Research on mobile robots at URJC

Vicente Matellán y José María Cañas

vmo,jmplaza@gsyc.escet.urjc.es

25 de Mayo de 2004

The short history of our group

Environment

- URJC (1998): 4 campus, ($\sim 12{,}000$ students)
- ESCET: 7 studies (ITIG, ITIS, II, IQ, LCM, ITI, IM)
- DIET (2002): ~ 125 teaching members (CCeIA(1++),LSI(~ 50),ATC(~ 35),IT(~ 15),EST(~ 20)
- GSyC: Operating systems, networking, and robotics (1CU, 4TU, 2TEU 12 Ayud & 3 Asoc)
- Robotics Group: 1TU, 1AD, 2Ay, 1 bec (TC), 2 bec (TP, project) & 1 Visiting

The evolution of the group

Academic Year	People	Robots
1999-2000	1 Doct	+20 Lego, +2 Eyebots
2000-2001	1 Doct 1 Ay	+4 Eyebots, +10 Lego
2001-2002	1 Doct 1 Ay	+1 Pioneer
2002-2003	1 Doct 2 Ay	+1 Pioneer + 1 laser
2003-2004	2 Doct 2 Ay 1 Vis 1 bec(TC), 2 bec(TP)	+3 Aibo + 1 laser

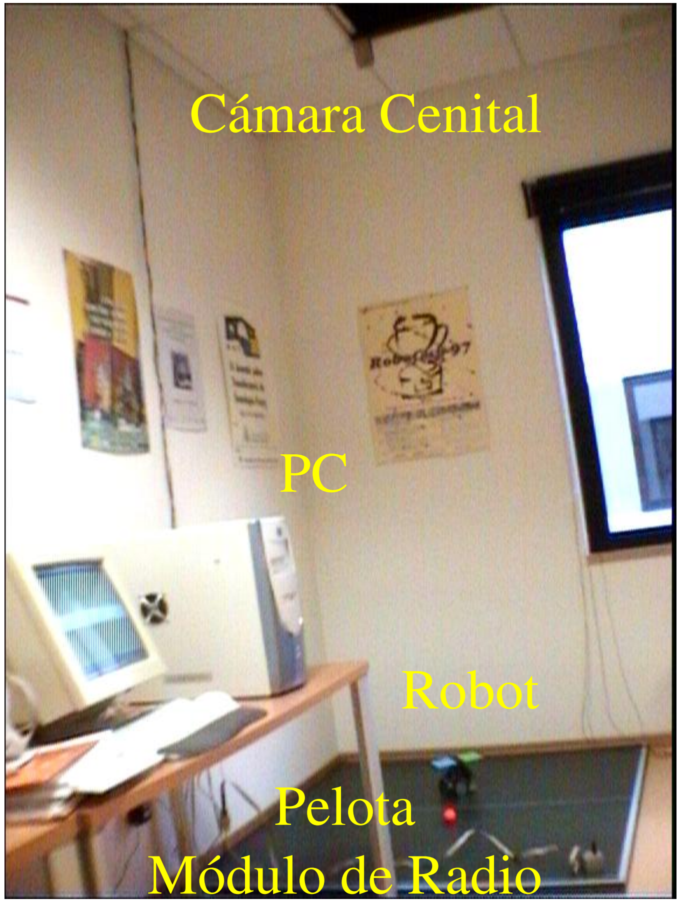
Ourselves

Research activities examples

- Basic behaviors for EyeBot robots (robosoccer)
- PERA: Ad-hoc networks for mobile robots
- Dynamic Schema Hierarchies
- Robot localization using WiFi signal
- Topological navigation in a legged robot
- SiS: Speed Intelligent System

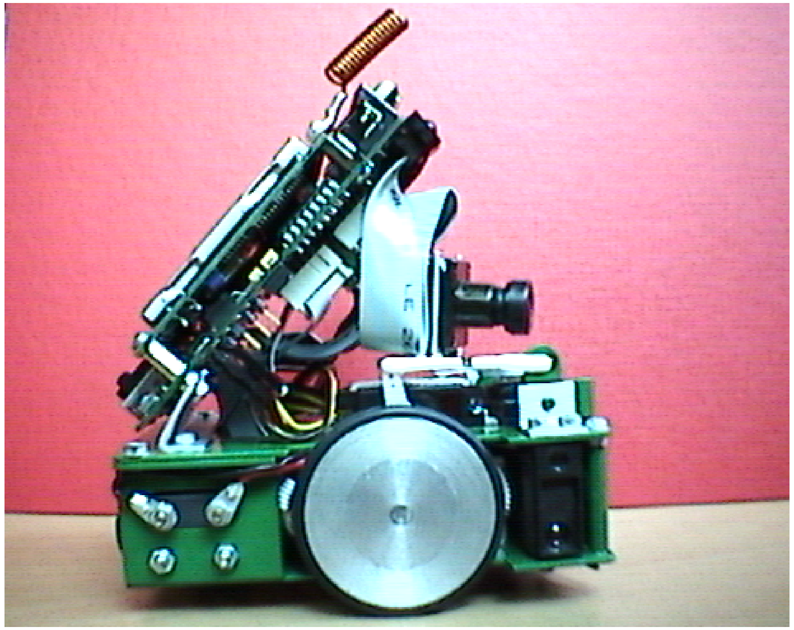
Basic behaviors for EyeBot robots (robosoccer)

Robosoccer environment



EyeBot robot

- 3 infrared sensors
- 1 camera (83 x 64 pixels)
- Programming: RoBIOS
- Radio communications
- On-board display (B&W)

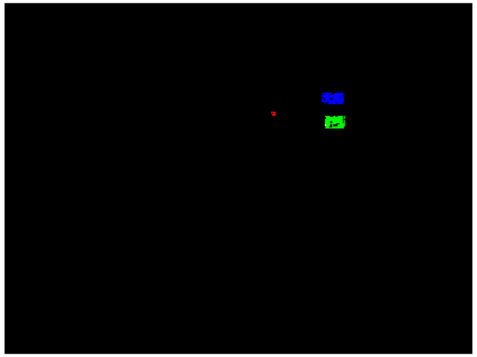


Finding robots and the ball (off-board)

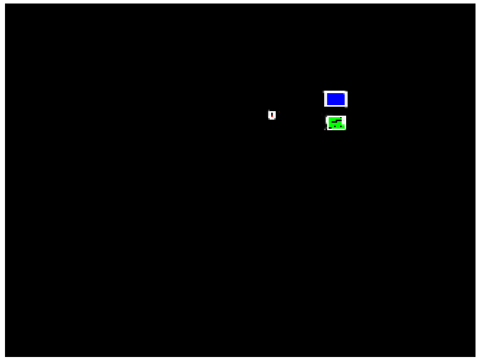
Initial image

Filtered image

image



Segmented

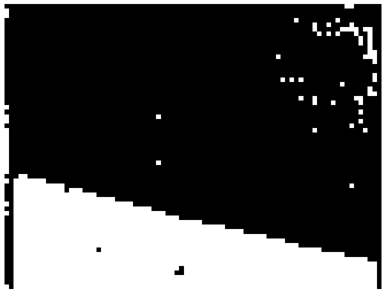
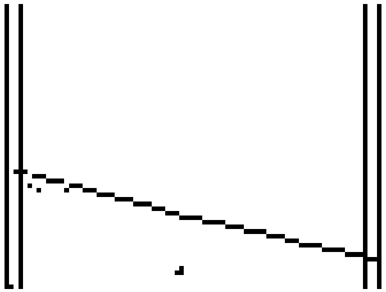


Finding and following lines (on-board)

Color filtering

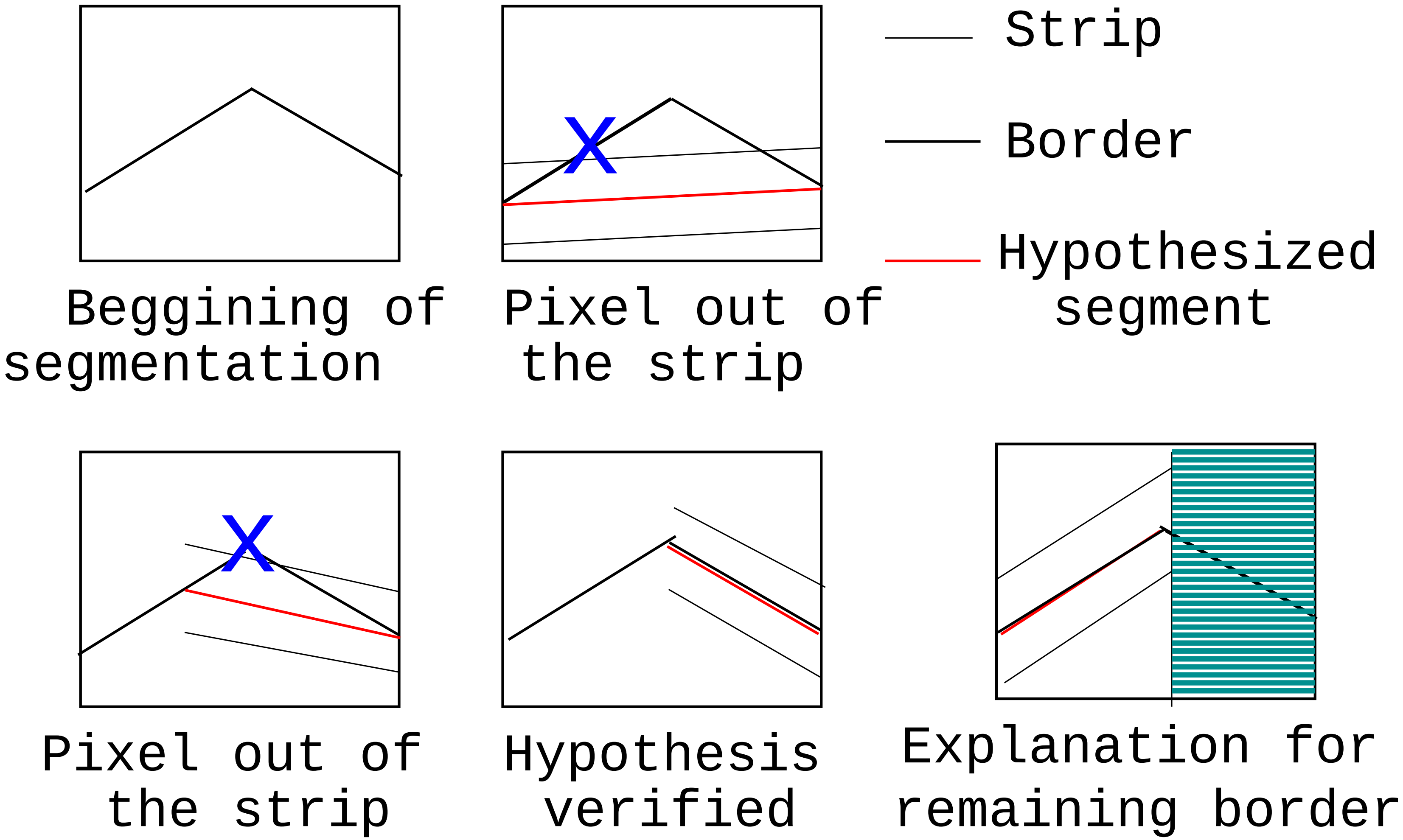

- RGB filter.
- Manual tunning.

