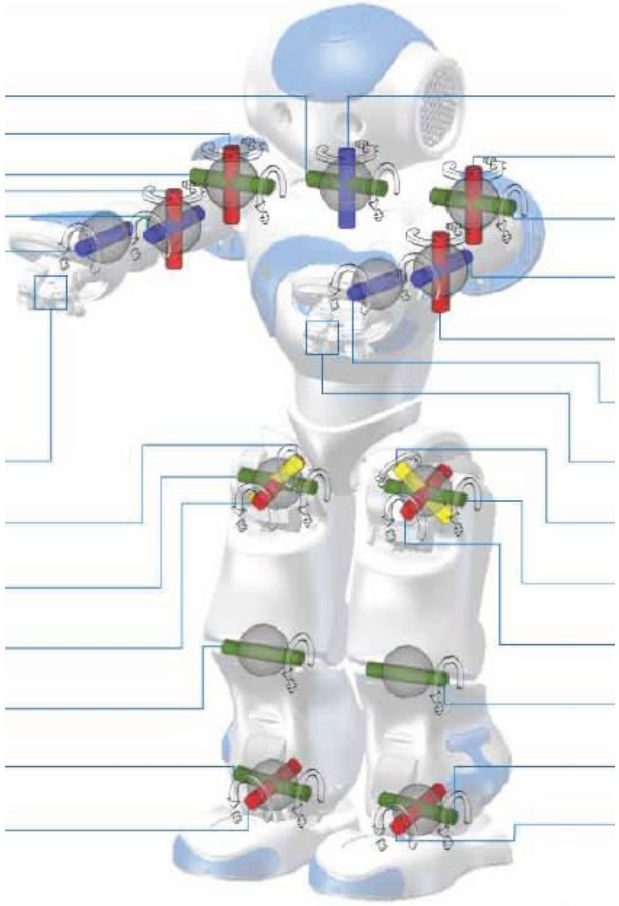
# Structure and Synthesis of Robot Motion

## Making Sense of Sensorimotor Systems

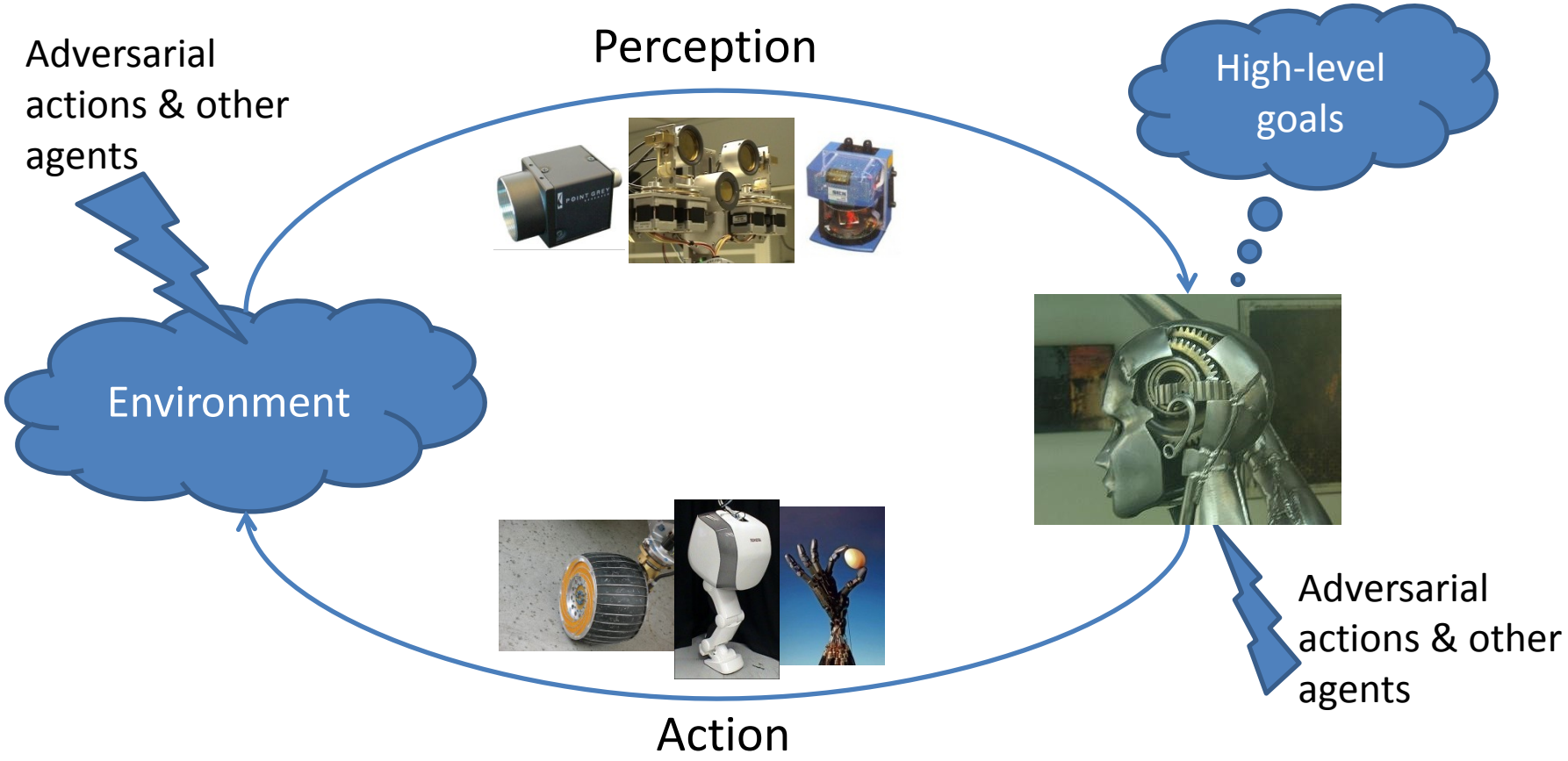
Subramanian Ramamoorthy
School of Informatics

26 January, 2012

# What do you Need to Know about your Robot?



# Recap: What Problem is the Robot Solving?




**<u>Problem</u>: How to generate actions, to achieve high-level goals, using limited perception and incomplete knowledge of environment & adversarial actions?**

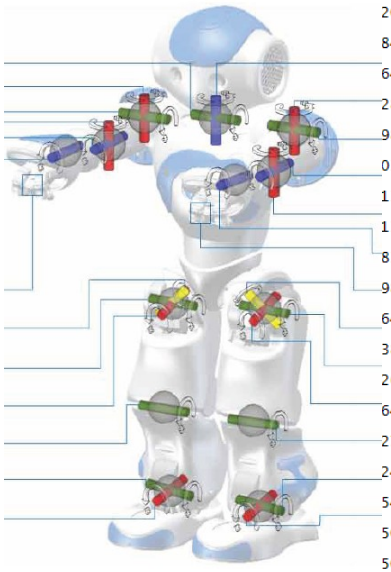
# What does Robot Need to Know?

