Multiagent Systems (SMA-UPC)

5. Situated Agents (Robots)
Part 1:
Introduction to Robotics.
Vision and uncertainty.

Javier Vázquez-Salceda
SMA-UPC

Mobile Robotics

- *"Robotics is an application area of AI where theoretical solutions have to cope with real problems"*
  - Problems in perception (incomplete, uncertain, noisy)
  - Problems in motion (drift, slippage, motion dynamics, obstacles)
- In Mobile Robotics some of those problems increase
  - Large-scale space (regions of space larger than those observed from a single vantage point
  - Local sensors
  - Need for
    - space representation
    - positional error estimation
    - Object and place recognition
    - Real-time response

Introduction to Robotics terminology

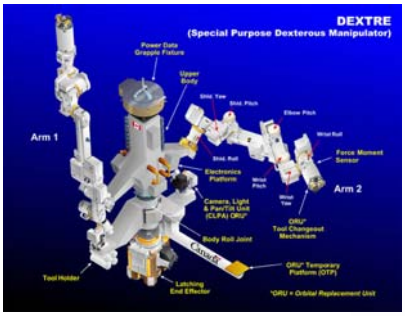
Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

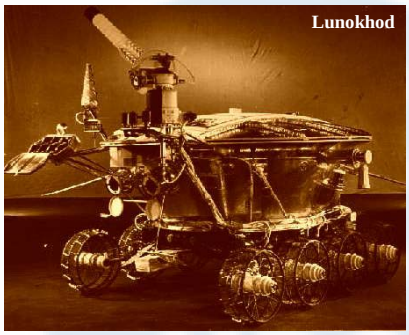
The Syllabus

- Sensors
- Vision
- Actuators
- Motion
- (Forward/Inverse) Kinematics
- Drift
- Localization
- Navigation
- Basic behaviours
- Complex behaviours
- Multi-robot behaviours
- Inertia
- Torque
- Compass
- Joint
- Bumpers
- Landmark
- Geometric Map
- Topological Map

Types of robots (I)

- Static Robots vs Mobile Robots





Types of robots (II)

- Wheeled Robots VS Legged Robots

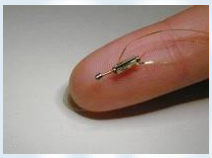
Types of robots (III)

- Robots,

- Microbots,

– Small, cheap robots,
– cheap sensors (no sonar or laser)

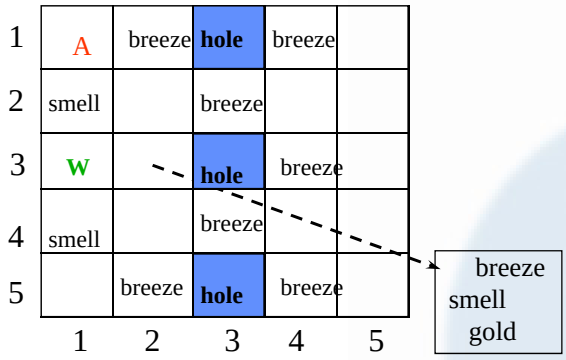
- Nanobots



Types of robots (IV)

- Operational regimes
  - Completely autonomous
  - Semi-autonomous
    - Telerobotic
    - Teleoperated

Back to theory: Wumpus World (I)



Actions:

- Forward ↑,
- 90^0 right ,
- 90^0 left,
- Shot,
- Pick-up,
- Leave,
- Out

Hypothesis 1: discretized world.

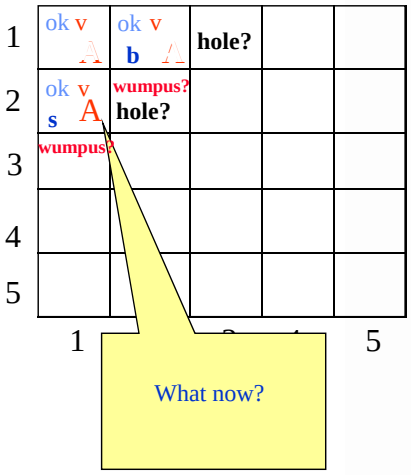
Hypothesis 2: totally deterministic actions.

Back to theory: Wumpus World (II)

- Agent cannot perceive anything in its own position
- In the square where the wumpus is and the 4 adjacent ones (non diagonal) the agent will perceive an smell (s=1).
- In squares adjacent to a hole, the agent will perceive a breeze (b=1)
- The square containing he gold will show a glitter (g=1)
- When the agent smashes against a wall, it will perceive a bump (u=1)
- Perceptions are expressed in lists
  [smell (s), breeze (b), glitter (g), bump(u), cry(c)]

Hypothesis 3: limited perception but perfect, without noise.