Border analysis

- Bottom-up analysis
- Robust

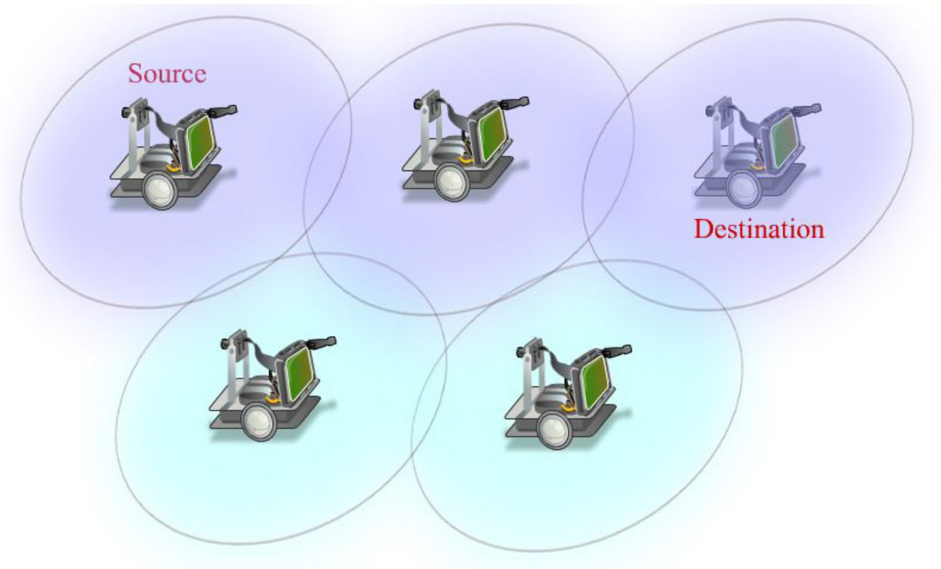
Segmentation



PERA: Ad-hoc networks for mobile robots

What is PERA?

- Library for Robot communication
- Wireless and dynamic network
  - Radio with limited scope
  - Robots are moving all the time
  - No infrastructure
- Each Robot operates as a router
- Fixed routing is not suitable

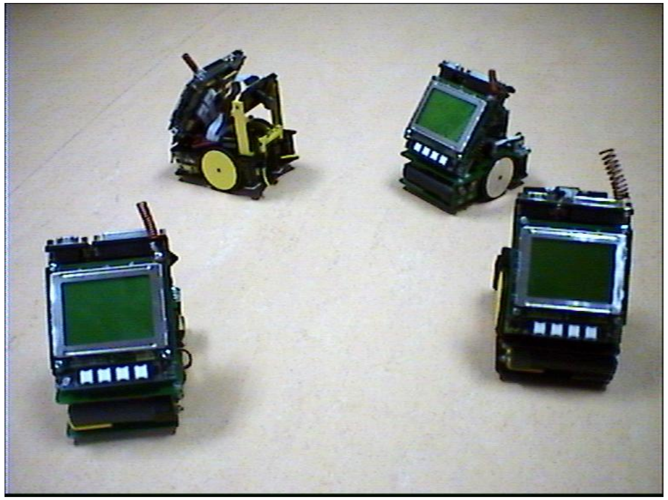


Ad-Hoc Routing protocols

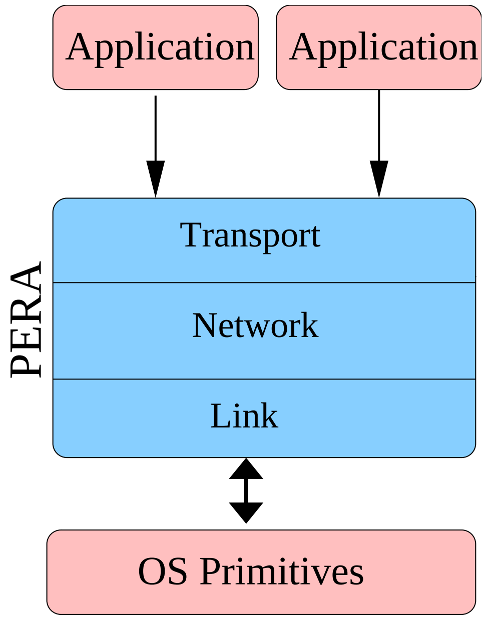
- Based on routing tables
- Based *on demand* routing
- AODV
  - Route discovery: (*RREQ*)/ (*RREP*)
  - Route maintenance

EyeBot Radio

- Radio module
- Small bandwidth (9600 baud)
- Limited data size (35 bytes)
- Robot OS (RoBios)
  - Radio system calls: *Send / receive*



Goals & Design



1. Network transparency for applications
   - PERA functions replace OS primitives
2. Multiple programs can communicate simultaneously

Top-down design: Application, transport, network, link

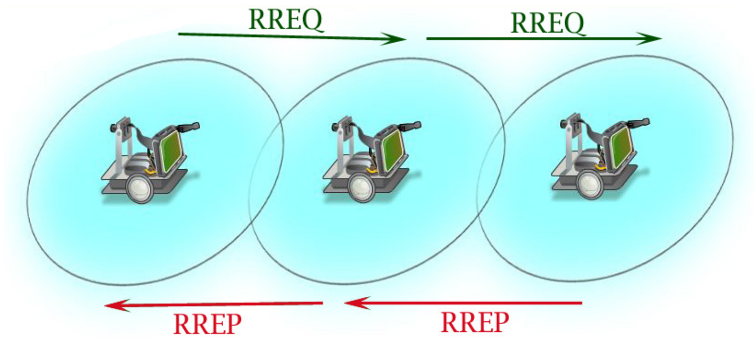
- Similar to TCP/IP model

Link level

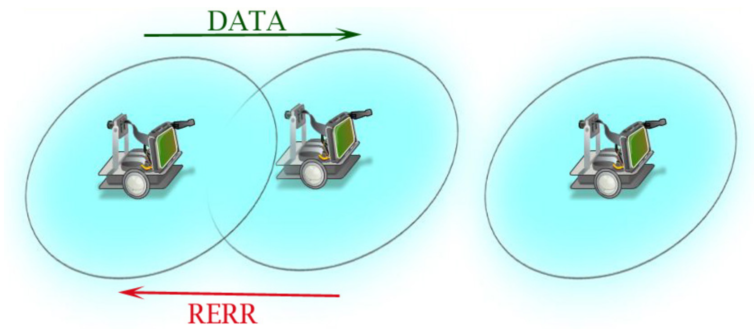
- Transmission is not reliable
- Non Blocking receive function
  - Check for new messages before *receive*

Network level

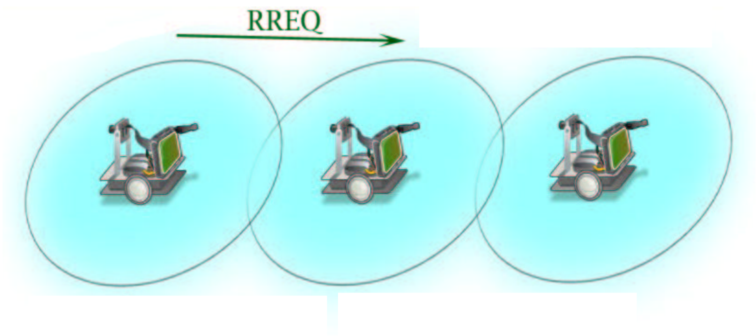
- Protocol based in *on demand* routing for *Ad-Hoc* networks
- Route discovery
  - RREQ (Route Request) message
  - RREP (Route Reply) message

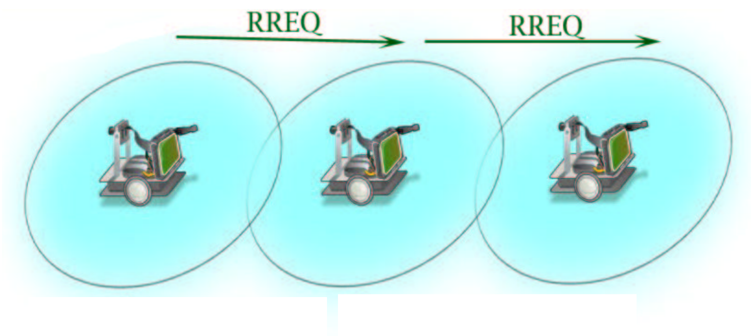


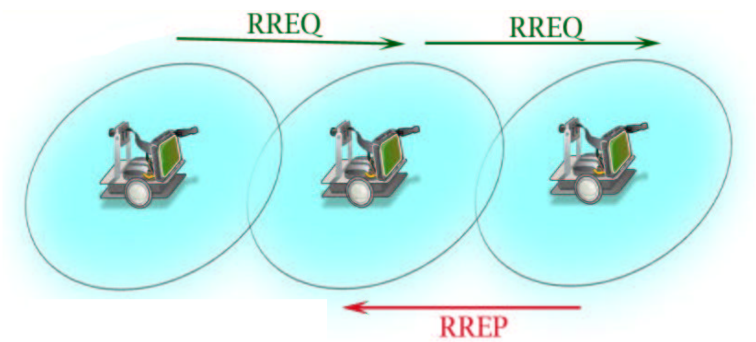
- Route maintenance
  - RERR (Route Error) message
  - HELLO packet