- Given access to raw data channels for various uninterpreted sensors and motors
- Devise a procedure for learning that will tell you what you need for various tasks (as yet unspecified)
  - What types of models?
  - What types of learning methods?

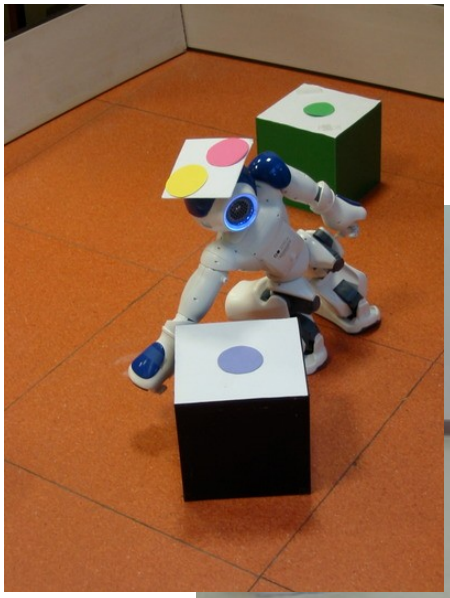
**Discussion: How will you do this?**

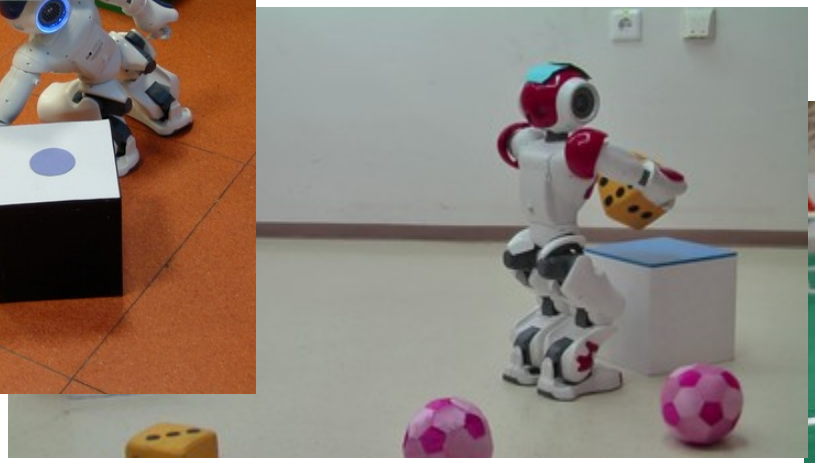
# What are you Learning from?

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1.896007 | 4.323841 | 22.664253 | 4.202899 | 1.213200 | 22.664253 | 1.967881 | 2.245012 | 22.664253 | 2.397054 |
| 1.635462 | 1.698352 | 22.664253 | 2.294771 | 2.627882 | | | | | |
| 1.894624 | 4.323150 | 22.664253 | 4.202899 | 1.213891 | 22.664253 | 1.968572 | 2.245012 | 22.664253 | 2.397745 |
| 1.635462 | 1.698352 | 22.664253 | 2.293389 | 2.627882 | | | | | |
| 1.894624 | 4.323150 | 22.664253 | 4.202899 | 1.211126 | 22.664253 | 1.968572 | 2.245012 | 22.664253 | 2.397745 |
| 1.635462 | 1.699043 | 22.664253 | 2.290625 | 2.627882 | | | | | |
| 1.895315 | 4.323150 | 22.664253 | 4.202899 | 1.211126 | 22.664253 | 1.967881 | 2.245012 | 22.664253 | 2.397745 |
| 1.635462 | 1.699043 | 22.664253 | 2.289934 | 2.630646 | | | | | |
| 1.895315 | 4.323150 | 22.664253 | 4.206354 | 1.210435 | 22.664253 | 1.968572 | 2.245012 | 22.664253 | 2.397745 |
| 1.636153 | 1.699043 | 22.664253 | 2.294080 | 2.630646 | | | | | |
| 1.895315 | 4.323150 | 22.664253 | 4.200825 | 1.211126 | | | 3630 | 22.664253 | 2.397745 |
| 1.630624 | 1.699043 | 22.664253 | 2.318960 | 2.629264 | | | | | |
| 4.375673 | 2.443358 | 22.664253 | 22.664253 | 1.119210 | | | 2080 | 2.341766 | 22.664253 |
| 1.638917 | 1.725996 | 2.295462 | 22.664253 | 1.953368 | | | | | |
| 22.664253 | 22.664253 | 1.250519 | 22.664253 | 22.664253 | | | 8499 | 22.664253 | 2.130981 |
| 22.664253 | 2.322415 | 22.664253 | 22.664253 | 2.000363 | | | | | |
| 22.664253 | 22.664253 | 1.148236 | 4.363925 | 1.913975 | | | 64253 | 2.050813 | 1.548383 |
| 22.664253 | 2.323106 | 2.730856 | 2.043211 | 22.664253 | | | | | |
| 22.664253 | 1.196613 | 4.400553 | 2.316196 | 2.352824 | | | 2802 | 1.486184 | 1.602980 |
| 2.359735 | 22.664253 | 2.068782 | 22.664253 | 22.664253 | | | | | |
| 1.259504 | 22.664253 | 1.965117 | 2.151714 | 22.664253 | | | 9839 | 1.538708 | 22.664253 |
| 3.287191 | 22.664253 | 22.664253 | 22.664253 | 22.664253 | | | | | |
| 22.664253 | 1.991378 | 22.664253 | 2.374248 | 22.664253 | | | 0373 | 22.664253 | 22.664253 |
| 2.467546 | 22.664253 | 22.664253 | 22.664253 | 1.321011 | | | | | |
| 22.664253 | 1.997598 | 2.197326 | 22.664253 | 2.168300 | | | 1388 | 22.664253 | 2.250541 |
| 2.450960 | 22.664253 | 2.661054 | 22.664253 | 1.303734 | | | | | |
| 22.664253 | 1.997598 | 2.194562 | 22.664253 | 2.168300 | | | 1388 | 22.664253 | 2.250541 |
| 2.450960 | 22.664253 | 2.660363 | 22.664253 | 1.303734 | | | | | |
| 22.664253 | 1.996216 | 22.664253 | 22.664253 | 2.171064 | | | 8300 | 22.664253 | 2.252614 |
| 2.457871 | 22.664253 | 2.656908 | 22.664253 | 1.306498 | | | | | |
| 1.244990 | 1.976174 | 2.004509 | 2.356970 | 22.664253 | | | 9881 | 1.555294 | 22.664253 |
| 2.756426 | 22.664253 | 4.558124 | 22.664253 | 22.664253 | | | | | |
| 1.193158 | 22.664253 | 1.908446 | 2.112321 | 2.002436 | | | 64253 | 1.617493 | 22.664253 |
| 3.316217 | 2.086059 | 4.537391 | 3.118563 | 22.664253 | | | | | |
| 22.664253 | 1.164823 | 1.886331 | 1.891860 | 2.328635 | | | 3818 | 1.526268 | 22.664253 |
| 2.411567 | 2.788908 | 2.090206 | 22.664253 | 22.664253 | | | | | |
| 22.664253 | 1.101242 | 4.308637 | 1.858687 | 2.087441 | | | 2645 | 1.585703 | 1.697661 |
| 2.382541 | 2.786835 | 2.070855 | 4.485559 | 22.664253 | | | | | |
| 22.664253 | 22.664253 | 1.065304 | 1.849012 | 2.227735 | | | 64253 | 1.788195 | 1.653431 |
| 2.406730 | 22.664253 | 2.050813 | 4.466898 | 22.664253 | | | | | |
| 22.664253 | 4.712930 | 1.023147 | 3.835233 | 2.220824 | | | 2740 | 2.190415 | 1.671399 |
| 22.664253 | 22.664253 | 2.786143 | 2.019023 | 22.664253 | | | | | |
| 22.664253 | 22.664253 | 1.087420 | 22.664253 | 3.797223 | | | 2499 | 22.664253 | 1.740509 |
| 22.664253 | 2.428845 | 22.664253 | 1.991378 | 4.363234 | | | | | |
| 22.664253 | 22.664253 | 22.664253 | 0.993430 | 1.832425 | | | 5410 | 22.664253 | 2.242248 |
| 22.664253 | 2.428845 | 22.664253 | 22.664253 | 2.000363 | | | | | |
| 22.664253 | 22.664253 | 4.680449 | 0.993430 | 22.664253 | | | 5043 | 2.344531 | 2.220133 |
| 1.815839 | 22.664253 | 22.664253 | 22.664253 | 2.001054 | | | | | |
| 22.664253 | 22.664253 | 4.681140 | 0.993430 | 22.664253 | | | 5043 | 2.344531 | 2.217368 |
| 1.815839 | 22.664253 | 22.664253 | 22.664253 | 2.001054 | | | | | |
| 22.664253 | 22.664253 | 4.691506 | 0.990666 | 22.664253 | 1.815839 | 3.524238 | 2.095734 | 2.345222 | 2.221515 |
| 1.815839 | 22.664253 | 22.664253 | 22.664253 | 2.001054 | | | | | |
| 22.664253 | 22.664253 | 4.680449 | 0.993430 | 22.664253 | 1.815839 | 3.524238 | 2.095734 | 2.344531 | 2.224279 |
| 1.815839 | 22.664253 | 22.664253 | 22.664253 | 2.001054 | | | | | |
| 22.664253 | 22.664253 | 4.676993 | 0.991357 | 22.664253 | 1.815839 | 3.524238 | 2.095734 | 2.345222 | 2.220824 |
| 1.815839 | 22.664253 | 22.664253 | 22.664253 | 2.001054 | | | | | |
| 22.664253 | 22.664253 | 4.682522 | 0.995503 | 22.664253 | 1.814457 | 2.156551 | 2.094352 | 2.339693 | 2.217368 |
| 1.809619 | 22.664253 | 2.470311 | 22.664253 | 2.001745 | | | | | |
| 4.424050 | 22.664253 | 22.664253 | 1.110226 | 22.664253 | 3.797223 | 1.822059 | 2.347986 | 2.070164 | 22.664253 |
| 1.807546 | 22.664253 | 22.664253 | 22.664253 | 22.664253 | | | | | |
| 2.054268 | 4.432344 | 22.664253 | 22.664253 | 22.664253 | 1.003797 | 1.808237 | 1.823441 | 2.343148 | 2.301682 |
| 2.142729 | 1.689368 | 22.664253 | 2.464782 | 22.664253 | | | | | |



