u_1 = left \ track \ velocity$

$M_1$
The BDS model assumes a generic bilinear relation among $y$, $u$, and $\dot{y}$.

$$\dot{y}^s = \sum_i \sum_v M_{iv}^s y^v u^i \quad y \in \mathbb{R}^n \quad u \in \mathbb{R}^k$$

The $k \times n \times n$ tensor $M$ consists of one $n \times n$ matrix for each command.

Planar case
Bilinear Dynamics Sensors (BDS)

- The BDS model assumes a generic bilinear relation among $y$, $u$, and $\dot{y}$.

$$\dot{y}^s = \sum_i \sum_v M_{iv} y^v u^i \quad y \in \mathbb{R}^n \quad u \in \mathbb{R}^k$$

The $k \times n \times n$ tensor $M$ consists of one $n \times n$ matrix for each command.

Planar case

The sensels are "messy" because they are not well ordered.
• The Vehicles universe provides many challenges.
• The *Vehicles* universe provides many **challenges**.

- **The BDS model** is $O(n^2)$.
  \[
  y(t) = \sum_i \sum_v M^v_{iv} y^v(t) u^i
  \quad \text{\(O(n^2)\)}
  \]

- **The BGDS model** assumes the observations are a function on a manifold and that the dynamics is a function of the gradient.
  \[
  y(s) = \sum_i \sum_j G^j_{i}(s) \nabla_j y(s) u^i
  \quad \text{\(O(n)\)}
  \]

  **efficiency**
  
  **non-instantaneous models**
  
  **hidden states**
  
  **higher-order analysis**

  **assumes the sensor geometry is known.**

[IROS'11]
• The *Vehicles* universe provides many **challenges**.

- **We cannot use a model of** \( \dot{y} \) **for long term predictions.**
- **Idea:** learn a **diffeomorphism** \( \phi_i \) **associated to each command** \( u_i \):
  
  \[
  y(s, T) = y(\phi_i(s), 0)
  \]

[ICRA’12a]
1. First levels of bootstrapping
   *From bits to sensor geometry*

2. The Vehicles universe.
   *Sensel-level description of sensorimotor cascades.*

3. Learning robotic sensorimotor cascades
   *Focus: BDS models.*

4. Bootstrapping tasks
   *Examples: servoing, anomaly detection.*

5. Assumptions on representation and invariance properties.

6. Modular analysis tools
   *Use case: sensor calibration.*

7. Outlook
How to specify the agent behavior?

- Extrinsic motivation
  e.g., *an opaque reward*

- Intrinsic motivation
  e.g., *“curiosity”*

- Explicit tasks
  - spatial tasks
    e.g., *servoing*
  - information tasks
    e.g., *prediction*
How to specify the agent behavior?

- Extrinsic motivation
e.g., *an opaque reward*
- Intrinsic motivation
e.g., “curiosity”
- Explicit tasks
  - spatial tasks
e.g., *servoing*
  - information tasks
e.g., *prediction*

✓ Very generic formulation

~ Who designs the reward?

~ “opaque” agents

✓ Can guide exploration

✓ Composable

✓ Agent must learn a generative model.
Two-stage protocol

1. “Learning” during an exploration phase
2. “Acting” in a separate phase

The same agent in different scenarios

- simulation
  - $u$ chosen by “motor babbling”

- robots
  - *iRobot Landroid*

- passive logs
  - $u_0$ chosen by someone else
Properties of a good task for bootstrapping agents

✓ It can be stated for different sensorimotor cascades.
✓ It reveals a basic skill of embodied intelligence.

Generalized servoing

• Servoing: Given the goal observations $y^*$, minimize $\|y - y^*\|$. 
• **Servoing:** Given the goal observations \( y^* \), minimize \( \| y - y^* \| \).

\[
u^i = \sum_s \sum_v y^v M^s_{iv} (y^*_s - y_s) \quad \text{(gradient descent assuming bilinear dynamics)}
\]
• Exactly the **same agent** (same code, no parameters tuned) works for **different sensorimotor cascades**.