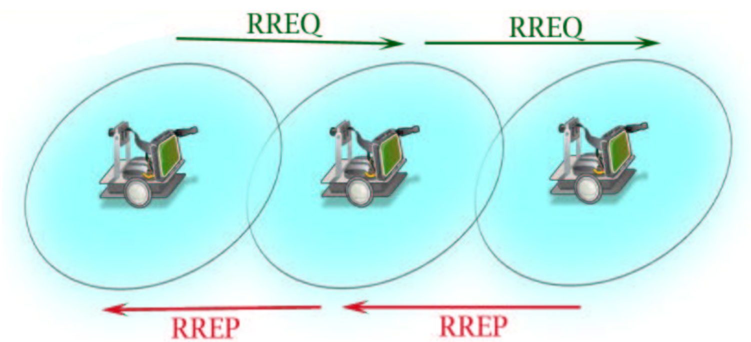


Route discovery



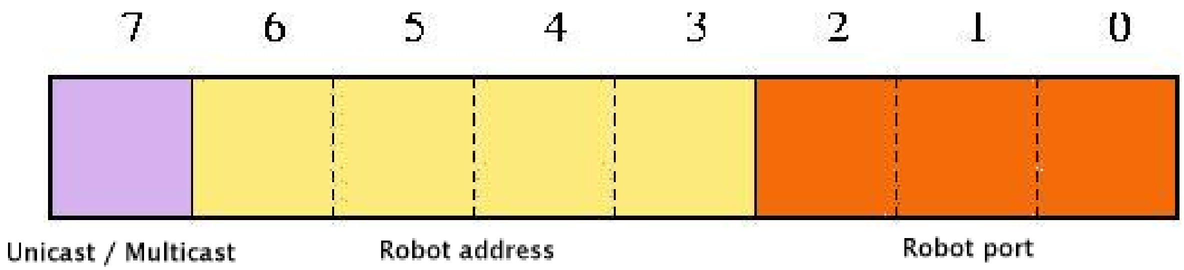




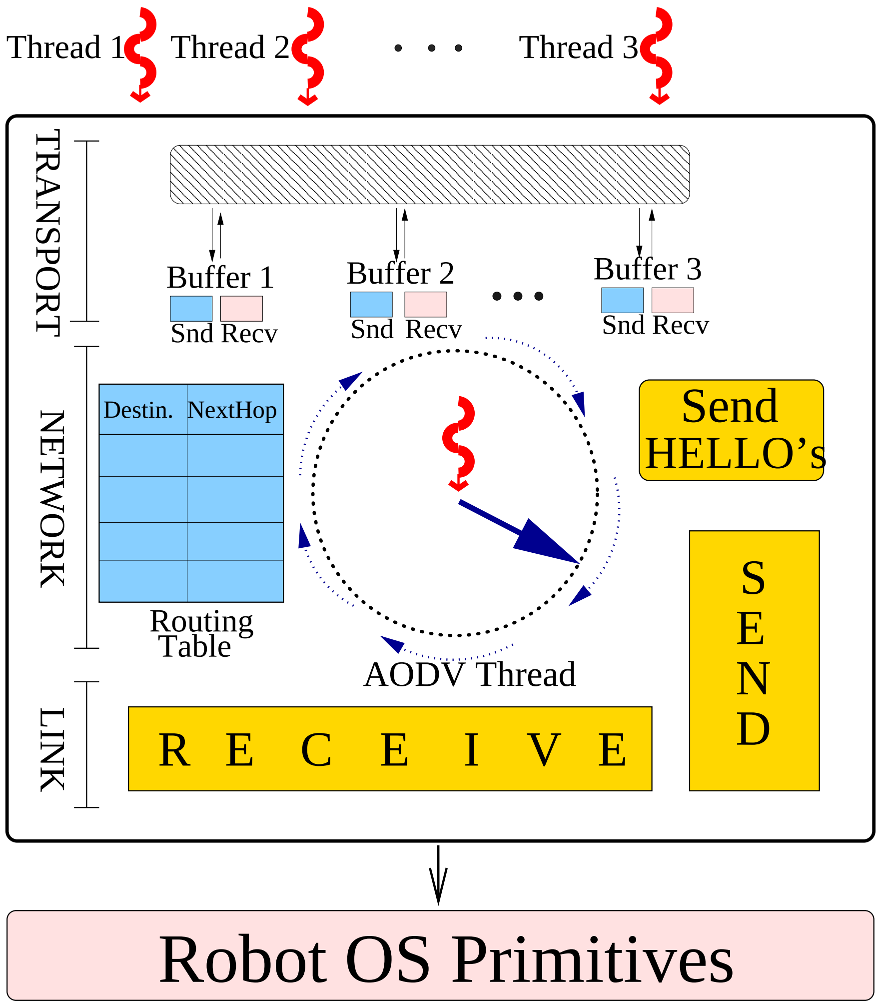


Transport level

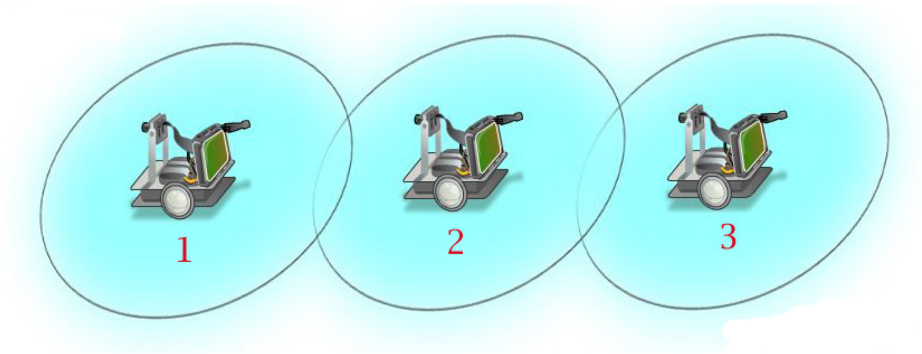
- Provides communication interface to applications
  - *bind (port)*
  - *send (robot,port,data)*
  - *receive (port, &data)*
- A good abstraction ->Ports.
  - Multiplexes the radio channel
  - Several applications communicate simultaneously
- Addressing scheme



Implementation



Tests



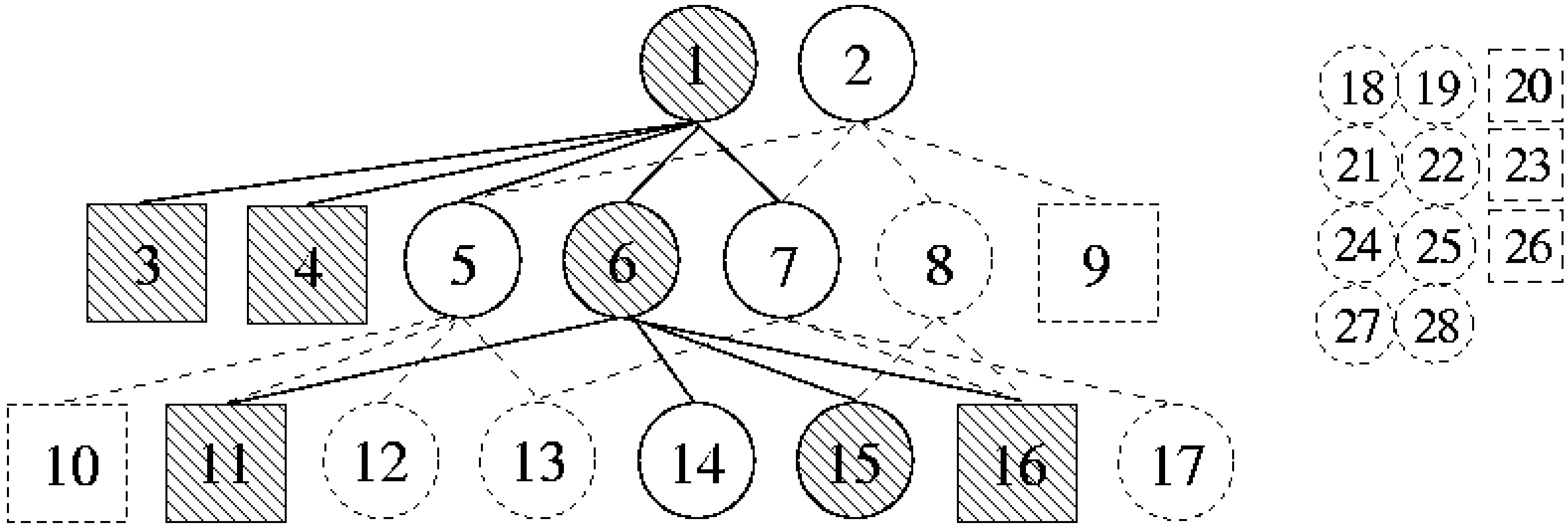
- $T_{Direct\ send\ 1\ -> \ 2} \approx 1{,}75\ secs$
- $T_{Direct\ send\ 1\ -> \ 3} \approx \infty\ secs$
- $T_{Route\ discovery\ 1\ -> \ 3} \approx 5{,}5\ secs$
- $T_{PERA\ send\ 1\ -> \ 3} \approx 5{,}5 + 3{,}5 \approx 9\ secs$

Dynamic Schema Hierarchies

Dynamic Schema Hierachies

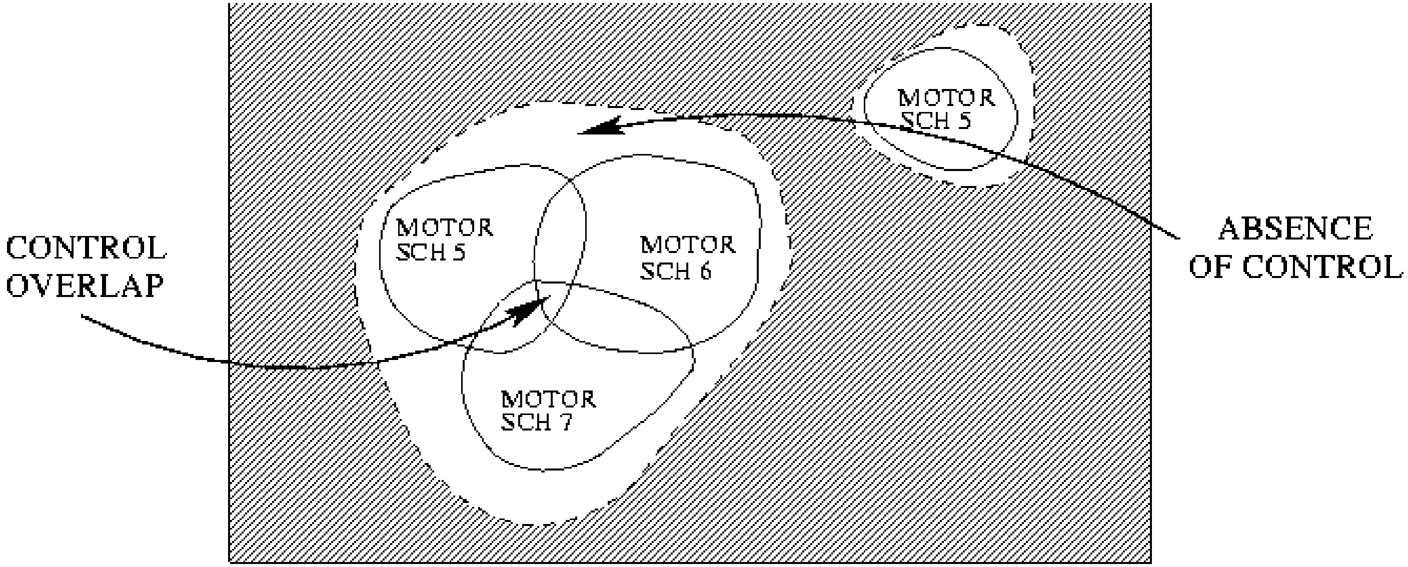
- architecture = perception + control
- perception and control are partitioned in small units: **schemas**
  - *perceptual schemas*
  - *control schemas*
- schemas can be combined in **hierarchies**
  - a control schema activates child perceptual schemas to identify relevant stimuli and child control schemas which react accordingly to them.
  - child control schemas implement parent's behavior while pursuing their own.