# Consider the Various Tasks





**What does the robot need to know?**
**How to get it?**

# Some Requirements

- Models of motion
  - Own dynamics
  - Object dynamics
  - Other agents
- Models of environment
  - Space & how I move in space
  - Other navigation considerations
- Models of self
  - What is the connection between my sensors and actuators?
  - What do the sensorimotor channels even mean?
  - How to ground all of the above at this low level?

# Linear Time Invariant (LTI) Systems

- Consider the simple spring-mass-damper system:
- The force applied by the spring is $F_s = -kz(t)$
- Correspondingly, for the damper: $F_d = \gamma \dot{z}(t)$
- The combined equation of motion of the mass becomes:

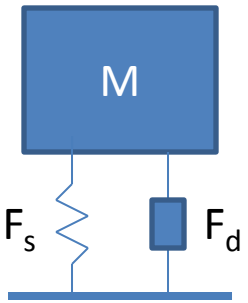
$$x\ddot{z}(t) = -\gamma \dot{z}(t) - kz(t)$$

- One could also express this in state space form:

$$x(t) = [x_1(t), x_2(t)]' = [z(t), \dot{z}(t)]'$$

$$\dot{x}(t) = \begin{pmatrix} \dot{z}(t) \\ \ddot{z}(t) \end{pmatrix} = \begin{pmatrix} x_2(t) \\ -\frac{1}{m}(\gamma x_2(t)) + kx_1(t) \end{pmatrix}$$

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & \frac{\gamma}{m} \end{pmatrix} x(t) = Ax(t)$$

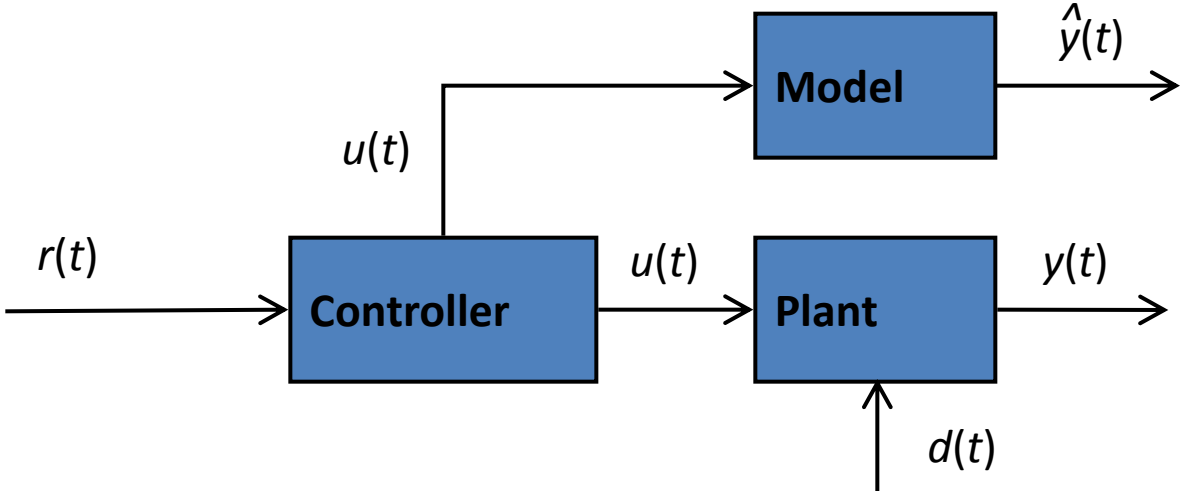


# How to Get Models?

- One may not always want to derive analytical models, one often obtains them from experiment
  - **System Identification**
- By observing inputs and outputs of a system, one tries to fit a suitable model that captures the dynamics
- The model may then be used as a basis for controller synthesis, planning, etc.
- In robotics, you need to do this at *many* different levels
  - Low level control of sensors and actuators
  - “Maps” of the world and objects
  - Models of people and other agents