1. **Training phase**
   
   *(motor babbling)*
• Exactly the **same agent** (same code, no parameters tuned) works for **different sensorimotor cascades**.

1. **Training phase**
   
   *(motor babbling)*

2. **Acting phase**
• **Servoing with second-order dynamics:** add a **damping term**

\[ u^i = \sum_s \sum_v y^v M^s_{iv} (y^*_v - y_v) - K_d \sum_s \sum_v y^v M^s_{iv} y_v \]

• The resulting control law is “bio-plausible”.

---

[CDC’09, IROS’10]

*pre-bootstrapping work (tensor M is hand-coded)*

with Shuo Han,
Sawyer Fuller
Simple skills can be composed to perform more complicated tasks.

*Example*: servoing → waypoint navigation
- Goal: a **hierarchy of tasks** for checking the skills needed by embodied agents.
• **Prediction:** given $y$ and $u$, predict the instantaneous change $\dot{y}$.

\[ \dot{y}(s) = \sum_i \sum_j G_i^j(s) \nabla_j y(s) u^i \]
• **Anomaly detection**
  Assuming a static world, failures of prediction are likely to be other agents (or equally interesting things).

\[
\begin{align*}
\text{observed } \dot{y}(t) & \quad \text{predicted } \dot{y}(t)
\end{align*}
\]

"anomaly"

- other agents and occlusions

- changes due to self-motion are anticipated
• Sensor “faults” can be detected by averaging anomaly. [ICRA’12b]

• Same idea works for actuator “faults” (not shown)
  e.g., detect if the planarity assumption fails;
  detect if there are delays in the commands application
1. First levels of bootstrapping
   *From bits to sensor geometry*

2. The Vehicles universe.
   *Sensel-level description of sensorimotor cascades*

3. Learning robotic sensorimotor cascades
   *Focus: BDS models*

4. Bootstrapping tasks
   *Examples: servoing, anomaly detection*

5. Assumptions on representation and invariance properties

6. Modular analysis tools
   *Use case: sensor calibration*

7. Outlook

- “Disciplined” bootstrapping requires specifying:
  1. The assumptions on the physical system.
  2. The assumptions on the representation of the data

- Necessary invariance properties for optimal agents
• What are the **assumptions** of an agent about the data representation?
• Representation is a nuisance that must be rejected.

\[
\begin{align*}
\text{Invertible:} & \quad \text{the devil does not lie, it only confuses you.} \\
\text{Fixed:} & \quad \text{the devil choses only once.}
\end{align*}
\]
agent’s assumptions about the representation

transformations of the data that can be tolerated
assumptions (not) needed about the data representation \[\leftrightarrow\] information-preserving \textbf{fixed} transformations tolerated

\begin{itemize}
  \item \textit{“the agent has no assumptions on the ordering (labeling) of the signals”}
  \item \textit{“the agent can tolerate a random (but fixed) permutation of the signals”}
\end{itemize}
assumptions (not) needed about the data representation

information-preserving fixed transformations tolerated

No assumptions on the sensor’s intrinsic calibration.

The agent can tolerate diffeomorphisms of the image.

\[ y : \mathcal{S} \rightarrow \mathbb{R} \]

\[ \dot{y}(s) = \sum_i \sum_j G_i^j(s) \nabla_j y(s) u^i \]

• PROP: The BGDS model has this invariance.
No assumptions on basis choice for $u/y$.

The agent can tolerate a linear transformation of $u/y$.

\[ u = (v, \omega) \quad \leftrightarrow \quad \tilde{u} = (v_L, v_R) \]

\[ \tilde{u} = A u \]

- PROP: The BDS model has this invariance.

\[ \dot{y}^s = \sum_i \sum_v M^s_{iv} y^v u^i \]
No assumptions at all?  