System snapshot



- Non blocking activation
- Continuous **modulation** through parameters
- Active perceptual schemas configure a **perceptual space** per level.

Action selection

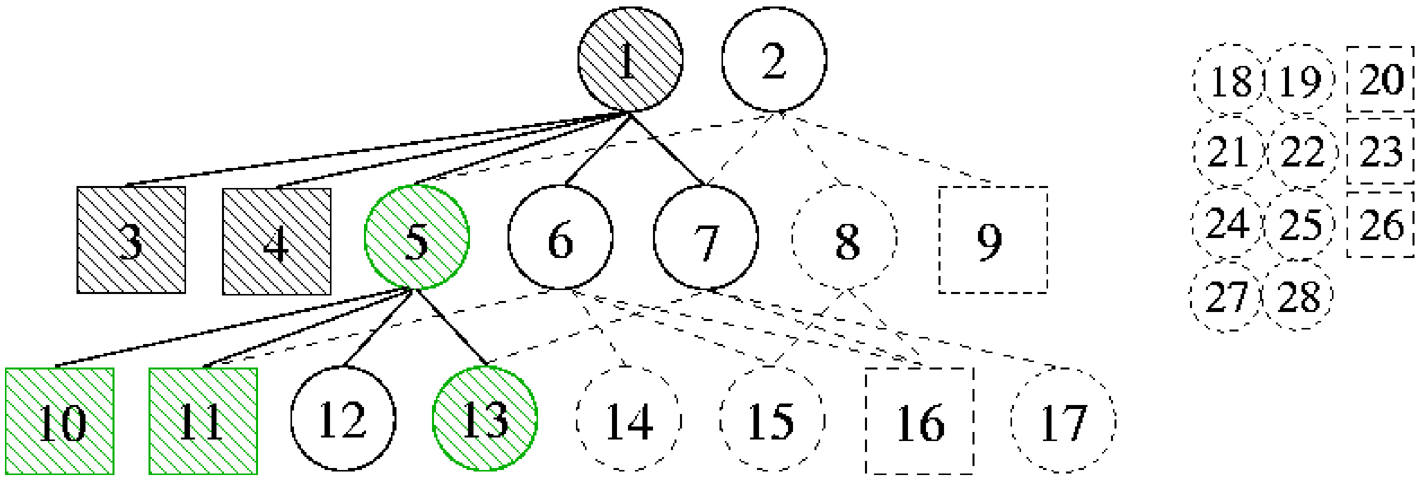


- Control competition per level.
- Winner must be awake and be appropriate for current situation.
- **Activation regions** for coarse grain arbitration.
- Parent is called for arbitration in *control overlaps* and *absences*.

Situated perception

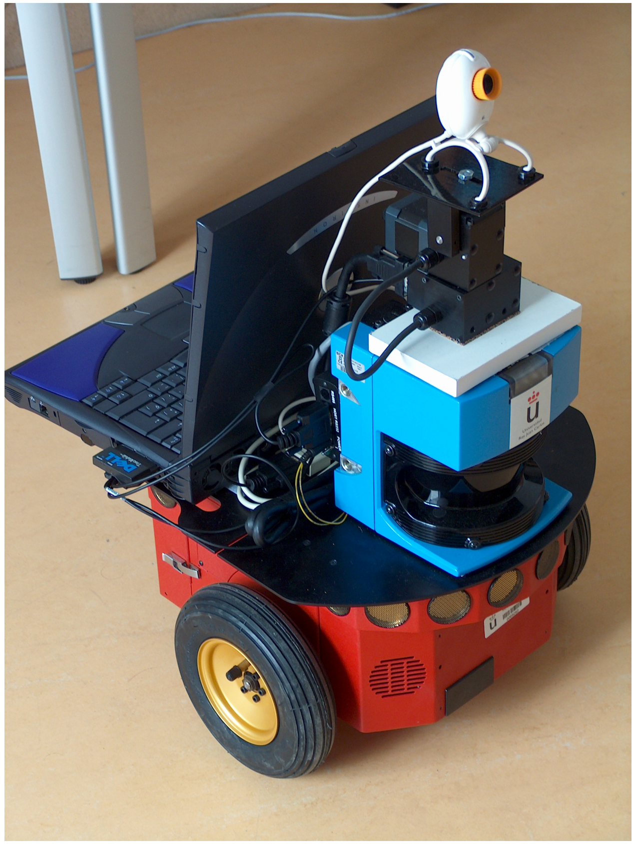
- perception is partitioned.
- it is explicitly taken into account in the architecture.
- context activation offers an **attention mechanism**.
- complex stimulus as a hierarchy of perceptual schemas (e.g. door = depth discontinuity + visual jamb)

Reconfiguration



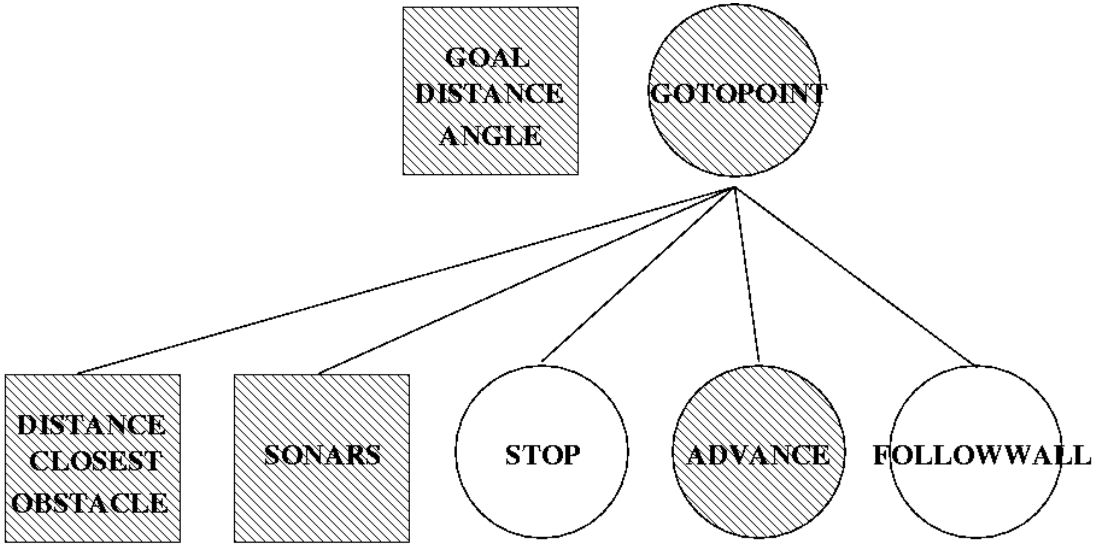
- Number of levels depends on the task and is dynamic.
- Schemas can be reused at different levels.
- **Monitoring** is included in each schema, causing reconfiguration.
- An exception climb up for an schema able to deal with it.

Implementation

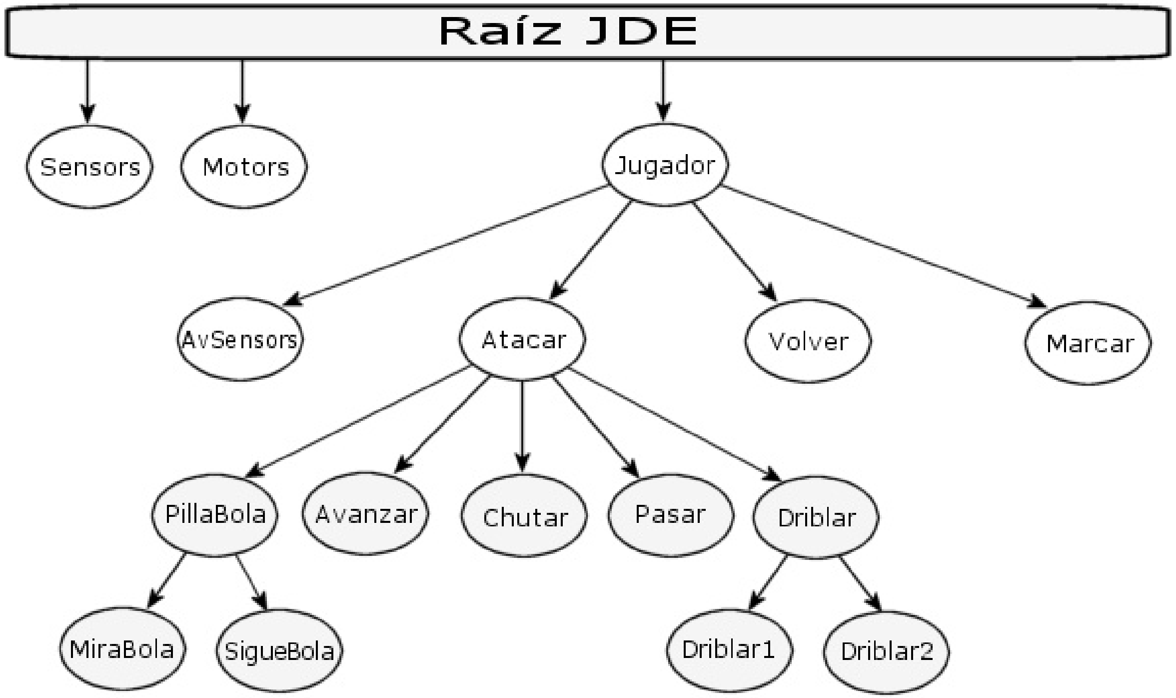


- Pioneer robot with a Linux laptop.
- Schemas ≈ programming threads.
- Control loops (e.g. 100 ms)
- Shared variables, semaphores.
- List of brothers, arbitration callback.