# Low-level System Identification

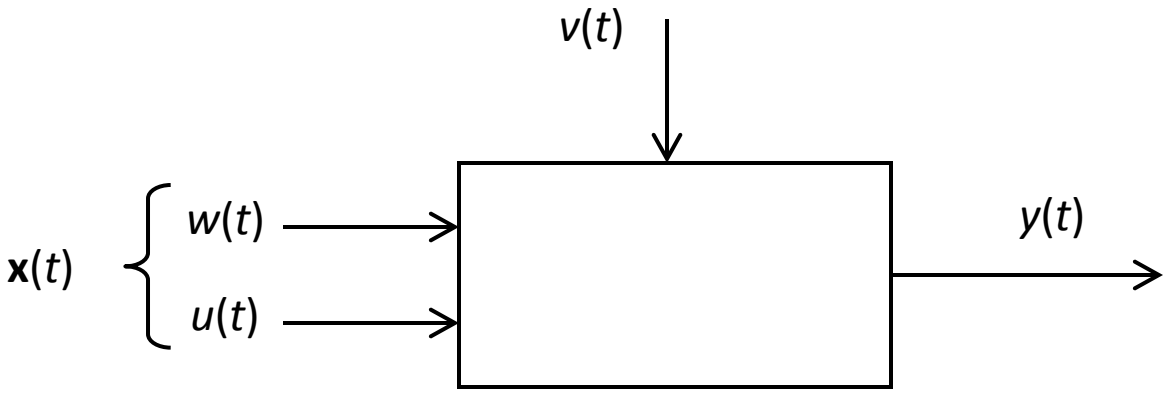
- In order to design a controller, you **must** have a **model**




- **Model** - system that **transforms** an **input signal** into **output signal**.
- Often, may be represented by **differential** or **difference equations**.

- A model is used, either explicitly or implicitly, within control design

# Signals & Systems Definitions




- Typically, a **system** receives an **input signal**, $x(t)$ (generally a vector) and transforms it into the **output signal**, $y(t)$. In modelling and control, this is further broken down:
- $u(t)$ is an input signal that can be manipulated (**control signal**)
- $w(t)$ is an input signal that can be measured (**measurable disturbance**)
- $v(t)$ is an input signal that cannot be measured (**unmeasurable disturbance**)
- $y(t)$ is the **output signal**
- Typically, the system may have hidden **states** $x(t)$

# Example: Solar-Heated House (Ljung)

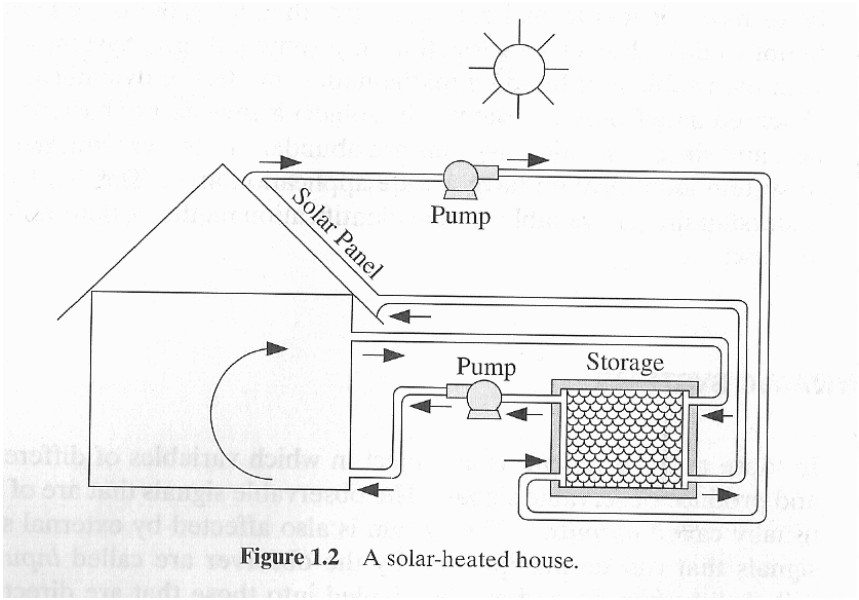
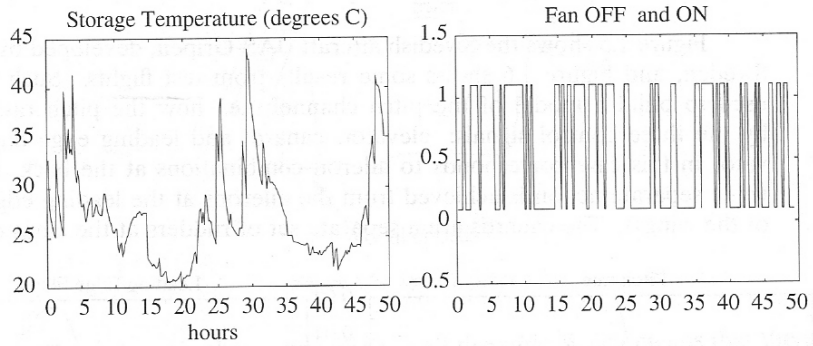
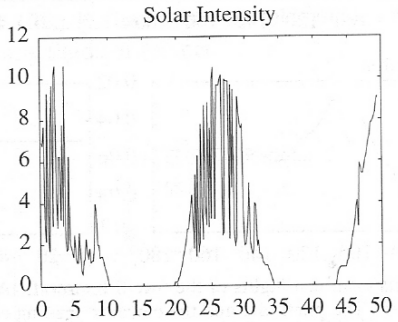



**Figure 1.2** A solar-heated house.




(a) Storage temperature

(a) Pump velocity




- The sun heats the air in the solar panels
- The air is pumped into a heat storage (box filled with pebbles)
- The stored energy can be later transferred to the house
- We're interested in how solar radiation, $w(t)$, and pump velocity, $u(t)$, affect heat storage temperature, $y(t)$.