The agent tolerates all possible transformations:

\[ u \mapsto \chi(u) \quad y \mapsto \gamma(y) \]

\(\text{sensels} \quad \longleftrightarrow \quad \text{encrypted}\)
• **Representation nuisances**
  are transformations of the data represented by two groups $G^Y, G^U$.

The two groups are subgroups of all automorphisms of $U$ and $Y$:

$$G^U \leq \text{Aut}(U) \quad G^Y \leq \text{Aut}(Y)$$

They induce an action on the world $W \in C \subset D(Y;U)$.

$$W \boxtimes hWg$$

$g \in G^u, \quad h \in G^y$
If the agent is optimal with respect to physical tasks/rewards, the behavior must be invariant to the representation nuisances.

The “physical” behavior must be the same, as nuisances do not change observability or controllability.
The agent’s commands must be **invariant** to a representation nuisances on the observations.

\[ y = h(y_0) \]

\[ G^Y \sim h \]

obtain same physical behavior in spite of the nuisance \( h \)
• Nuisances on the **commands**, instead, must be pre-compensated (because they act later).
• “Disciplined” bootstrapping requires specifying, for an agent:

  1) The subset $C$ of dynamical systems to which the world is assumed to belong.

  2) The transformations group $G = G^Y \times G^U$ to which the closed-loop behavior is invariant.

• The main problem of bootstrapping is enlarging $C$ or $G$.

**The geometry of bootstrapping**

- The action of $G = G^Y \times G^U$ on systems up to representation $C$ moves an equivalent representation of my robot.
• Invariance properties induce a partial order on the agents:

\[
\text{more powerful} = \text{less assumptions} = \text{more invariant}
\]

• Examples for the models seen previously:

<table>
<thead>
<tr>
<th>Model</th>
<th>Domain</th>
<th>Range</th>
<th>Invariances for ( u )</th>
<th>Invariances for ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDS</td>
<td>( u \in \mathbb{R}^k )</td>
<td>( y \in \mathbb{R}^n )</td>
<td>( \mathbf{GL}(k) )</td>
<td>( \mathbf{GL}(n) )</td>
</tr>
<tr>
<td>BGDS</td>
<td>( u \in \mathbb{R}^k )</td>
<td>( y : S \to \mathbb{R} )</td>
<td>( \mathbf{GL}(k) )</td>
<td>( \text{Diff}(S) )</td>
</tr>
<tr>
<td>DDS</td>
<td>( u \in \mathcal{U} )</td>
<td>( y : S \to \mathbb{R} )</td>
<td>( \text{Aut}(&lt;\mathcal{U}&gt;) )</td>
<td>( \text{Diff}(S) )</td>
</tr>
</tbody>
</table>

\[
\dot{y}^s = \sum_i \sum_v M_{iv}^s y_v u^i
\]

\[
\dot{y}(s) = \sum_i \sum_j G_i^j(s) (\nabla_j y(s) + B_i(s)) u^i
\]

\[
y_T(s) = y_0(\phi_i(s))
\]
1. First levels of bootstrapping
   *From bits to sensor geometry*

2. The Vehicles universe.
   *Sensel-level description of sensorimotor cascades*

3. Learning robotic sensorimotor cascades
   *Focus: BDS models*

4. Bootstrapping tasks
   *Examples: servoing, anomaly detection*

5. Assumptions on representation and invariance properties

6. Modular analysis tools
   *Use case: sensor calibration*

7. Outlook

- **Modeling**
  How to represent nuisances with group actions?

- **Analysis**
  Given an agent, how to find $C$ and $G$?

- **Design**
  How to design invariant agents? How to enlarge $C$ or $G$?

- **Extensions**
  From instantaneous to causally invertible nuisances.
• **Goal:** Reason about the **invariance properties** of bootstrapping components **interconnections**.

