Robótica Móvel e Inteligente

Mobile Robot Localization

Artur Pereira <artur@ua.pt>

DETI / Universidade de Aveiro

Sumário

Navigation
Questions and topics

- **Where am I?**
  - **localization**
- Where have I been?
  - mapping
- Where should I going?
  - decision
- What's the best way to get there?
  - Path planning
- How do I get there?
  - Path following and obstacle avoidance (Motion)

Localization
Introduction

- How to determine the pose of a mobile robot relative to a given map of the environment?
  - Using sensors – beacons for triangulation, distance sensors, compass, odometry, inertial sensors, motion orders, ...
  - Using an appropriate, accurate enough map of the environment
- Difficulties:
  - In general, the pose cannot be sensed directly
    - it has to be inferred from data
  - A single sensor measurement is usually insufficient to determine the pose
    - robot has to integrate data over time and/or from different sources
  - The exact pose of a robot is, in general, not known
    - pose must be given by a probabilistic distribution
    - the robot only knows the probability of being at a given pose

Localization

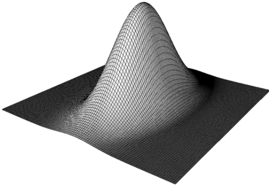
The localization problem

- Goal:
  - Localize the robot in a known map of the environment
- Inputs:
  - Map of the environment
  - Perceptions and actions of robot
- Output:
  - Estimation of pose relative to the map
    - In 2D spaces, this is expressed as the triple (x, y, θ), where (x, y) is the robot's position and θ its heading
    - In 3D spaces, 6 coordinates can be required, 3 for position and 3 for heading (roll, pitch and yaw)
- There are different approaches to tackle this problem

Localization
Markov Localization

- Probabilistic state estimation is applied to the localization problem through Bayes filters
  - It is called Markov Localization
- Markov assumption:
  - Past and future are independent, if one knows the current state
  - Sensor measures do not depend on previous measures if position is known
- In localization the state is the robot's pose

- Pose is given by a belief function
  - it is the probability distribution of the estimated pose of the robot for every possible pose



Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$**):**

for all x_t do

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1}) \;\; dx_{t-1}$$

$$bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$$

endfor

return $bel(x_t)$

- bel$(x_{t\text{-}1})$ is the belief at time t-1; u_t the actions at time interval $[t-1, t)$; z_t the measurements at time t; and m the map of the environment
- $\overline{\text{bel}}(x_t)$ is the belief at time t based only on the actions
- bel(x_t) is the belief at time t based on actions and measurements
- η is a normalization factor (from Bayes filter)

- Splitting actuation and measurement
  - Prediction phase – update previous estimate only based on actuation
  - Correction phase – correct prediction based on measurements

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$**):**

for all x_t do

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1}) \;\; dx_{t-1}$$

endfor

for all x_t do

$$bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$$

endfor

return $bel(x_t)$

Localization
Markov Localization

- Transposing to the discrete domain

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$**):**

for all x_t do

$$\overline{bel}(x_t) = \sum p(x_t \mid u_t, x_{t-1}, m)\; bel(x_{t-1})$$

endfor

for all x_t do

$$bel(x_t) = \eta\, p(z_t \mid x_t, m)\; \overline{bel}(x_t)$$

endfor

return $bel(x_t)$

Localization
Markov Localization – prediction phase

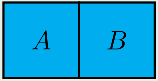
$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1}) \; dx_{t-1}$$

$$\overline{bel}(x_t) = \sum p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})$$

- Incorporates only motion model
- Input:
  - Previous belief distribution: bel(x_{t-1})
  - Action taken: u_t
- How does u_t changes bel?
  - Every possible value for x_{t-1} has to be considered on its probability of transition to x_t

Localization
Markov Localization – prediction example

$$\overline{bel}(x_t) = \sum p(x_t \mid u_t, x_{t-1}, m)\; bel(x_{t-1})$$



- Consider a world with 2 cells, A and B
- Assume the following motion model

$$P(A \mid \text{left}, A) = 0.99 \qquad P(B \mid \text{left}, A) = 0.01$$
$$P(A \mid \text{left}, B) = 0.12 \qquad P(B \mid \text{left}, B) = 0.88$$

- Assume the following previous belief

$$\text{bel}(x_{t-1}) = (0.3, 0.7)$$

- Which $\overline{\text{bel}}(x_t)$ after event left?