# Example: Military Aircraft (Ljung)

- Aim is to construct a mathematical model of dynamic behaviour to develop simulators, synthesis of autopilots and analysis of its properties
- Data below used to build a model of pitch channel, i.e. how pitch rate, $y(t)$, is affected by three separate control signals: elevator (aileron combinations at the back of wings), canard (separate set of rudders at front of wings) and leading flap edge

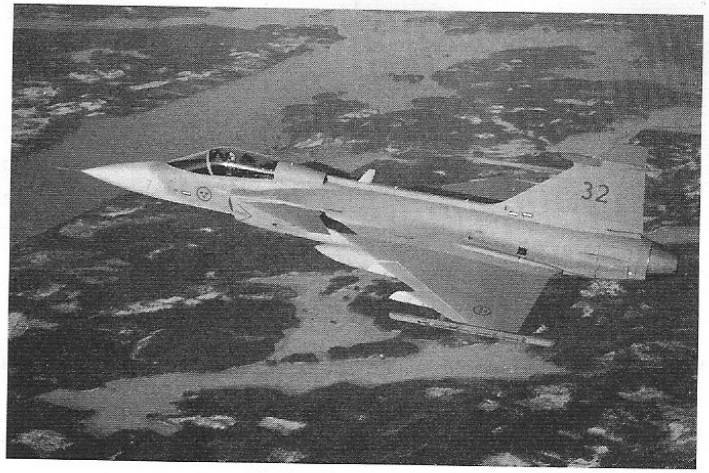


**Figure 1.5** The Swedish fighter aircraft JAS-Gripen.

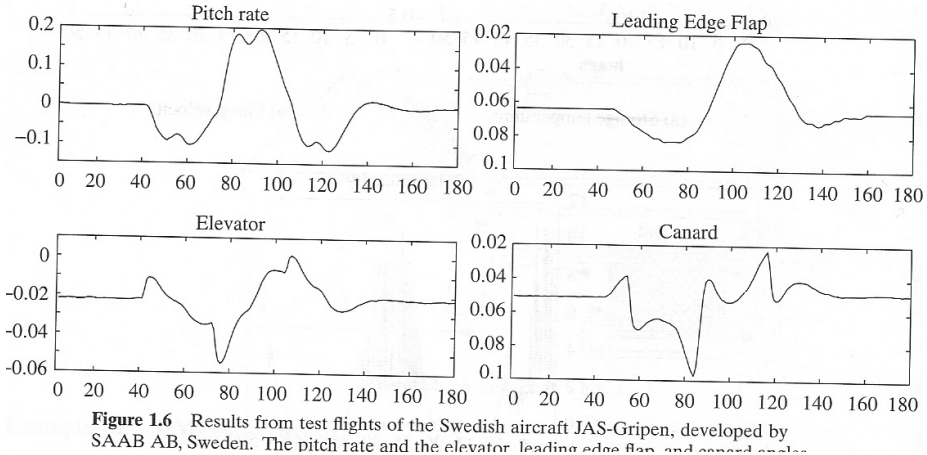



**Figure 1.6** Results from test flights of the Swedish aircraft JAS-Gripen, developed by SAAB AB, Sweden. The pitch rate and the elevator, leading edge flap, and canard angles

# Identifying a System

- Models of car behaviour (acceleration/steering) are built from experience/observational data
- Generally, there may exist some prior knowledge (often formed from earlier observational data) that can be used with the existing data to build a model. This can be combined in several ways:
- Use past experience to express the equations (ODEs/difference equations) of the system/sub-systems and use observed data to estimate the parameters
- Use past experience to specify prior distributions over parameters
- The term **modelling** generally refers to the case when substantial **prior knowledge** exists, the term **system identification** refers to the case when the process is largely based on **observed input-output data**.

# Auto-Regressive with eXogenous inputs Model

- Consider this ARX model with no disturbances

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_m u(t-m)$$

- This can be used for prediction using:

$$y(t) = -a_1 y(t-1) - \cdots - a_n y(t-n) + b_1 u(t-1) + \cdots + b_m u(t-m)$$

- and introducing the vectors:

$$\boldsymbol{\theta} = \begin{bmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_m \end{bmatrix}^T$$

$$\mathbf{x}(t) = \begin{bmatrix} -y(t-1) & \cdots & -y(t-n) & u(t-1) & \cdots & u(t-m) \end{bmatrix}^T$$

- the model can be written as $y(t) = \mathbf{x}^T(t)\boldsymbol{\theta}$
- Note, that the prediction is a function of the estimated parameters which is sometimes written as

$$\hat{y}(t \mid \boldsymbol{\theta}) = \mathbf{x}^T(t)\boldsymbol{\theta}$$

# Least Squares Estimation

- By manipulating the control signal $u(t)$ over a period of time $1 \le t \le N$, we can collect the data set: $Z^N = \{u(1), y(1), \ldots, u(N), y(N)\}$
- Lets assume that the data is generated by: $y(t) = \mathbf{x}^{'}(t)\boldsymbol{\theta} + \ \ \ , \sigma \ )$

- Where $\theta$ is the "true" parameter vector and $\aleph(0, \sigma^2)$ generates zero mean, normally distributed **measurement noise** with standard deviation $\sigma$.
- We want to find the estimated parameter vector, $\hat{\boldsymbol{\theta}}$, that "best fits" this data.

- Note that because of the random noise, we can't fit the data exactly, but we can minimise the prediction errors squared using $\hat{\boldsymbol{\theta}} = \ \mathbf{X}^{'}\mathbf{X})^{-} \ \mathbf{X}^{'}\mathbf{y}$

  Here, **X** is the matrix formed from input vectors (one row per observation, one column per input/parameter) and **y** is the measured output vector (one row per observation

# Example: First Order ARX model

- Consider the simple, first order, linear difference equation, ARX model:

$$y(t) + ay(t - \ ) = bu(t - \ )$$

where 10 data points $Z^{10}$ = {$u$(1),$y$(1), …,$u$(10),$y$(10)} are collected.

- This produces the (9*2) input matrix and (9*1) output vector:

$$\mathbf{X} = \begin{bmatrix} y(1) & u(1) \\ y(2) & u(2) \\ \vdots & \vdots \\ y(9) & u(9) \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y(2) \\ y(3) \\ \vdots \\ y(10) \end{bmatrix}$$

- Therefore the parameters $\theta=[a\ b]^T$ can be estimated from

$$\hat{\boldsymbol{\theta}} = \ \mathbf{X}'\mathbf{X})^{-}\ \mathbf{X}'\mathbf{y}$$

In Matlab: `thetaHat = inv(X'*X)*X'*y`

# Model Quality and Experimental Design

- The variance/covariance matrix to be inverted

$$\mathbf{X}'\mathbf{X} = \begin{bmatrix} \sum_t x_1^2(t) & \sum_t x_1(t)x_2(t) & \cdots & \sum_t x_1(t)x_n(t) \\ \sum_t x_2(t)x_1(t) & \sum_t x_2^2(t) & \cdots & \sum_t x_2(t)x_n(t) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_t x_n(t)x_1(t) & \sum_t x_n(t)x_2(t) & \cdots & \sum_t x_n^2(t) \end{bmatrix}$$