Legend:

- $\text{invariance}$
- $G \cdot \text{Id}$
• **Goal**: Reason about the **invariance properties** of bootstrapping components **interconnections**.

Legend:

- **invariance**
- **G** → **Id**

This agent’s behavior is invariant to diffeomorphisms of the image.
• **Goal:** Reason about the invariance properties of bootstrapping components **interconnections**.
- **Goal:** Reason about the invariance properties of bootstrapping components interconnections.

Legend:

- \( G \bullet \text{Id} \) for invariance

This agent's behavior is invariant to diffeomorphisms of the image.
• Image smoothing with a spherical kernel $k$

$$(y * k)(s) = \int_{v \in \mathcal{S}} k(d(s, v))y(v)d\mathcal{S}$$

**commutes** with isomorphisms.
• “Massaging the data” typically reduces the invariance. 
  (also filtering, regularization, etc.)

Legend: invariance equivariance unstructured
G • Id G G ?
"Massaging the data" typically reduces the invariance. (also filtering, regularization, etc.)

The series is invariant to isomorphisms…

\[ \text{Iso}(\mathcal{S}) \subset \text{Diff}(\mathcal{S}) \]

Legend:

- **invariance**
- **equivariance**
- **unstructured**
• “Massaging the data” typically reduces the invariance. (also filtering, regularization, etc.)

The series is invariant to isomorphisms...

\[ \text{Iso}(S) \subset \text{Diff}(S) \]

… but we don’t know what happens for diffeomorphisms that are not isomorphisms.

Legend:

- **invariance**
  - \( G \rightarrow \text{Id} \)

- **equivariance**
  - \( G \rightarrow G \)

- **unstructured**
  - \( G \rightarrow ? \)
• Intuition: the statistics of the data encode the sensor geometry.

PROP: Assuming random motion, the correlation between pixel values is an unknown function $f$ of the pixel distance on the visual sphere.

$$\text{corr}(y_i, y_j) = f(d(s_i, s_j))$$
Modeling with representation nuisances

- The calibration procedure must be **invariant to two nuisances**:

1. The function $f \in \text{Diff}(\mathbb{R})$ is a nuisance of the problem:
   $$\text{corr}(y_i, y_j) = f(d(s_i, s_j))$$

2. A permutation models the fact that pixels are scrambled.
Algorithm “CalibA”

- Jointly estimate both \(f\) and \(\{s_i\}\).
  (quite a bit of work!)

The shape is **metrically accurate**

**Invariant** to \(f\)

**Invariant** to ordering.

Algorithm “CalibB”

- Fix a “reasonable” \(f_0\)
- Obtain approximate distances:
  \[\tilde{d}(s_i, s_j) = f_0^{-1}(\text{corr}(y_i, y_j))\]
- Solve for \(\{s_i\}\) using MDS
  (multidimensional scaling).

The shape is **topologically accurate**

**Invariant** to ordering.

(The absolute orientation is unobservable)
Algorithm "CalibA"

- Jointly estimate both $f$ and $\{s_i\}$.
  (quite a bit of work!)

The shape is **metrically accurate**

**Invariant** to $f$

**Invariant** to ordering.

(The absolute orientation is unobservable)

Algorithm "CalibB"

- Fix a "reasonable" $f_0$
- Obtain approximate distances:
  $$\tilde{d}(s_i, s_j) = f_0^{-1}(\text{corr}(y_i, y_j))$$
- Solve for $\{s_i\}$ using MDS
  (multidimensional scaling).

The shape is **topologically accurate**

Not invariant to $f$.

**Invariant** to ordering.

Legend:

- **invariance**
- **equivariance**
- **unstructured**
- **nuisance introduced**
- **structured effect**
Algorithm “CalibA”

• Jointly estimate both \( f \) and \( \{s_i\} \).
(quite a bit of work!)