Back to theory: Wumpus World (III)

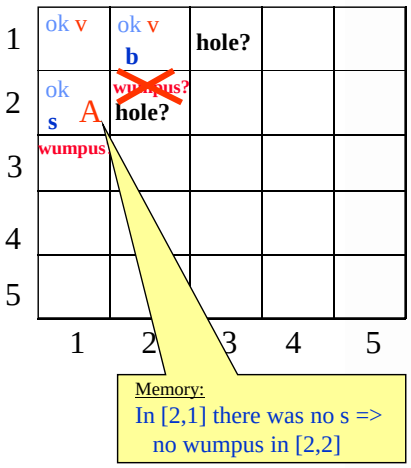


[s,nil,nil,nil,nil]

A = agent
ok= safe
v = visited
s = smell
b = breeze
g = glitter
u = bump
c = cry

5. Situated Agents (Robots)

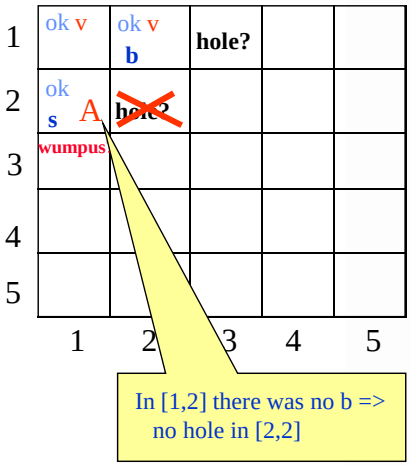
Back to theory: Wumpus World (III)



[s,nil,nil,nil,nil]

A = agent
ok= safe
v = visited
s = smell
b = breeze
g = glitter
u = bump
c = cry

Back to theory: Wumpus World (III)

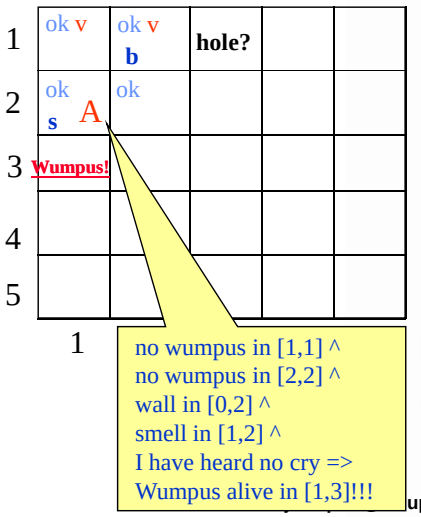


[s,nil,nil,nil,nil]

A = agent
ok= safe
v = visited
s = smell
b = breeze
g = glitter
u = bump
c = cry

5. Situated Agents (Robots)

Back to theory: Wumpus World (III)



[s,nil,nil,nil,nil]

A = agent
ok= safe
v = visited
s = smell
b = breeze
g = glitter
u = bump
c = cry

Back to theory: Situation Calculus

- Perceptual uncertainty problem
  - solution: Situation Calculus
  - Allows the description of the world as a sequence of *situations*, and each one as a snapshot from a world state.
- Problems:
  - Based in hypothesis 1 (discretizable environment)
  - Additional hypothesis: environment won't change without my action
    - → clock is driven by my actions.

Hypothesis 4: static environment (it won't change if I do not change it)

Back to theory: Localization

Heading(Agent, S_0) = 0^o

next-location

$\forall p,l,S\ At(p,l,S) \Rightarrow$next-location$(p,S) =$ direction(l, Heading (p,S))

Adjacency

$\forall l_1,l_2$ adjacent$(l_1,l_2) \Leftrightarrow \exists d \mid l_1 =$direction$(l_2,d)$

Wall

$\forall x,y$ wall$([x,y]) \Leftrightarrow (x=0 \vee x=lim \vee y=0 \vee y=lim)$

Problem: based in perfect knowledge about initial position + hypothesis in totally determinisic actions
→ Hypothesis 5: perfect knowledge about actual location.

Back to theory: Environment properties

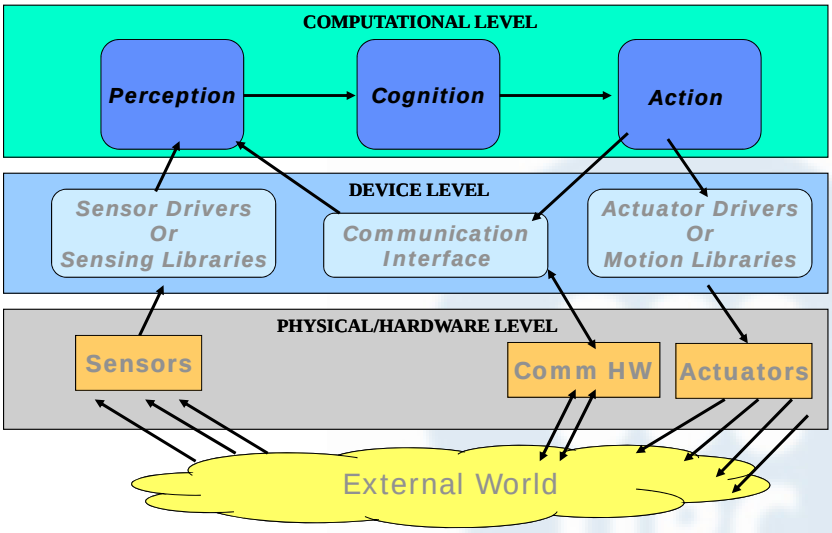
Theory: Wumpus World	Real world: Robots

• Partially accesible
  ▪ Partially observable environment, but perfect perception
• Deterministic
  ▪ Predictable effect for actions
• Sequential (non episodic)
• Static
  ▪ Clock driven by my actions
• Discrete
	• Partially accesible
	  ▪ Partially observable environment and inperfect perception (noise)
	• Stocastic
	  ▪ Unpredictable effect for actions (inertia, drift, slipage)
	• Sequential (non episodic)
	  ▪ Cumulative errors
	• Dynamic
	  ▪ Real time
	• Continuous