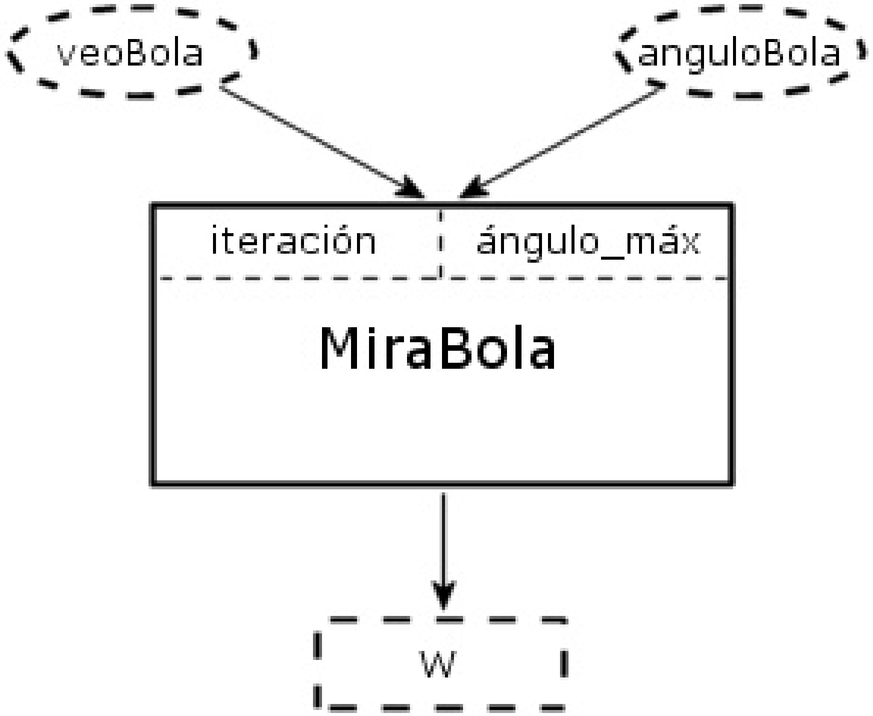
Gotopoint behavior



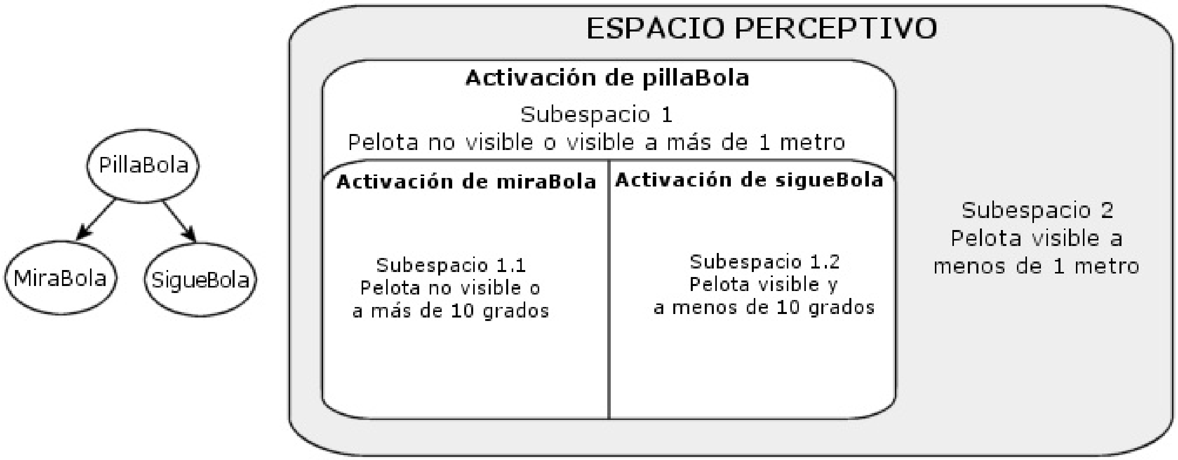
RoboCup behavior (attack)



RoboCup behavior (modulation)



RoboCup behavior (perceptive space)



Robot localization using WiFi signal

Localization

Localization: Problem of determining the position of a robot in a map

- Different solutions (and problems):

Specific sensors (i.e. GPS): Outdoors only

Odometry: Noisy sensors, accumulated errors

Artificial landmarks (+Kalman): Engineerization

Natural landmarks (+Kalman): Complex recognition of landmarks

Range sensors (+probabilistic framework): Computational cost

Our approach

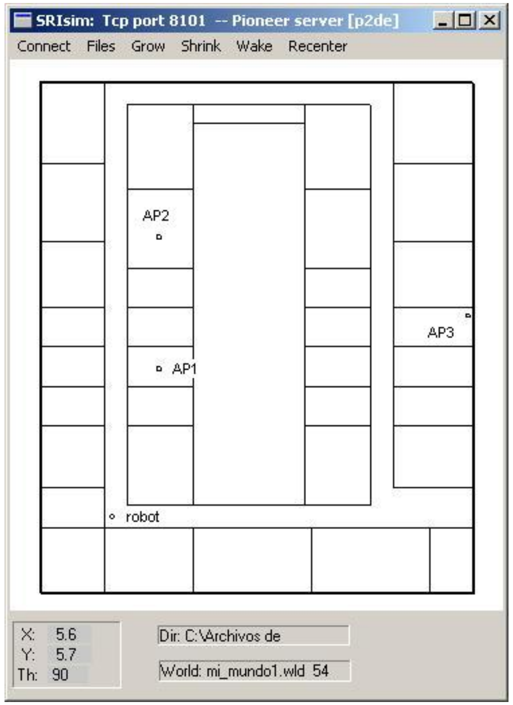
Use a probability distribution to accumulate position estimations using information from odometry and WiFi energy received from Access Points placed in a-priori known position

Advantages:

- Infrastructure has already been deployed
- Probabilistic localization has been successfully tested (Simmons95, Thrun00).

Probabilistic localization

Our environment



Theoretical aspects

- $p(x(t))$ is the probability of the robot to be in the location x
- Considering Markovian independence in (equation 1) and using (thrun98), probability can be computed incrementally (equation 3).
- Action model (robot movements) are integrated in 4.

$$p(x(t)) = p(x(t)/obs(t), obs(t-1), ...) \quad (1)$$

$$p(x(t)) = p(x(t)/obs(t)) * p(x(t)/obs(t-1), obs(t-2), ...) \quad (2)$$

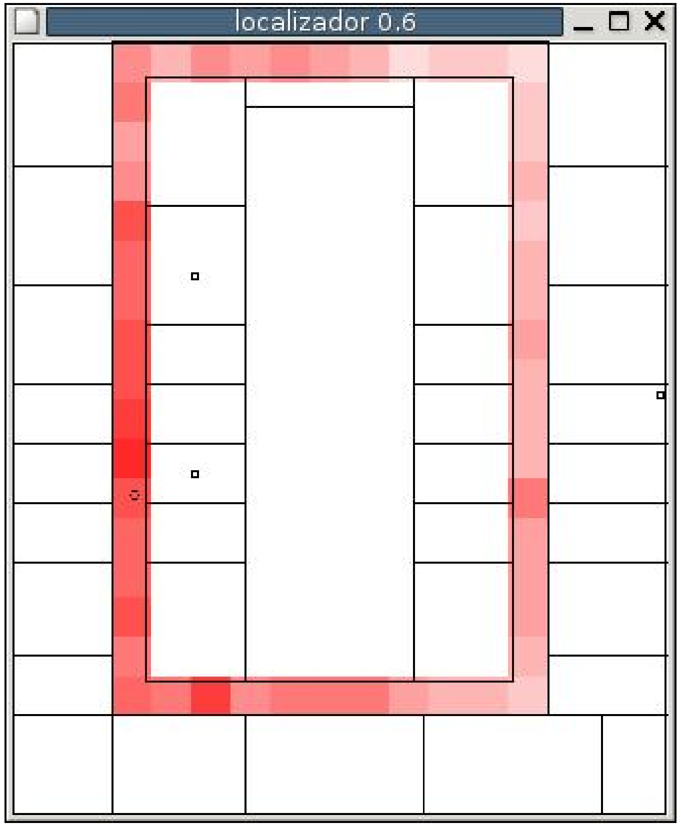
$$p(x(t)) = p(x(t)/obs(t)) * p(x(t-1)) \quad (3)$$

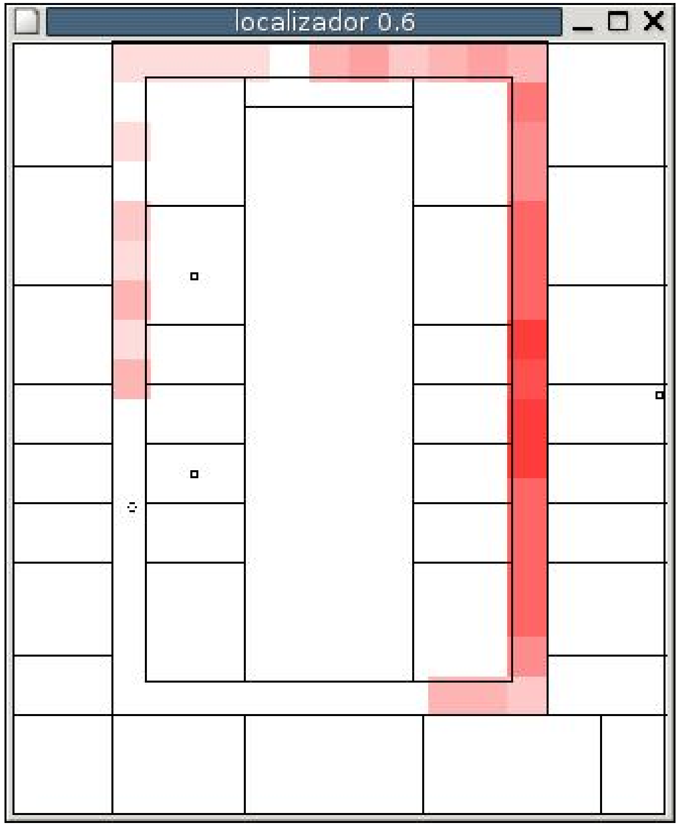
$$p(x(t)) = p(x(t)/mov(t-1), x(t-1)) * p(x(t-1)) \quad (4)$$

Sensor model

- Posterior sensor model: $p(x(t)/obs(t))$ contains all the position information carried by the observation.
- A priori sensor model: $p(obs(t)/x(t))$, which contains the probability to obtain the given sensor measurement $obs(t)$ in time t if the robot were at position x at time t.
- We will use WiFi energy measurements as main sensor observations, defined as a vector of visible AP and their signal level
- Ad-hoc posterior model: compare signal values with expected ones at each location. Normalize distance function.
- Two versions: a priori compiled energy map, and a theoretical WiFi propagation model.