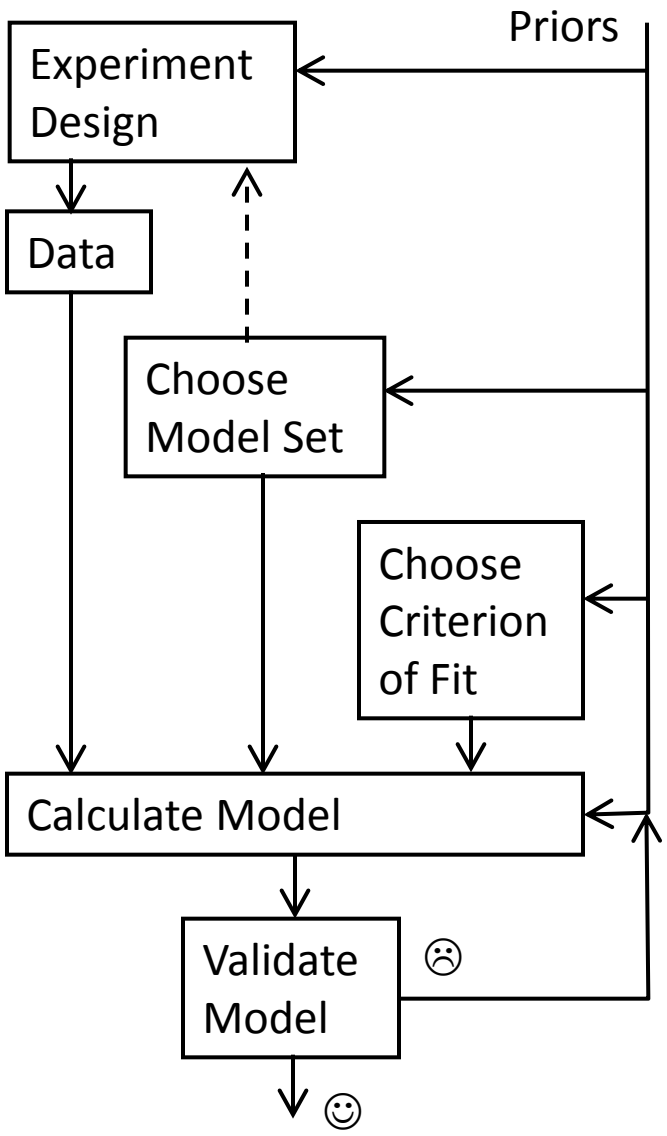
- Strongly determines the quality of the parameter estimates. This in turn is determined by the distribution of the measured input.
- Control *signal should be chosen* to make the matrix as well-conditioned as possible (similar eigenvalues)
- Number of training data & sampling time both affect the accuracy and condition of the matrix
- **Experimental design:** choose experiments to optimally estimate model parameters

# System Identification Process

- In building a model, the designer has control over three parts of the process

1. Generating the **data set** $Z^N$
2. Selecting a (set of) **model structure** (ARX for instance)
3. Selecting the **criteria** (least squares for instance), used to specify the optimal parameter estimates

- A very popular approach involves (recursive) parameter estimation



# Some Questions

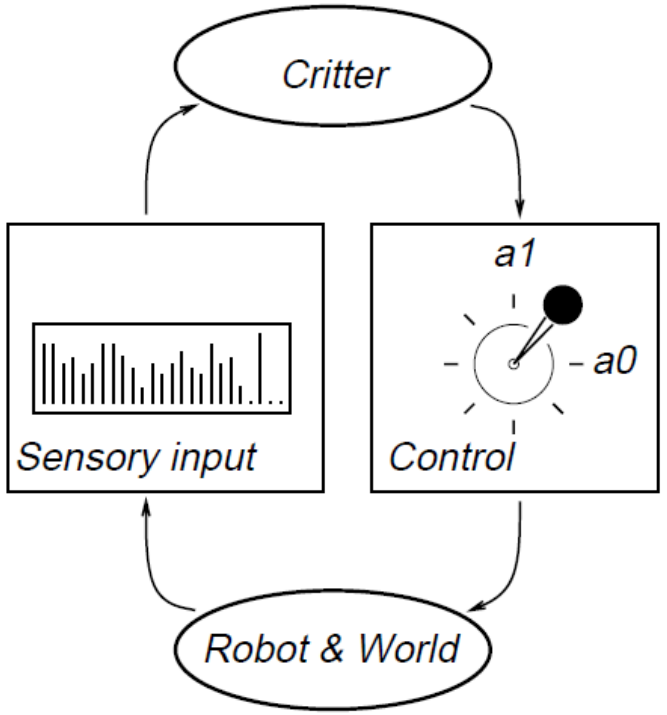
Who makes all the decisions? How to deploy that aspect?

- What variables should I be using for the model?
  - Egocentric/allocentric frames, transformations, etc.
- Once we have such a model, we can define control tasks in terms of this – where do the specifications come from?

- What happens at the higher levels (maps, objects) and how do we tie this to that?

# A Bootstrap Learning Framework

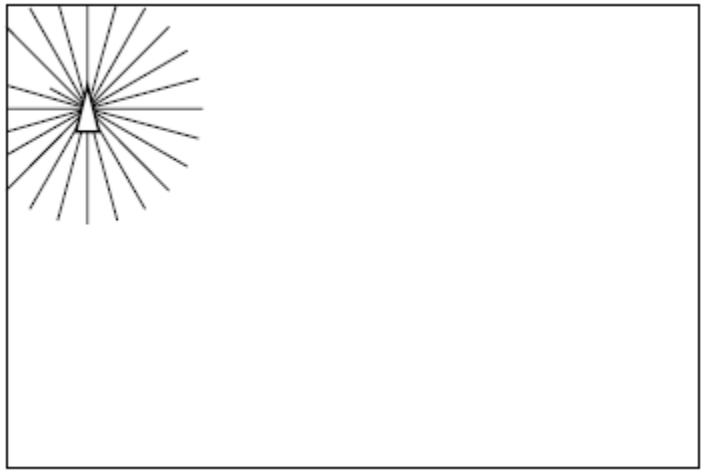
- Learn models of robot and environment with no initial knowledge of what sensors and actuators are doing
- Many learning methods begin this way, e.g., RL, but the goal here is to construct a representation incrementally and continually as well




D. Pierce, B.J. Kuipers, Map learning with un-interpreted sensors and effectors, *Artificial Intelligence* 91:169-227, 1997.

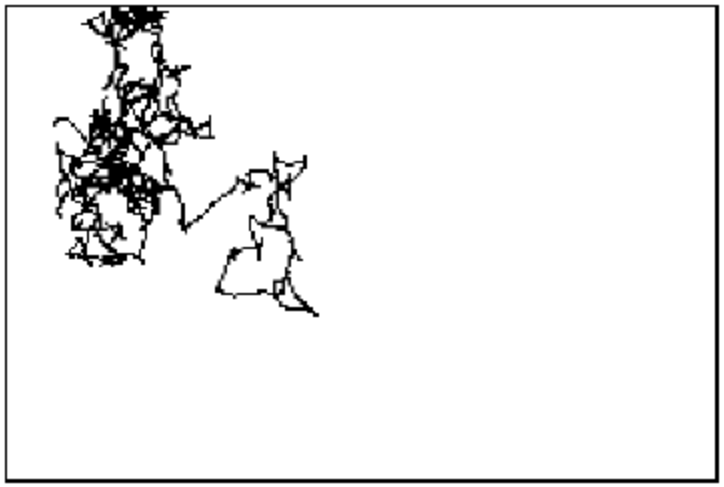
# Simple Scenario

- Robot critter has a set of distance sensors (range) – one of which is defective – but it doesn't know that yet
- Other sensors: battery power, digital compass
- It has a track-style motor apparatus – turn by differentially actuating its wheels



# What do you learn from?

Randomized actions (hold a randomly chosen action for 10 time steps), repeatedly applied



**How does environment appear in the data?**

# A Simple but Complete Procedure

**Problem.**

*Given*: A robot with an uninterpreted, *almost-everywhere approximately linear* sensorimotor apparatus in a *continuous*, *static* environment.