Robot architectures

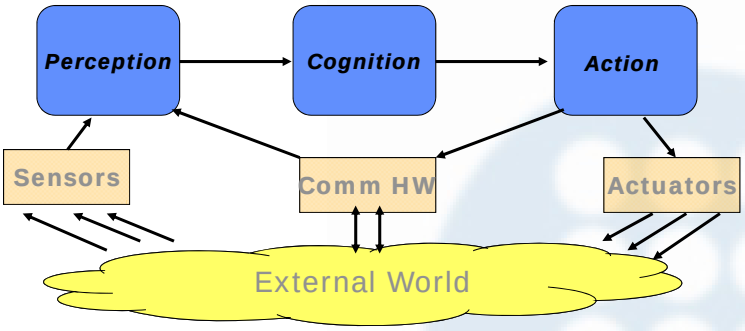
Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

Levels of abstraction



Intelligent Robot (I)

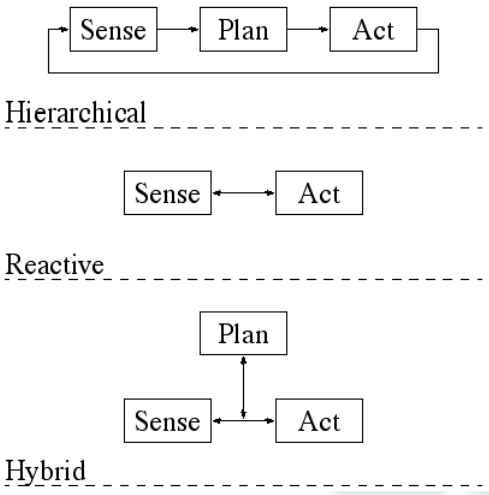
Tasks



Intelligent Robot (II)
Tasks

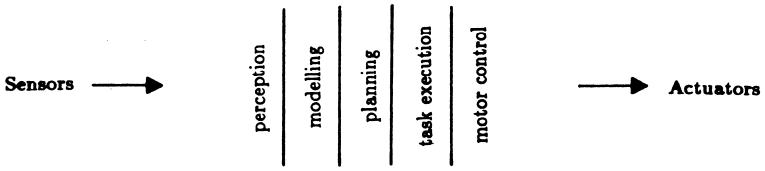
- Perception
  - sensing, modeling of the world
  - Communication (*listening*)
- Cognition
  - behaviors, action selection, planning, learning
  - multi-robot coordination, teamwork
  - response to opponent, multi-agent learning
- Action
  - motion, navigation, obstacle avoidance
  - Communication (*telling*)

Intelligent Robot (III)
Architectural Paradigms



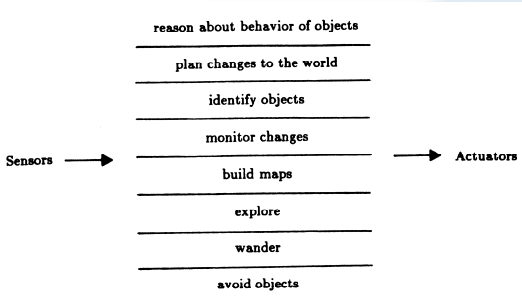
Intelligent Robot (III)
Hierarchical Paradigm

- Used in early times of robotics
- Problem: time of reaction
- Example: Shakey (Standford Univ, 1970)



Intelligent Robot (III)
Reactive Paradigm

- Reactive paradigm organizes the components vertically so that there is a more direct route from sensors to actuators.
- Schematically Brooks (1986) depicts the paradigm as follows:

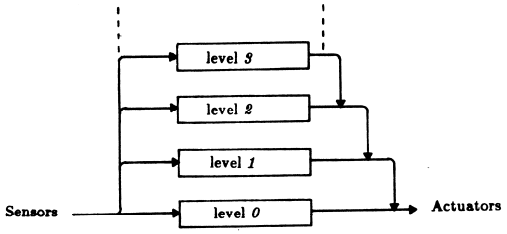


- Problem: conflicting orders to Actuators

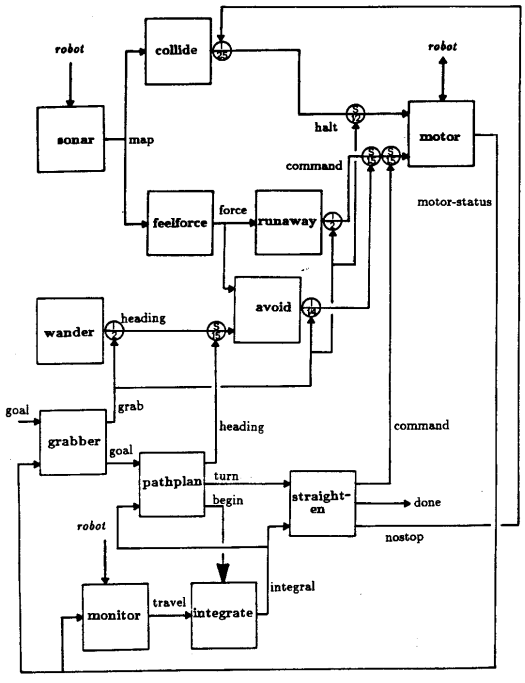
Intelligent Robot (III)

Brooks' Subsumption Architecture

- Components behaviors are divided into layers (modules) with inputs, outputs and a reset.
- Arbitration scheme: a module at a higher level can
  - suppress the input of a module at a lower level thereby preventing the module from seeing a value at its input.
  - inhibit the output of a module at a lower level thereby preventing that output from being propagated to other modules.

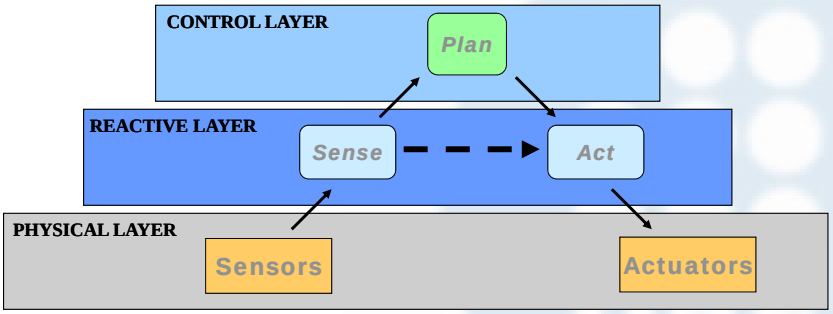


» Problem: complex set-up of modules to avoid low-level reaction problems

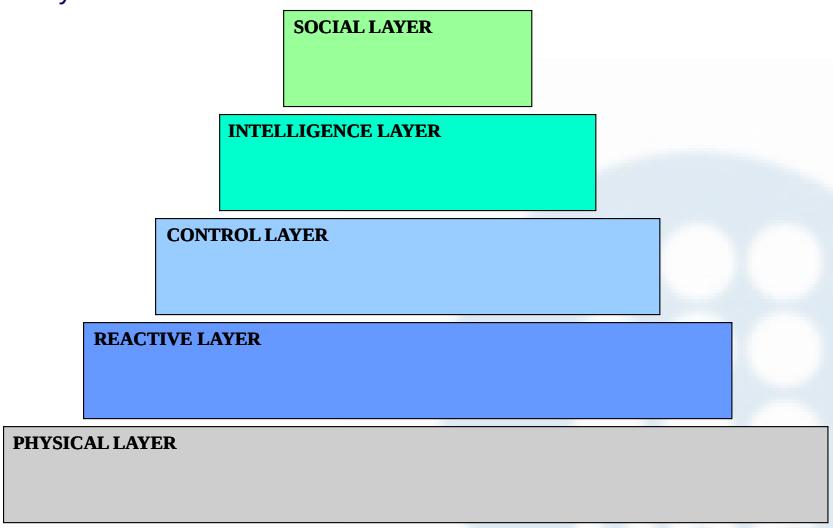


Intelligent Robot (III)
Hybrid Architectures

- Tries to equilibrate deliberation and reactivity
- Usually deliberation UNLESS immediate reaction is needed



Intelligent Robot (III)
Layers



Perception

- **Non-visual sensors**
- **Actuators and feedback**
- **Vision (segmentation, color)**
- **Localization**

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

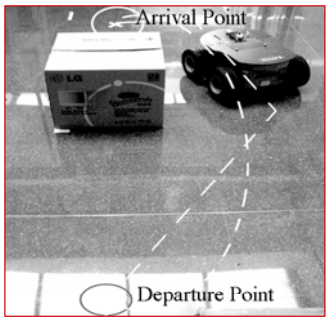
Perception: Non-visual sensors

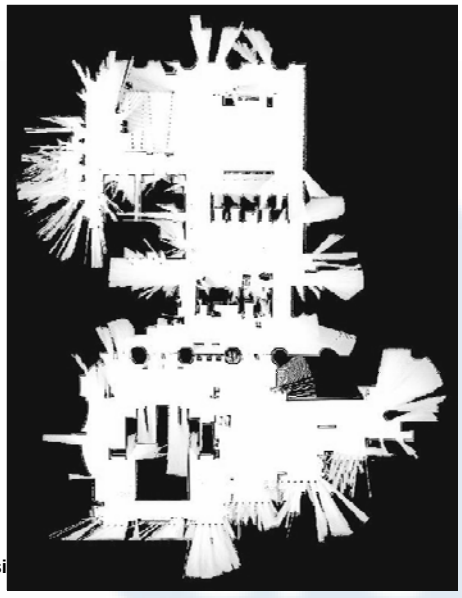
- Bumpers / pressure sensors
- Sonar sensors
- Radar sensors
- Laser sensor
- Compass (brújula)
- Inclinometers
- Odometers
  - wheeled
  - optics