The shape is **metrically accurate**

\[
\text{Diff}(\mathbb{R}) \xrightarrow{\bullet} \text{Id}
\]

**Invariant** to ordering.

Algorithm “CalibB”

• Fix a “reasonable” \( f_0 \)
• Obtain approximate distances:
\[
\tilde{d}(s_i, s_j) = f_0^{-1}(\text{corr}(y_i, y_j))
\]
• Solve for \( \{s_i\} \) using MDS (multidimensional scaling).

The shape is **topologically accurate**

\[
\text{Diff}(\mathbb{R}) \xrightarrow{\bullet} \text{Diff}(S)
\]

**Invariant** to ordering.

(The absolute orientation is unobservable)

---

Legend:

- **invariance**
- **equivariance**
- **unstructured effect**
- **nuisance introduced**
- **structured effect**

- \( G \xrightarrow{\bullet} \text{Id} \)
- \( G \xrightarrow{\text{G}} \text{G} \)
- \( G \xrightarrow{?} \text{?} \)
- \( \text{Id} \xrightarrow{\bullet} \text{H} \)
- \( G \xrightarrow{\text{H}} \text{H} \)
Algorithm “CalibA”

- Jointly estimate both $f$ and $\{s_i\}$.
  (quite a bit of work!)

The shape is **metrically accurate**

- $\text{Diff}(\mathbb{R}) \rightarrow \bullet \text{Id}$
- $\text{Perm}(n) \rightarrow \bullet \text{Id}$

(The absolute orientation is unobservable)

Algorithm “CalibB”

- Fix a “reasonable” $f_0$
- Obtain approximate distances:
  $\tilde{d}(s_i, s_j) = f_0^{-1}(\text{corr}(y_i, y_j))$
- Solve for $\{s_i\}$ using MDS
  (multidimensional scaling).

The shape is **topologically accurate**

- $\text{Diff}(\mathbb{R}) \rightarrow \text{Diff}(\mathcal{S})$
- $\text{Perm}(n) \rightarrow \bullet \text{Id}$

Legend:

- **invariance**
  - $G \rightarrow \bullet \text{Id}$

- **equivariance**
  - $G \text{ G}$

- **unstructured**
  - $G \text{ ?}$

- **nuisance introduced**
  - $\text{Id} \rightarrow \text{H}$

- **structured effect**
  - $G \rightarrow \text{H}$
Algorithm “CalibA”

- Jointly estimate both $f$ and $\{s_i\}$.
  (quite a bit of work!)

The shape is **metrically accurate**

- $\text{Diff}(\mathbb{R}) \bullet \text{Id}$
- $\text{Perm}(n) \bullet \text{Id}$
- $\text{Id} \bullet \text{Iso} (\mathcal{S})$

Algorithm “CalibB”

- Fix a “reasonable” $f_0$
- Obtain approximate distances:
  \[
  \tilde{d}(s_i, s_j) = f_0^{-1}(\text{corr}(y_i, y_j))
  \]
- Solve for $\{s_i\}$ using MDS (multidimensional scaling).

The shape is **topologically accurate**

- $\text{Diff}(\mathbb{R}) \quad \text{Diff}(\mathcal{S})$
- $\text{Perm}(n) \bullet \text{Id}$
- $\text{Id} \bullet \text{Iso} (\mathcal{S})$

**Legend:**
- **invariance**
  - $G \bullet \text{Id}$
- **equivariance**
  - $G \quad G$
- **unstructured**
  - $G \quad ?$
- **nuisance introduced**
  - $\text{Id} \bullet H$
- **structured effect**
  - $G \quad H$
scrambled sensels → sensor geometry reconstruction → smoothing → agent → behavior

Legend:
- **invariance**
  - \(\text{Iso}(\mathcal{S})\) in blue
  - \(\text{Diff}(\mathcal{S})\) in red