*Learn*: Descriptions of the structure of the robot's sensorimotor apparatus and environment and an abstract interface to the robot suitable for *prediction* and *navigation*.

**Solution.**

*Representation*: A hierarchical model. At the bottom of the hierarchy are egocentric models of the robot's sensorimotor apparatus. At the top of the hierarchy is a discrete abstraction of the robot's environment defined by a set of discrete views and actions.

*Method*: A sequence of statistical and generate-and-test methods for learning the objects of the hierarchical model.

# One Step: Go from Raw Channels to Structure of Sensor Array

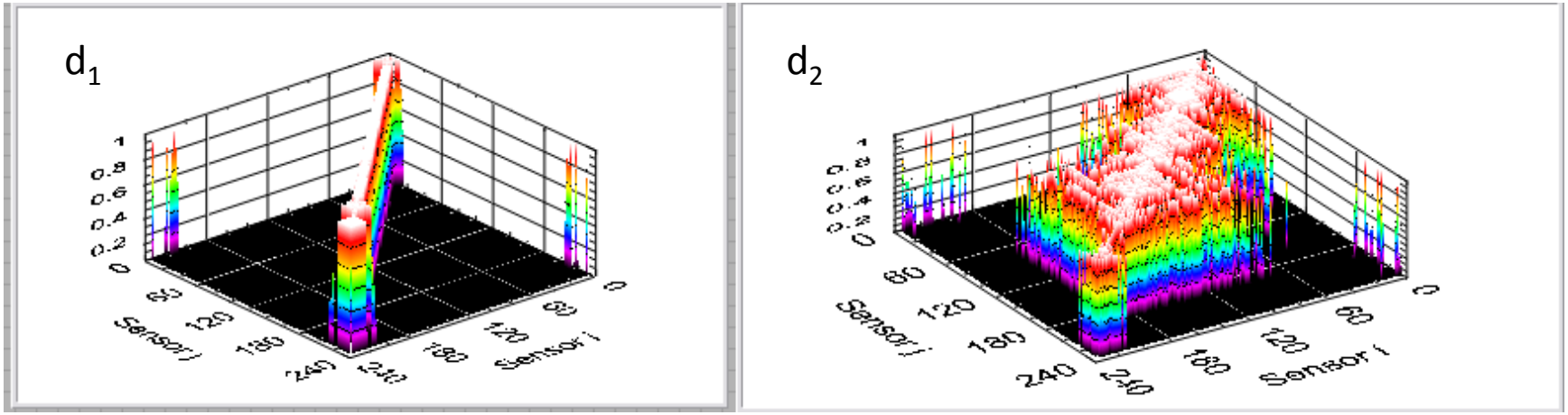
- Sensors may come in groupings: ring of distance sensors, array of photoreceptors, video camera, etc.
- We first want to extract groupings based on two criteria:
  - Sensors that have similar values over time
  - Sensors that have a similar frequency domain behaviour
- Two distance metrics:

$$d_{1,ij}(t) = \frac{1}{t+1} \sum_{\tau=0}^{t} |x_i(\tau) - x_j(\tau)|.$$

$$d_{2,ij} = \frac{1}{2} \sum_{l} |(\boldsymbol{dist}\, x_i)_l - (\boldsymbol{dist}\, x_j)_l|,$$

distribution

# Example Trace




$$i \approx_k j \quad \text{if } d_{k,ij} < \min\{\varepsilon_{k,i}, \varepsilon_{k,j}\}.$$

$$\varepsilon_{k,i} = 2 \min_j \{d_{k,ij}\}.$$

# Extending the Group Notion

We can reason transitively about similarity:

$$i \sim j \quad \text{iff} \quad i \approx j \vee \exists k\colon\ (i \sim k) \wedge (k \sim j).$$

So, a wandering trace might yield something like this as groups:

(0 1 2 22 23) (0 1 2 3 23) (0 1 2 3 4) (1 2 3 4 5) (2 3 4 5 6) (3 4 5 6 7)
(4 5 6 7) (5 6 7 8 9) (7 8 9 10) (7 8 9 10 11) (8 9 10 11 12) (9 10 11 12 13)
(10 11 12 13 14) (11 12 13 14 15) (12 13 14 15 16) (13 14 15 16 17)
(14 15 16 17 18) (15 16 17 18 19) (16 17 18 19) (17 18 19 21) (20)
(19 21 22 23) (0 21 22 23) (0 1 21 22 23) (24) (25) (26) (27) (28).

# After Transitive Closure

(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 21 22 23)

(20) *defective*

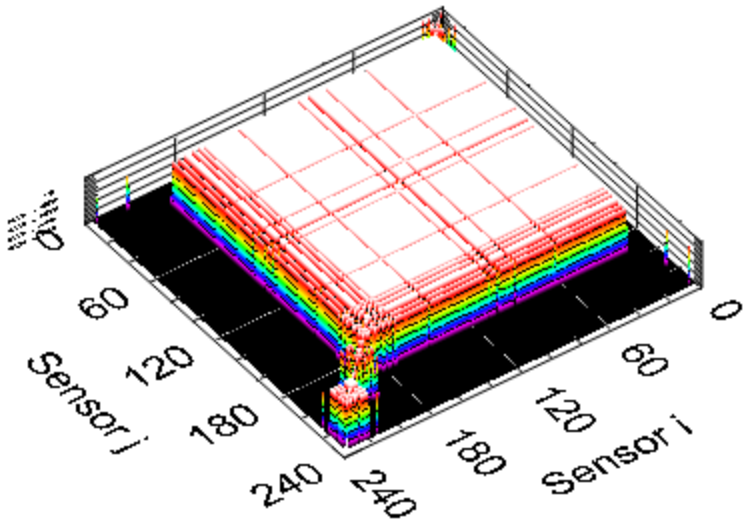
(24) *battery voltage*

(25) *east*

(26) *north*

(27) *west*

(28) *south*



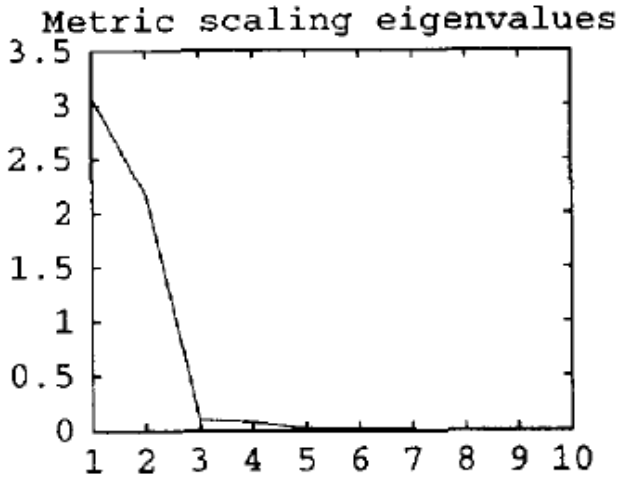
# Getting at Structure of Array

- Task is to find an assignment of positions (in space) to elements that captures the structure of the array as reflected in distance metric $d_1$.
- Distance between positions in image ≈ distance between elements according to $d_1$.