Perception: Actuators and feedback

- Junctions and motors may include sensors
  - Step-by-step motors give an acceptable estimate of how many steps (discretized degrees) they have rotated
  - Servo motors have high accuracy and give good estimate of degrees they have rotated.
  - Some junctions have some sensors outside the servos/motors to estimate their position.
- The position of the junctions/servos/motors can be used as perception to estimate the position of the robot or parts of it (arms/limbs/head).

Example: perception with 7 sonars





Perception: Vision

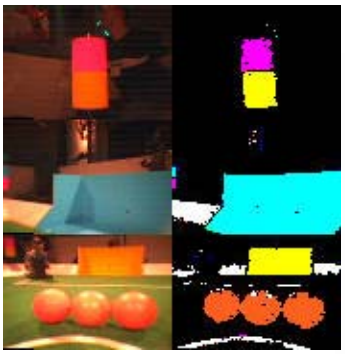
- Vision is a way to relate measurement to scene structure
  - Our human environments are shaped to be navegated by vision
    - E.g., road lines
  - Problem: vision technology is not well-developed
    - Recognition of shapes, forms, ….
  - Solution: in most of cases, we don't need full recognition
    - Use our knowledge of the domain to ease vision
      - E.g.: Green space in a soccer field means free (void) space.
- Two kinds of vision:
  - Passive vision: static cameras
    - Processing of snapshots
  - Active vision: the camera moves.
    - Intricate relation between camera and the environment

Perception: Vision
Active Vision

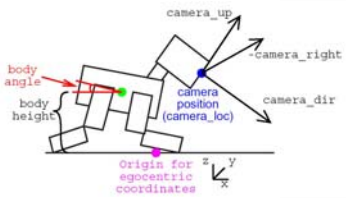
- Important geometric relation between the camera and the enviroment
  - Movement of the camera should produce an (expected) change in the image
- Useful to increase the visual information of an item
  - Move to avoid another object that blocks the vision
  - Move to have another viewpoint of the object, and ease recognition
  - Move to measure distances by comparison of images
    - An improvement: Stereo Vision
- Active Vision is highly sensible to calibration
  - Geometric calibration
  - Color calibration

Perception: Vision

Active Vision



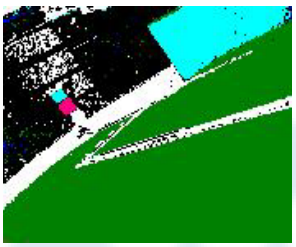
- Color calibration
  - Identify the colors of landmarks and important objects
  - Adaptation to local light condition
  - Saturation of color
    - Colored blobs identified as objects
    - Problem: threshold selection
- Geometric calibration
  - Position of the camera related to the floor
  - At least 3 coordinate systems
    - Egocentric coordinates
    - Camera coordinates
      - Translation matrix counting intermediate joints



Perception: Vision

Image Segmentation

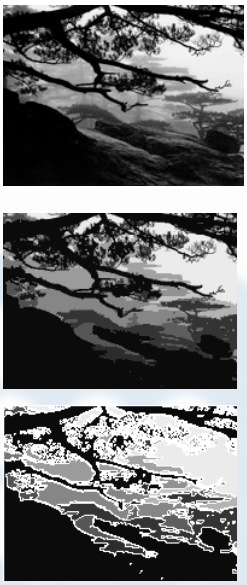
- Sort pixels into classes
- Obstacle:
  - Red robot
  - Blue robot
  - White wall
  - Yellow goal
  - Cyan goal
  - Unknown color
- Freespace:
  - Green field
- Undefined occupancy:
  - Orange ball
  - White line



Perception: Vision

Image Segmentation by region growing

- Start with a single pixel *p* and wish to expand from that *seed pixel* to fill a coherent region.
- Define a similarity measure *S*(*i, j*) such that it produces a high result if pixels *i* and *j* are similar
- Add pixel *q* to neighbouring pixel *p*'s region iff *S*(*p, q*) > *T* for some threshold *T*.
- We can then proceed to the other neighbours of *p* and do likewise, and then those of *q*.
- Problems:
  - highly sensible to the selection of the seed and the Threshold.
  - computationally expensive because the merging process starts from small initial regions (individual points).