- **equivariance**
  - \(\text{Iso}(\mathcal{S})\) in green

- **unstructured**
  - \(\text{Diff}(\mathcal{S})\) in orange

- **nuisance introduced**
  - \(\text{Id} \cdot H\) in yellow

- **structured effect**
  - \(\text{Id} \cdot H\) in orange
Legend: invariance, equivariance, unstructured, nuisance introduced, structured effect
Legend:

- invariance: $G \bullet \text{Id}$
- equivariance: $G \text{ blue } G$
- unstructured: $G \text{ red } ?$
- nuisance introduced: $\text{Id} \bullet \text{ orange } H$
- structured effect: $G \text{ red } H$
Legend:

- **invariance**
  - $G \bullet \text{Id}$

- **equivariance**
  - $G \quad G$

- **unstructured**
  - $G \quad ?$

- **nuisance introduced**
  - $\text{Id} \bullet \text{H}$

- **structured effect**
  - $G \quad \text{H}$
1. **First levels of bootstrapping**  
   *From bits to sensor geometry*

2. **The Vehicles universe.**  
   *Sensel-level description of sensorimotor cascades.*

3. **Learning robotic sensorimotor cascades**  
   *Focus: BDS models.*

4. **Bootstrapping tasks**  
   *Examples: servoing, anomaly detection.*

5. **Assumptions on representation and invariance properties.**

6. **Modular analysis tools**  
   *Use case: sensor calibration.*

7. **Outlook**
What my work so far has shown:

- Generic low level sensorimotor learning for a large class of systems is possible…
- …but being completely invariant to the representation is challenging.

Some future bootstrapping-related projects

1. Extensions to more complicated robots.
2. Completing the theoretical picture.
3. Integration within traditional robotic architecture.
Extend models/algorithms to more complicated systems

grid worlds

“Vehicles”

Things that run, jump, climb, and grab.
Completing the theoretical picture

\[ R \times T \times G \]

all bootstrapping problems roboticists care about

- Define formally \( R \), the set of all robots.
- Define formally \( T \), the sets of all tasks we might possibly care about.
- Define the set \( G \), of transformations corresponding to “reasonable” assumptions on the data representation.

- **Goal:** develop a systematic treatment of bootstrapping for robotics.
- Use \( R/T/G \) as a way to measure progress.
**Goal:** integrate these techniques in traditional robot software architectures. 

*e.g., who decides what are the signals \( y \) and \( u \)?

---

**Engineering issues**

- **original ROS “computation graph”**

- **computation graph augmented with bootstrapping modules**

- \( y \) → “one click” automated procedure → \( u \)

- ✓ the bootstrapping modules filters out the anomalies

- ✓ robust to sensor/actuators faults
The end.

Thanks for your attention!
Bootstrapping Vehicles

The Vehicles universe

Identification methods

\[ y^s(t) = \sum_i \sum_o M_{i,o} y^o(t) u^i \]

Challenges

Bootstrapping

Invariance properties of bootstrapping agents

Mature applications

Modular analysis

anomaly/fault detection

camera calibration

Perm(\(n\)) $\bullet$ Id

Diff(\(\mathbb{R}\)) $\bullet$ Diff(\(S\))

Id $\bullet$ Iso(\(S\))
Backup slides
• **Failure example:**
  the simple servo strategy
does not work
for car-like dynamics.
Proof techniques

- Assuming the system belongs to a given class, e.g., a camera mounted on an omnidirectional robot.
- ... and the environment satisfies certain conditions, e.g., the textures are rich enough (and the light is on!)
- ... and the training data is collected in a certain way, e.g., “motor babbling”
- ... then we learn an approximation (explicitly computed)
- ... which allows to perform a given task, e.g., servoing
- ... with a certain performance, e.g., local instead of global convergence
• The Vehicles universe provides many challenges.

- Mount a turret on an omnidirectional robot:

  - efficiency for rich data streams
  - non-instantaneous models
  - hidden states
  - higher-order analysis

The dynamics is too complex for BDS models.
• The *Vehicles* universe provides many **challenges**.