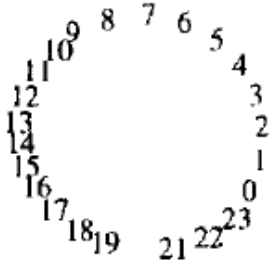
$$\| (\boldsymbol{pos}\ y_i) - (\boldsymbol{pos}\ y_j) \| = d_{1,ij},$$

- This is a constraint satisfaction problem: n sensor elements yield n(n-1)/2 constraints.
- Solve by metric scaling: $$E = \frac{1}{2} \sum_{ij} (\| (\boldsymbol{pos}\ y_i) - (\boldsymbol{pos}\ y_j) \| - d_{ij})^2.$$

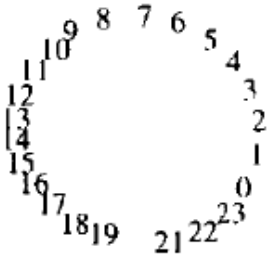
# Structural Model of Distance Array




a

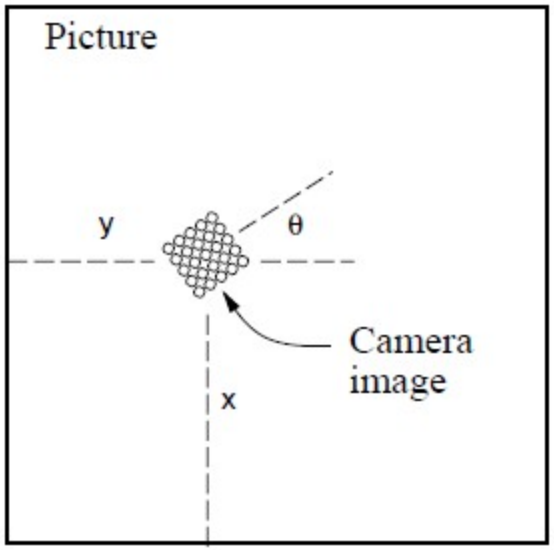


b



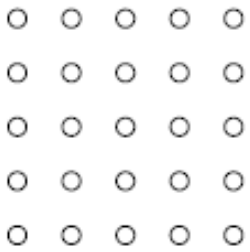
c

# Roving Eye

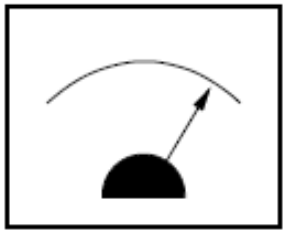



If the array slowly roves over a much larger image, can you recover the structure of this array?

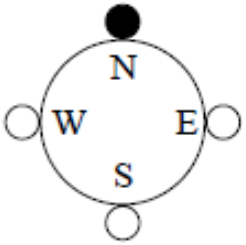
# Roving Eye Robot



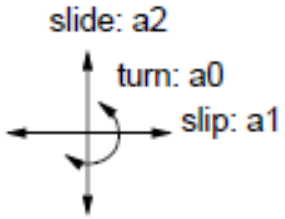
retinal array



battery voltage

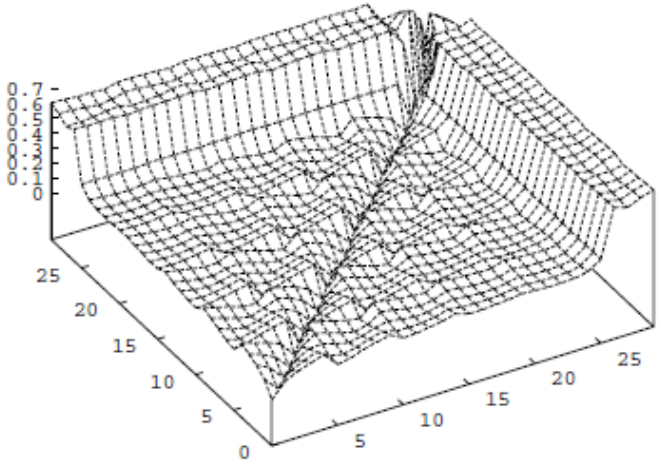


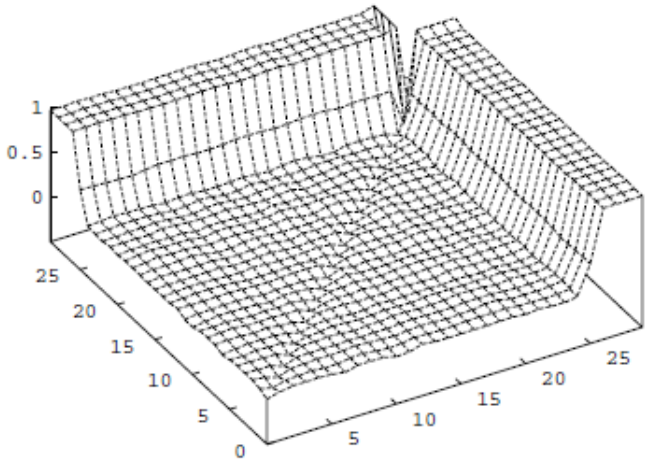

digital compass



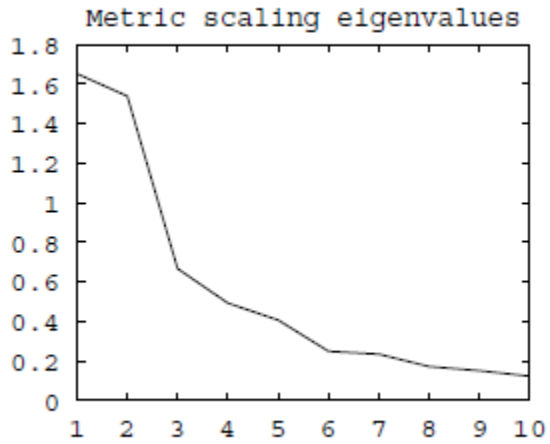

turn-slip-slide

# $d_1$ and $d_2$ for Roving Eye (after 5 mins)

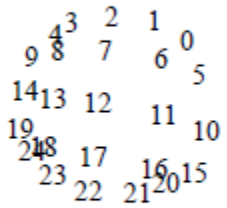


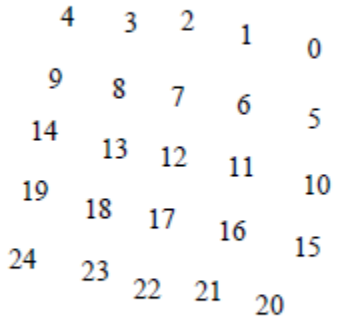
# Metric Scaling Procedure




a



b



c

# Where are we going with this?