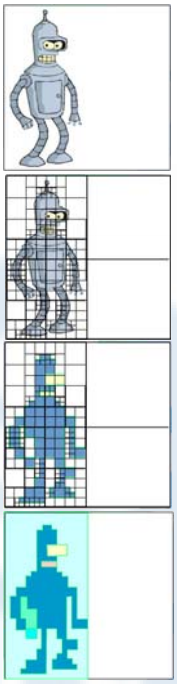


(Example by L. Saad and C. Bordenade)
http://stuff.mit.edu/people/leonide/segmentation/

Perception: Vision

Image Segmentation by Split and Merge

- **Split** the image. Start by considering the entire image as one region.
  - If the entire region is coherent (i.e., if all pixels in the region have sufficient similarity), leave it unmodified.
  - If the region is not sufficiently coherent, split it into four quadrants and recursively apply these steps to each new region.
- The "splitting" phase builds a quadtree
  - several adjacent squares of varying sizes might have similar characteristics.
- **Merge** these squares into larger coherent regions from the bottom up.
  - Since starts with regions (hopefully) larger than single pixels, this method is more efficient.

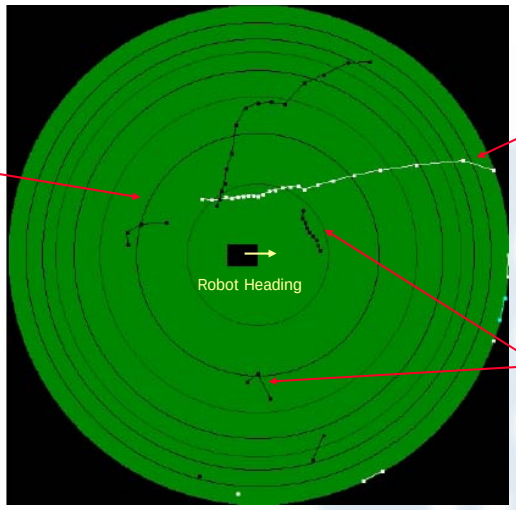


(Example by C. Urdiales)

Perception: Localization

- *Where am I?*
- Given a map, determine the robot's location
  - Landmark locations are known, but the robot's position is not
  - From sensor readings, the robot must be able to infer its most likely position on the field
  - Example : where are the AIBOs on the soccer field?

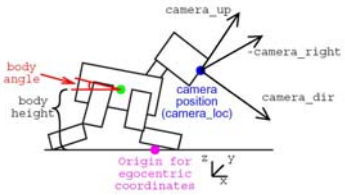
Visual Sonar



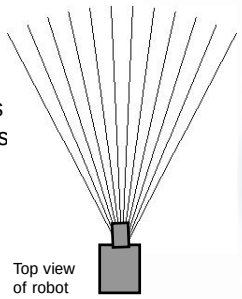
Visual Sonar Algorithm

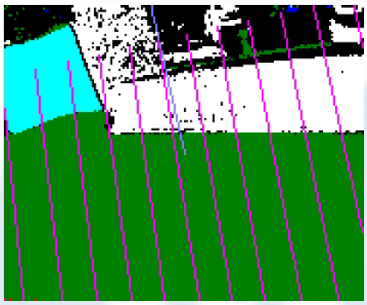
1) Segment image by colors
2) Vertically scan image at fixed increments
3) Identify regions of freespace and obstacles in each scan line
4) Determine relative egocentric (x,y) point for the start of each region
5) Update points
    1) Compensate for egomotion
    2) Compensate for uncertainty
    3) Remove unseen points that are too old

Scanning Image for Objects



Scanlines projected from origin for egocentric coordinates in 5 degree increments

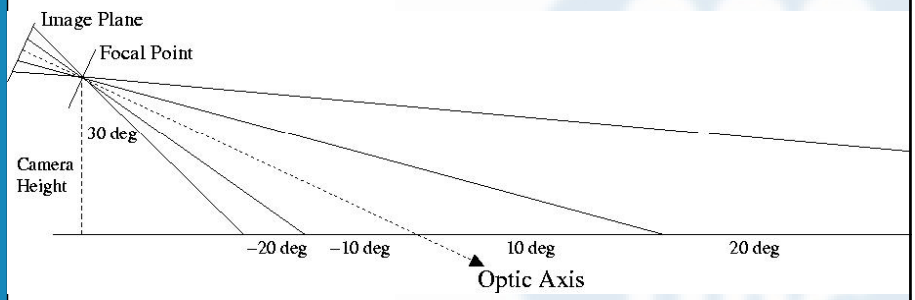




Scanlines projected onto RLE image

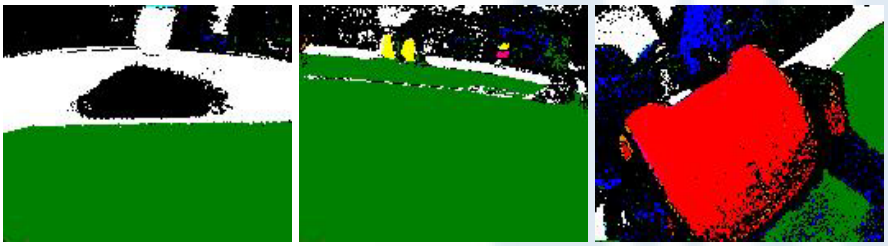
Measuring Distances with the AIBO's Camera