  - **efficiency for rich data streams**
  - **non-instantaneous models**
  - **hidden states**
  - **higher-order analysis**

• **An open problem**: behaviorally relevant states can be arbitrarily complicated functions of the [history] of the observations.

  - *e.g.*, can we bootstrap the notion of “depth” from first principles?
Invariant task design

• Tasks should be invariant too!

• The error function \( E(y) = \|y - y^*\| \) is not invariant to \( y \mapsto Ay \quad A \in \text{GL}(n) \)

The agent changes trajectory if the representation of data changes.
Invariant task design

- Tasks should be invariant too!

- The error function \( E(y) = ||y - y^*|| \) is not invariant to \( y \mapsto Ay \) \( A \in \text{GL}(n) \)

One possible fix

The error function \( E_0(y) = ||P^{-\frac{1}{2}}(y - y^*)|| \), with \( P = \text{cov}(y) \), is invariant to \( \text{GL}(n) \):

\[ P \mapsto APA^T \]

\( y_1 = A_1 y \)

\( y_2 = A_2 y \)
All algorithms have “hidden” assumptions

Assumptions = failed invariances

Watch out for hidden assumption in the algorithms...

• Let’s smooth the data...
• Let’s linearize the system...
• Let’s compute the correlation...
• Let’s cluster points...

...and in the problem formalization:

• Let’s minimize this error function...
• Let’s assume Gaussian noise...

\[
\min_{x \in \mathbb{R}^n} \|Ax - y\|_2
\]

base does not matter

\[
\min_{x \in \mathbb{R}^n} \|Ax - y\|_1
\]

base matters
$u_1 = \text{left wheel velocity}$

$u_2 = \text{right wheel velocity}$
Figure 3
Vehicles 2a and 2b in the vicinity of a source (circle with rays emanating from it). Vehicle 2b orients toward the source, 2a away from it.
**Sensori-motor models** have a hourglass structure.

* Sensor models have not had a mathematical formulation as rigorous as mechanics.
* Actuators kinematics/dynamics are well studied using tools from differential geometry.
A technical remark:
These invariance conditions are on the closed-loop behavior, after learning.

Call $A(W)$ the behavior instantiated by the agent, after the learning phase with the world $W$.

$\text{loop}(A(W), W) = \text{constant}$

The “physical” behavior remains the same

$A(Wh) = h^{-1}A(W)$
(ignore the observations nuisances)

$A(gW) = A(W)g^{-1}$
(compensate the commands nuisances)
• The analysis is mostly symmetric for the **sensing** part and the **motor** part.

- **Sensor data**
  - high/dimensional
  - easy to visualize

- **Motor data**
  - low dimensional
  - hard to visualize
• Usually, reward-based methods assume that we can see the relevant state \( x \) along with the reward \( R \).

**Interesting questions**

• How to build skills independently of the reward? “what I can do” vs. “what I should do”

• How to interpret raw sensory data?
“Pontifical” features

- Pontifical features are a systematic way to find the “canonical” version of a system. Unfortunately, very hard to find.

\[ \phi = 0 \]

\[ X \]

\[ G \text{ orbits} \]

\[ \text{Can}^{\phi}(x) \]

\[ x \]
• Current research:
  • Extension to “weak” and “mild” features.
  • Compositional properties of these operators.

Strong features

"Mild" features

"Weak" features
“Intrinsic definition” of sensors/actuators faults independently of a nominal model.
Actuators equivalent

Start of log 20100616_000059 (indoor)

(a) Disagreement between actual and predicted values of commands

Legend: ■ ■ actual commands

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

Estimated scale is not correct.

Delayed effects

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

slipping

synchronization issues

commands delay

Start of log 20100616_000059 (outdoor)

(d) Disagreement between actual and predicted values of commands

Legend: ■ ■ estimated commands

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

not detected

missing data

apparent rotation due to bump

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

Estimated scale is lower.

more exciting visual pattern