Three a priori WiFi energy maps





Sensorial model when using energy maps

$$p(x/obs(t)) = 1 - d(t)\sigma$$

where:

- σ is an amplification factor
- $d(t)$ is computed as the percentage of energies from the sensor reading vector that fall close to its corresponding element in the expected vector
- A given threshold is set to consider two energies as close enough

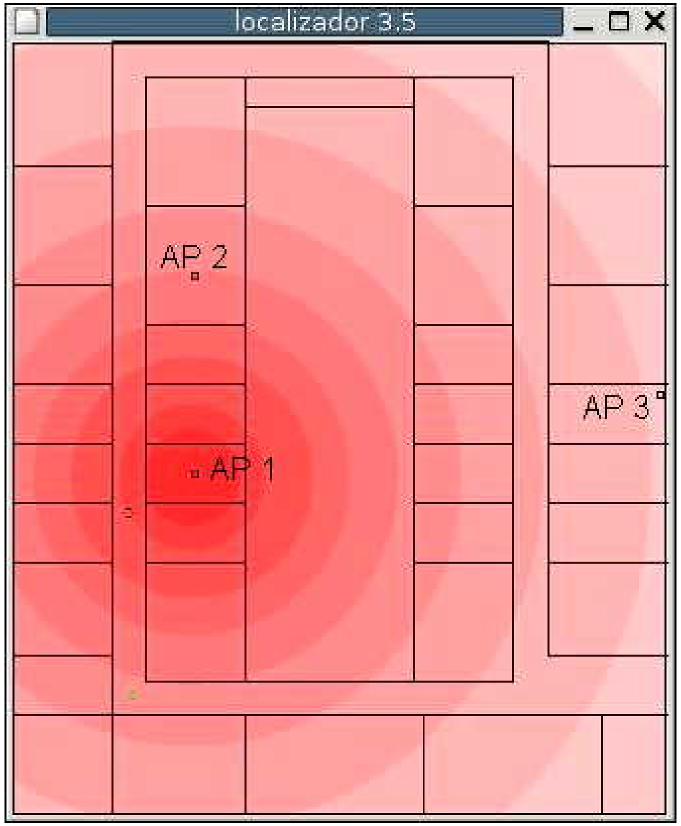
WiFi propagation model

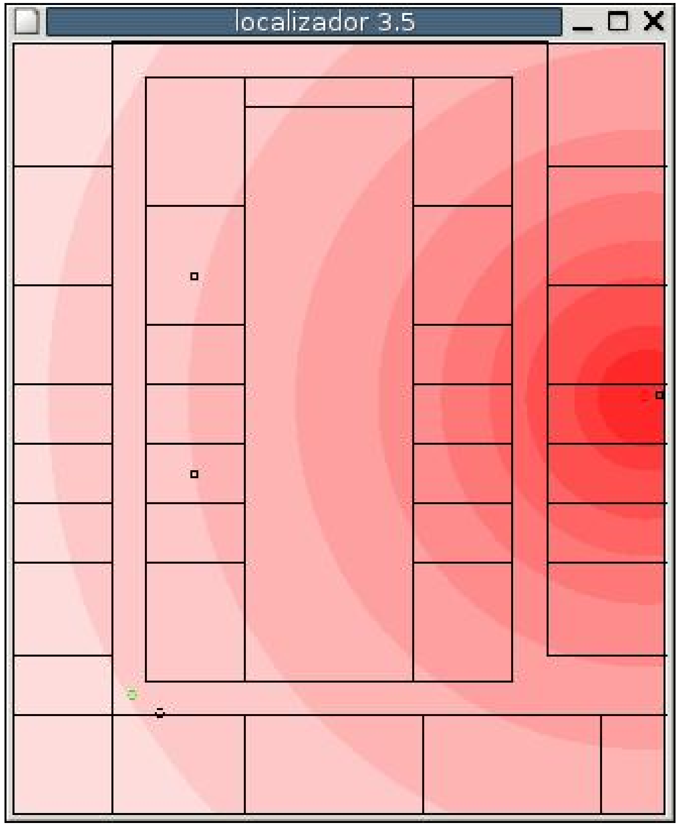
$$d(t) = e^{-\left(\sum_{i=0}^{AP}\left(\frac{|r^i_{obs} - r^i_x|}{100}\right)\sigma\right)^2} \tag{5}$$

where:

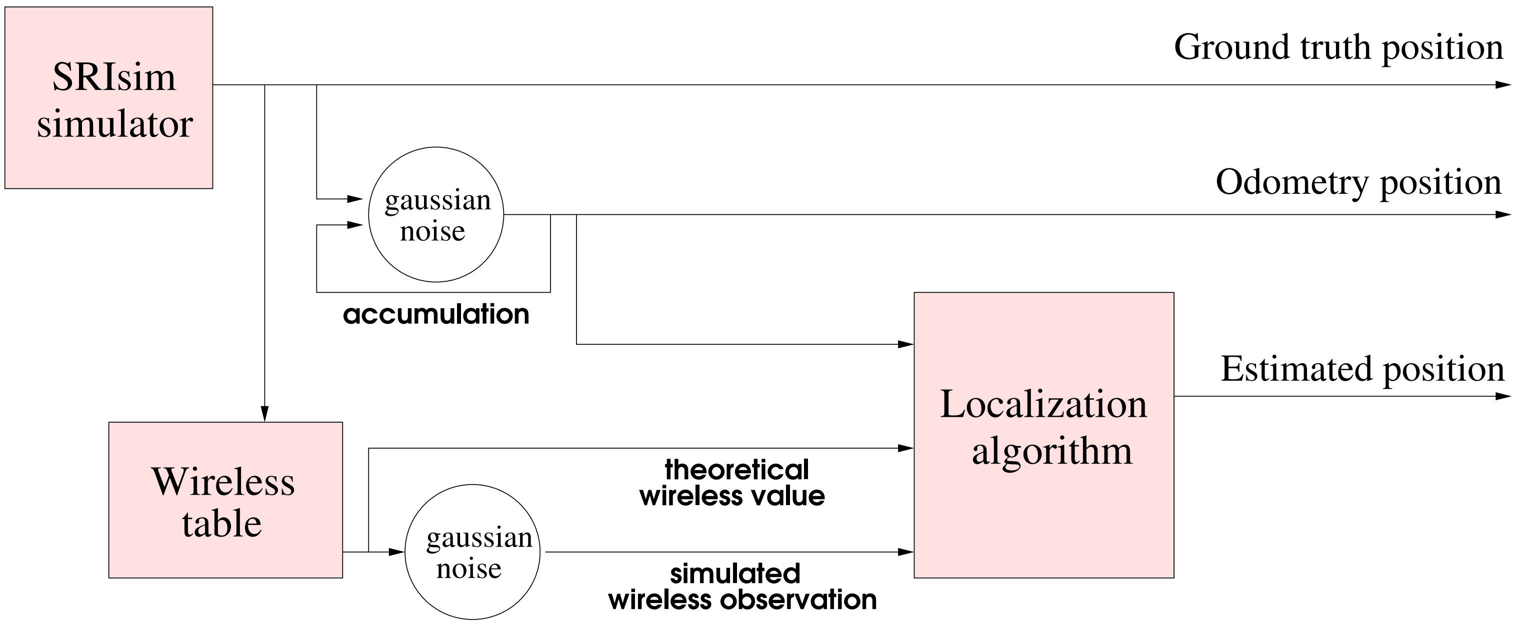
- Based on the breakpoint model (Clarke02).
- This is a free space loss model that takes into account only the distance from the emitter
- Two different regions are defined: before and after a *breakpoint*