- Assume a common ground plane
- Assume objects are on the ground plane
  - Elevated objects will appear further away
  - Increased distance causes loss of resolution



Identifying Objects in Image

- Along each scanline:
  - Identify continuous line of object colors
  - Filter out noise pixels
  - Identify colors out to 2 meters



Differentiate walls and lines

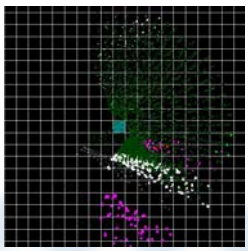
- Filter #1
  - Object is a wall if it is a least 50mm wide
- Filter #2
  - Object is a wall if the number of white pixels in the image is greater than the number of green pixels *after* it in scanline

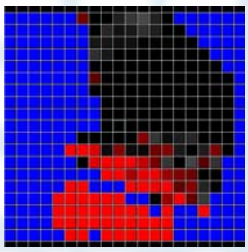
Keeping Maps Current

- Spatial:
  - All points are updated according to the robot's estimated egomotion
  - Position uncertainty will increase due to odometric drift and cumulative errors due to collisions
  - Positions of moving objects will change
- Temporal:
  - Point certainty decreases as age increases
  - Unseen points are "forgotten" after 4 seconds

Interpreting the Data

- Point representations
  - Single points are very noisy
  - Overlaps are hard to interpret
  - Point clusters show trends
- Occupancy grids
  - Probabilistic tessellation of space
  - Each grid cell maintains a probability (likelihood) of occupancy





Calculating Occupancy of Grid Cells

- Consider all of the points found in a grid cell
- If there are any points at all, the grid is marked as being observed
- Obstacles increase likelihood of occupancy
- Freespace decreases likelihood of occupancy
- Contributions are summed and normalized
- If the sum is greater than a threshold (0.3), the cell is considered occupied with an associated confidence

Open Questions

- How easy is it to follow boundaries?
  - Odometric drift will cause misalignments
  - Noise merges obstacle & non-obstacle points
    - Where do you define the boundary?
- How can we do path planning?
  - Local view provides poor global spatial awareness
  - Shape of robot body must be taken into account in order to avoid collisions and leg tangles

Bayesian Filter

- Why should you care?
  - Robot and environmental state estimation is a fundamental problem!
- Nearly all algorithms that exist for spatial reasoning make use of this approach
  - If you're working in mobile robotics, you'll see it over and over!
  - Very important to understand and appreciate
- Efficient state estimator
  - Recursively compute the robot's current state based on the previous state of the robot

What is the robot's state?

Bayesian Filter

- Estimate state *x* from data *d*
  - *What is the probability of the robot being at x?*
- *x* could be robot location, map information, locations of targets, etc…
- *d* could be sensor readings such as range, actions, odometry from encoders, etc…)
- This is a general formalism that does not depend on the particular probability representation
- Bayes filter **recursively** computes the posterior distribution:

$$Bel(x_T) = P(x_T \mid Z_T)$$

Derivation of the Bayesian Filter

Estimation of the robot's state given the data:

$$Bel(x_t) = p(x_t \mid Z_T)$$

The robot's data, Z, is expanded into two types: observations o_i and actions a_i

$$Bel(x_t) = p(x_t \mid o_t, a_{t-1}, o_{t-1}, a_{t-2}, ..., o_0)$$

Invoking the Bayesian theorem

$$Bel(x_t) = \frac{p(o_t \mid x_t, a_{t-1}, ..., o_0)\, p(x_t \mid a_{t-1}, ..., o_0)}{p(o_t \mid a_{t-1}, ..., o_0)}$$

Derivation of the Bayesian Filter

First-order Markov assumption shortens middle term:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) p(x_{t-1} \mid a_{t-1}, ..., o_0) dx_{t-1}$$

Finally, substituting the definition of $Bel(x_{t-1})$:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

The above is the probability distribution that must be estimated from the robot's data

Iterating the Bayesian Filter

- Propagate the motion model:

$$Bel_{\_}(x_t) = \int P(x_t \mid a_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Compute the current state estimate before taking a sensor reading by integrating over all possible previous state estimates and applying the motion model

- Update the sensor model:

$$Bel(x_t) = \eta P(o_t \mid x_t) Bel_{\_}(x_t)$$

Compute the current state estimate by taking a sensor reading and multiplying by the current estimate based on the most recent motion history

Perception: Localization with Uncertainty

Initial state
detects nothing:

Moves and
detects landmark:

Moves and
detects nothing:

Moves and
detects landmark:

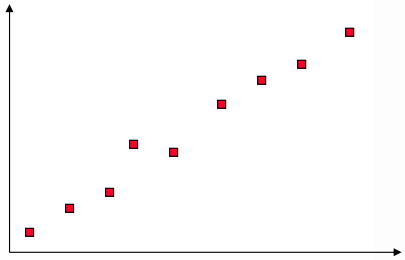
Bayesian Filter : Requirements for Implementation

- Representation for the belief function
- Update equations
- Motion model
- Sensor model
- Initial belief state