Propagation models for AP1 and AP3

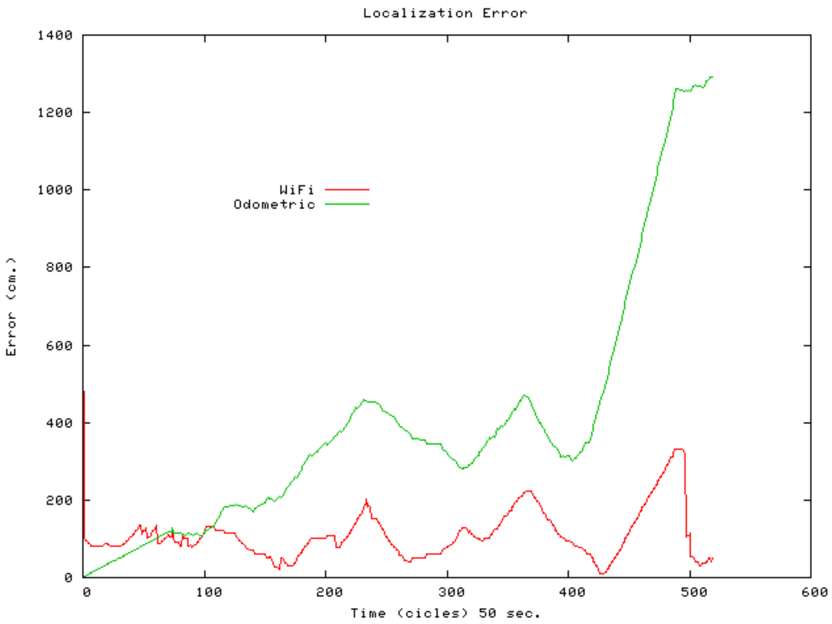


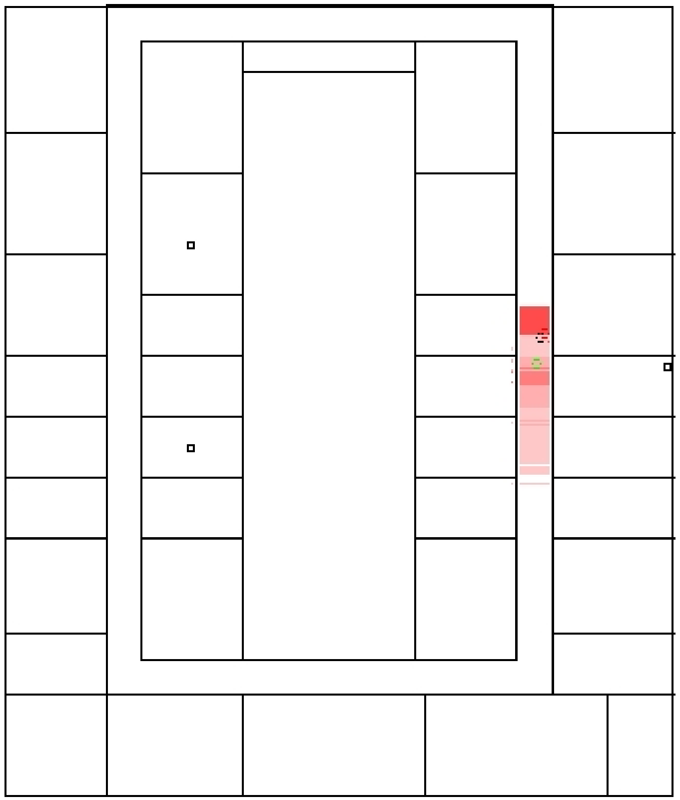


Experiments

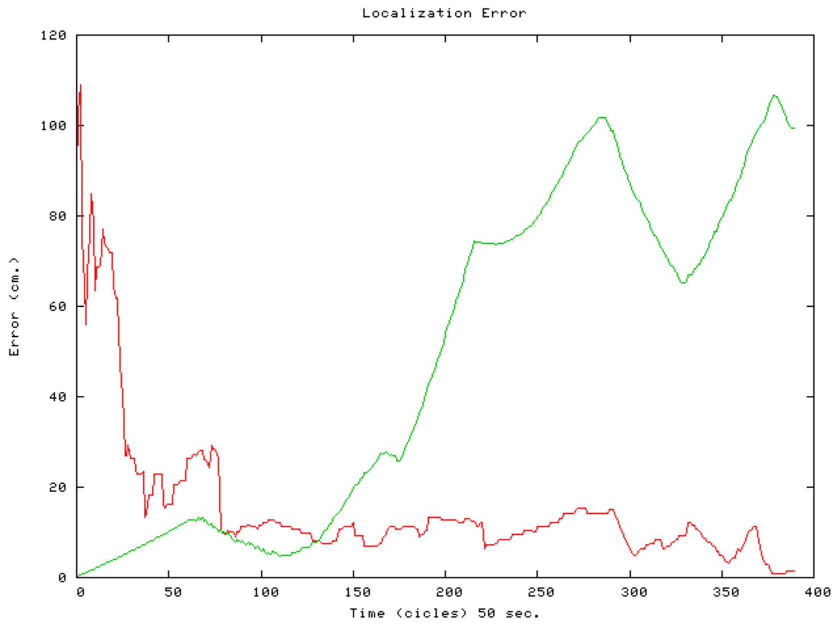


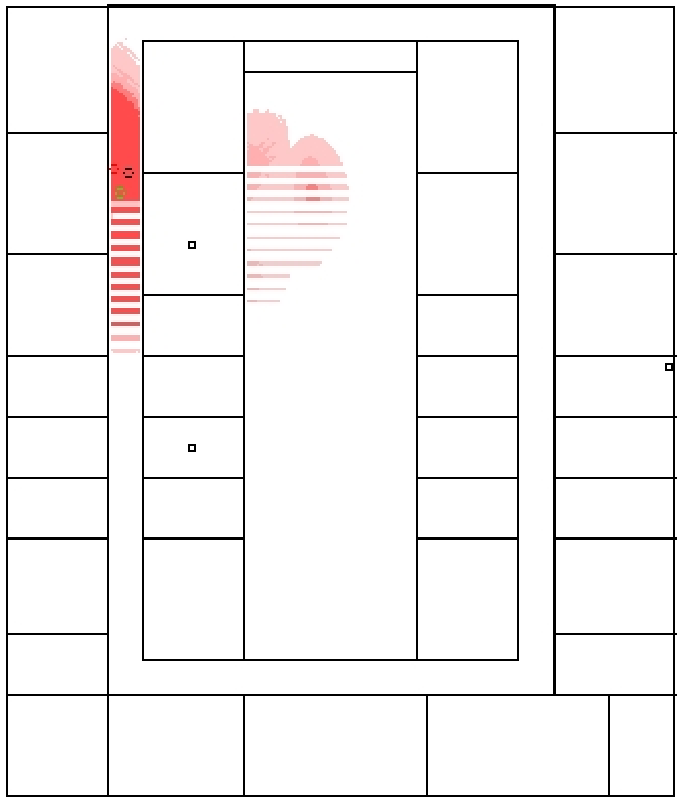
Results using energy maps





Results using WiFi propagation model



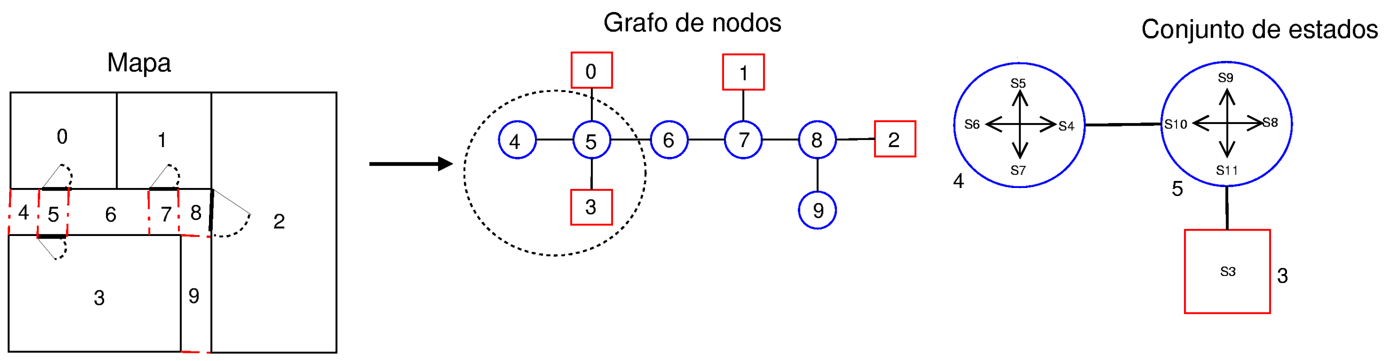


Probabilistic topological navigation in a legged robot

Idea

- Interested in localization indoor for a legged robot
- Odometry is not reliable, we will use topological maps (set of states S)
- We will calculate the probability $p(s_i$ of robot being located at state $s_i \in S$ using POMDP
- In order to calculate it we will use:

Set of states S



- We will use a topological map of the environment.
- We will use represent rooms and corridors as nodes

  Room node Creates a single state.

  Corridor node Creates 4 states, depending on robot orientation

- $S = \{s_0, s_1, s_2, ..., s_{29}\}$

Set of actions A

Set of actions: $A = \{a_r, a_l, a_f, a_o, a_e\}$

Action a_r *Turns 90º right*

Action a_l *Girar 90º left*

Action a_f *Follows corridor*

Action a_o *Leaves room*

Action a_e *Comes into room*

Transition function T

- We establish a general uncertainty model for the actions:

  Action a_r p(doing nothing)=0.05, p(turning right 90°)=0.90, p(turning right more than 90°)=0.05

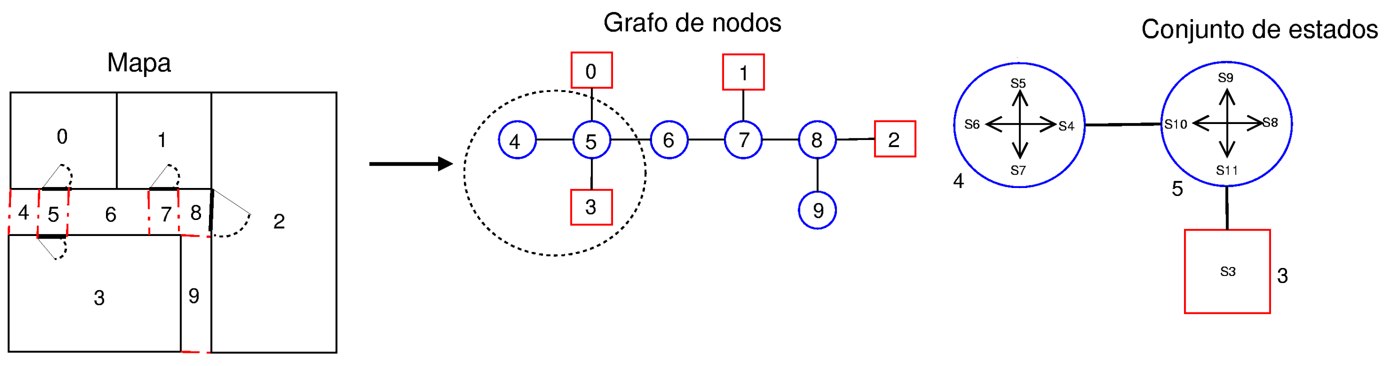
  Action a_l p(doing nothing)=0.05, p(turning left 90°)=0.90, p(turning left more than 90°)=0.05

  Action a_f p(doing nothing)=0.10, p(goes forward enough to get the next state)=0.70, p(goes further than desired)=0.15, p(goes much more further)=0.05

  Action a_o p(doing nothing)=0.05, p(leaving room)=0.85, p(leaves room and goes further than desired)=0.10

  Action a_e p(doing nothing)=0.10, p(Comes into the room)=0.90

- From this model we create a transtion fuction, which is modeled as a table for each action. This table summarizes the probability of going from state s to state s' taking that action.