Representation of the Belief Function

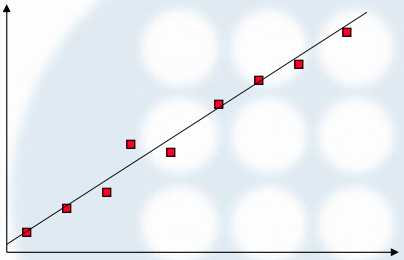
Sample-based representations

e.g. Particle filters



$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots (x_n, y_n)$

Parametric representations



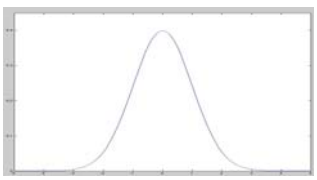
$y = mx + b$

Example of a Parameterized Bayesian Filter : Kalman Filter

Kalman filters (KF) represent posterior belief by a Gaussian (normal) distribution

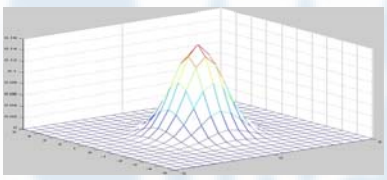
A 1-d Gaussian distribution is given by:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



An n-d Gaussian distribution is given by:

$$P(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$



Kalman Filter : a Bayesian Filter

- Initial belief *Bel(x_0)* is a Gaussian distribution
  - *What do we do for an unknown starting position?*
- State at time *t+1* is a linear function of state at time *t:*

$$x_{t+1} = Fx_t + Bu_t + \varepsilon_{t(action)}$$

- Observations are linear in the state:

$$o_t = Hx_t + \varepsilon_{t(observation)}$$

- Error terms are zero-mean random variables which are normally distributed
- These assumptions guarantee that the posterior belief is Gaussian
  - The Kalman Filter is an efficient algorithm to compute the posterior
  - Normally, an update of this nature would require a matrix inversion (similar to a least squares estimator)
  - The Kalman Filter avoids this computationally complex operation

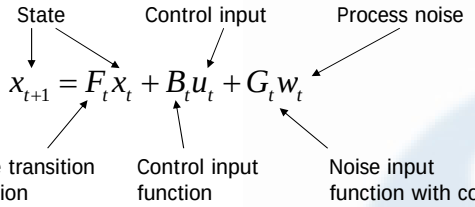
The Kalman Filter

- Motion model is Gaussian…
- Sensor model is Gaussian…
- Each belief function is uniquely characterized by its mean μ and covariance matrix Σ
- Computing the posterior means computing a new mean μ and covariance Σ from old data using actions and sensor readings
- *What are the key limitations?*

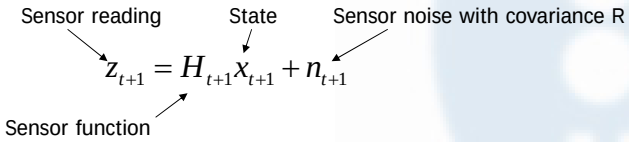
  1) Unimodal distribution
  2) Linear assumptions

The Kalman Filter

Linear discrete time dynamic system (motion model)



Measurement equation (sensor model)



What we know…
What we don't know…

- We know what the control inputs of our process are
  - We know what we've told the system to do and have a model for what the expected output should be if everything works right
- We don't know what the noise in the system truly is
  - We can only estimate what the noise might be and try to put some sort of upper bound on it
- When estimating the state of a system, we try to find a set of values that comes as close to the truth as possible
  - There will always be some mismatch between our estimate of the system and the true state of the system itself. We just try to figure out how much mismatch there is and try to get the best estimate possible

…but what does that mean in English?!?

Propagation (motion model):

$$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$$

- State estimate is updated from system dynamics

$$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$$

- Covariance matrix for the state
→ Uncertainty estimate *GROWS*

Update (sensor model):

$$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$$

- Sensor estimate: expected value of sensor reading

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$

- Compute the difference between expected and "true"

$$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^T + R_{t+1}$$

- Compute covariance matrix of sensor reading

$$K_{t+1} = P_{t+1/t} H_{t+1}^T S_{t+1}^{-1}$$

- Compute the Kalman Gain (how much to correct est.)

$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

- Multiply residual times gain to correct state estimate

$$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t}$$

- Uncertainty estimate SHRINKS

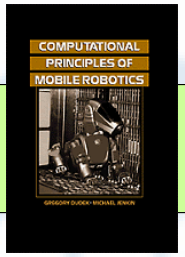
5. Situated Agents (Robots)

References

[1] Russell, S. & Norvig, P. "Artificial Intelligence: A Modern Approach" Prentice-Hall Series in Artificial Intelligence. 1995 ISBN 0-13-103805-2

[2] Recommended book:

Computational Principles of Mobile Robotics
Gregory Dudek, Michael Jenkin
Cambridge University Press 2000



[3] More information on AIBO robots and OPEN-R
http://openr.aibo.com

[4] Robocup league
http://www.robocup.org

These slides are based mainly in [2], [1] and material from M. Veloso and E. Rybski