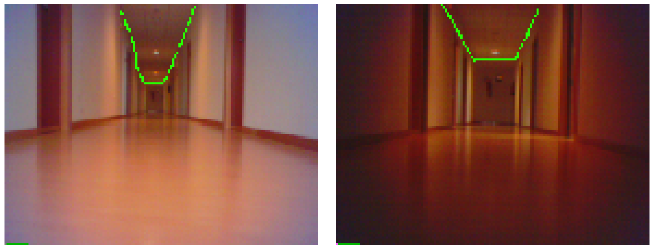
p	$\cdots$	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	$\cdots$
$origen\mathbf{4}$	$\cdots$	10	0	0	0	70	0	0	0	15	0	$\cdots$
$origen\mathbf{5}$	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
$origen\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$origen\mathbf{8}$	$\cdots$	0	0	0	0	10	0	0	0	70	0	$\cdots$

Observation function ϑ

- We model the probabilities of getting an observation from each stata, that is, $p(\mathbf{o}|s)$
- We will use various types of observations

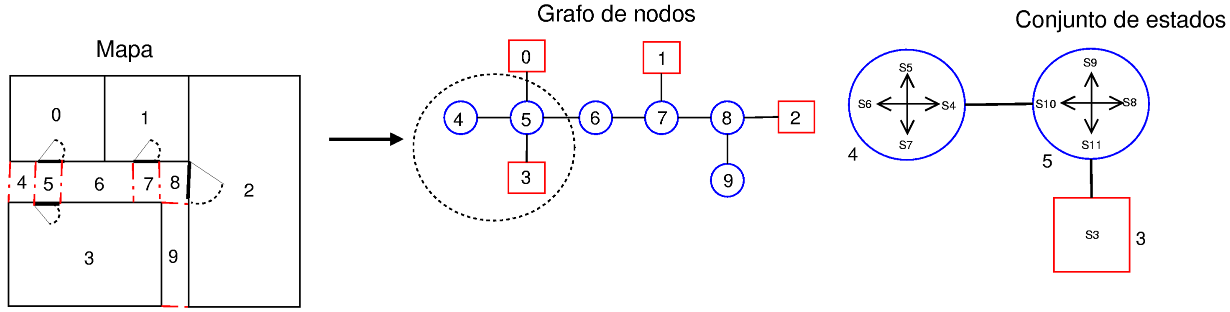
  OVDoors Number of doors in an image

  OVDepth Distance to the end of a corridor (using ceiling measures)



- Giving that observations are independent:

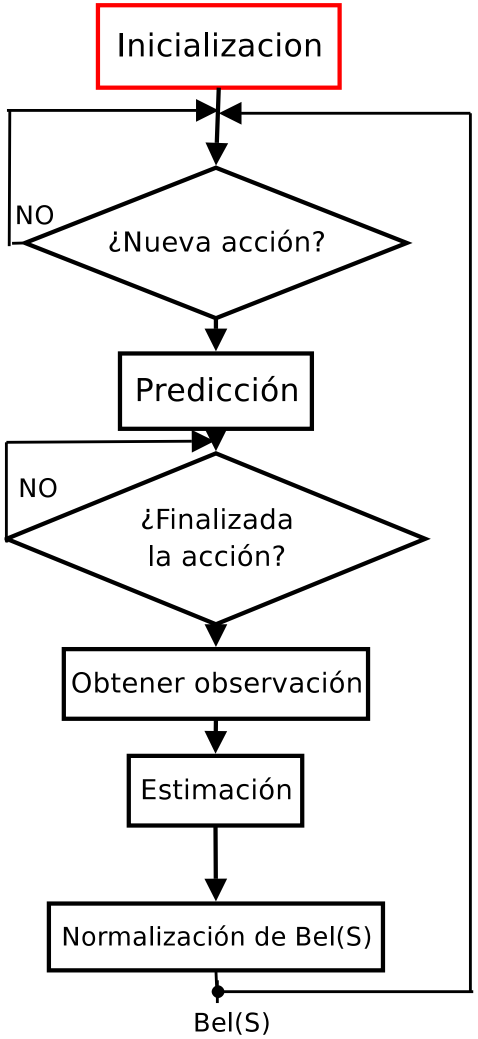
$$p(\mathbf{o}|s) = p(o_{OV\,Doors}, o_{OV\,Depth}|s) = p(o_{OV\,Doors}|s) \cdot p(o_{OV\,Depth}|s)$$



Sample matrix: de $p(o_{OV\,Depth}|s)$

p	$\mathbf{o_{ovp0}}$	$\mathbf{o_{ovp1}}$	$\mathbf{o_{ovp2}}$
...	...	...	...
$\mathbf{s_4}$	3	7	90
$\mathbf{s_5}$	95	4	1
$\mathbf{s_8}$	5	15	80
$\mathbf{s_{10}}$	85	10	5

Initialization

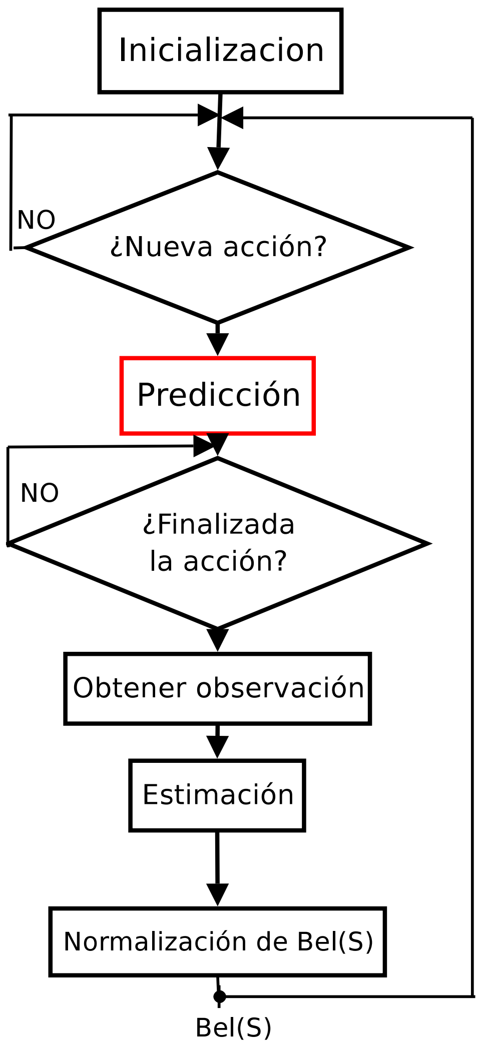


- If we do not know where we are,

$$Belief(s) = \frac{1}{n^{o} of states}, \forall s \in S$$

- I we do know where we are,

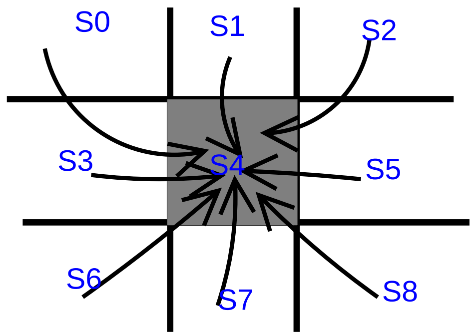
$$Belief(s) = \begin{cases} 1 & s = s_{ini} \\ 0 & s \neq s_{ini} \end{cases}$$

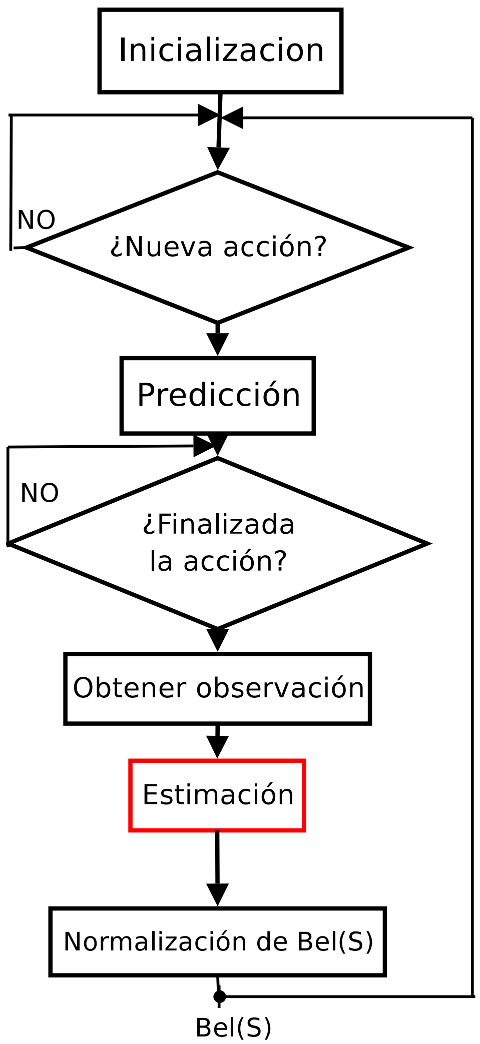


- We calculate the belief $Belief(S')$ after doing an action

$$Belief_t(S') = \sum_{s \in S} p(s'|s,a) \cdot Belief_{t-1}(s), \forall s' \in S$$

- $p(s'|s,a)$ has already been calculated as a table
- $Belief_{t-1}(s)$ is the belief in the previous instant
- So, the belief for every state s depends on the rest of states





- $Belief(S')$ is calculated from the observations that we get from the environment after an action.

$$\mathbf{Belief}_{\mathbf{posterior}}(\mathbf{s}) = p(\mathbf{o}|s) \cdot Belief_{apriori}(s), \forall s \in S$$

$$\mathbf{Belief}_{\mathbf{posterior}}(\mathbf{s}) = p(o_{OV\,Doors}, o_{OV\,Depth}|s) \cdot \\ \cdot Belief_{apriori}(s), \forall s \in S$$

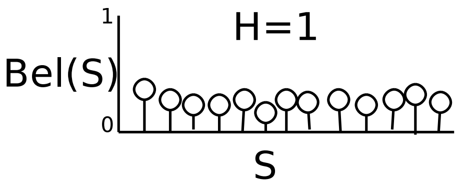
$$\mathbf{Belief}_{\mathbf{posterior}}(\mathbf{s}) = p(o_{OV\,Doors}|s) \cdot p(o_{OV\,Depth}|s) \cdot \\ \cdot Belief_{apriori}(s), \forall s \in S$$

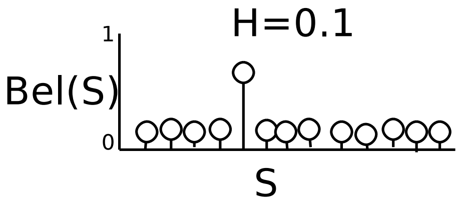
Interpretation of $Belief(S)$

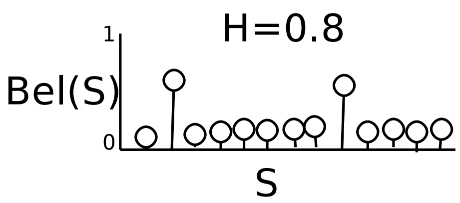
- Uncertainty of the position of the robot can be calculated using *normalized entropy*:

$$\widetilde{H} = -\frac{\sum_{Belief(s)\neq 0} Belief(s) \cdot log(Belief(s))}{log(m)}$$

- If $\widetilde{H}$ is 0, there is no uncertainty about the pose of the robot, and the larger value of $Belief(s_i), 0 \leq i < n$ will show that the robot is in state s_i.
- If $\widetilde{H}$ es 1, that means that values $Belief(s_i), 0 \leq i < n$ are equal, so we do not have any information about the position.







SiS: Speed Intelligent System

Work in progress