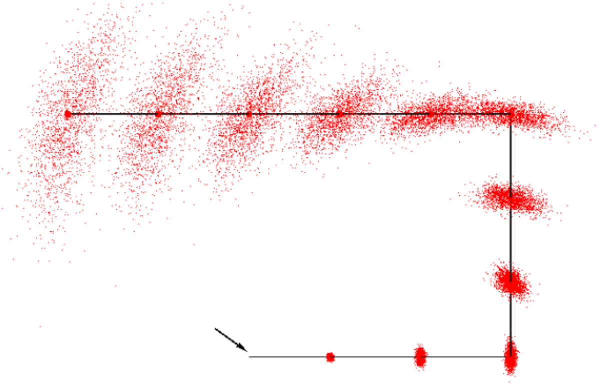
$$\overline{\text{bel}}(x_t) = (P_A, P_B)$$
$$P_A = P(A \mid \text{left}, A) * P(A) + P(A \mid \text{left}, B) * P(B) = 0.99 * 0.3 + 0.12 * 0.7 = 0.381$$
$$P_B = P(B \mid \text{left}, A) * P(A) + P(B \mid \text{left}, B) * P(B) = 0.01 * 0.3 + 0.88 * 0.7 = 0.619$$

Hence:

$$\overline{\text{bel}}(x_t) = (0.381, 0.619)$$

Localization
Markov Localization – prediction example (2)

- Example of evolution on pose estimation based only on motion model
  - every point represents a possible pose
  - as robot moves, points scatter



Localization
Markov Localization – correction phase

$$bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$$

- Incorporates sensor model
- Input:
  - Predicted belief distribution: $\overline{\text{bel}}(x_t)$
  - Sensor model
- Based on Bayes formula

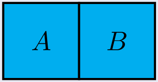
$$p(x_t \mid z_t) = \frac{p(z_t \mid x_t) * p(x_t)}{p(z_t)}$$

- $p(z_t)$ does not depend on x and may be substituted by a constant

Localization

Markov Localization – correction example

$$bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$$



- Consider the previous world and the befief after prediction

  $\overline{\text{bel}}(x_t) = (0.381, 0.619)$

- Assume the following sensor model

  $P(A|A) = 0.80 \qquad P(B|A) = 0.15 \qquad P(N|A) = 0.05$

  $P(A|B) = 0.70 \qquad P(B|B) = 0.23 \qquad P(N|B) = 0.07$

- And assume the sensor reads A
- What is the belief after the correction phase?

$$\overline{\text{bel}}(x_t)/\eta = P(A|A) * \overline{\text{bel}}(A), P(A|B) * \overline{\text{bel}}(B)$$

$$= (0.80 * 0.381, 0.23 * 0.619) = (0.3048, 0.1437)$$

Choosing η as to normalize the belief

$$\overline{\text{bel}}(x_t) = (0.6816, 0.3184)$$

Markov localization
Illustration example

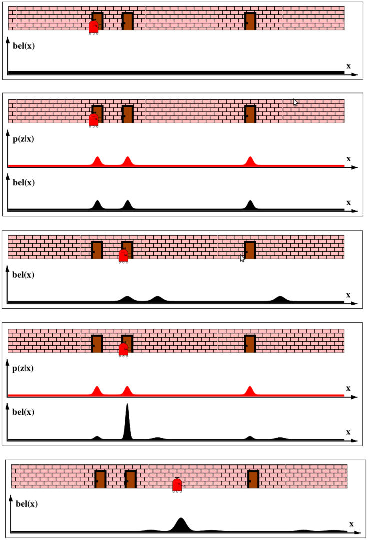
(a) Assuming the initial pose is unknown, belief is uniform

(b) Robot senses it is facing a door
- Integration of sensor data results in a multimodal distribution

(c) Robot moves some distance to the right
- convolution with motion model shifts and flattens belief

(d) Robot senses it is facing a door
- integration of sensor data allows robot to localize itself

(e) Robot moves some distance to the right
- convolution with motion model shifts and flattens belief, but robot keeps itself localized (with less confidence)



Taken from "Probabilistic robotics", Thrun, Burgard & Fox.

Localization
Kalman filter localization

- A case of Markov localization
- Implements belief computation in continuous states
- A belief is represented by a Gaussian (mean and covariance)
  - Belief shape is unimodal
- Prediction phase

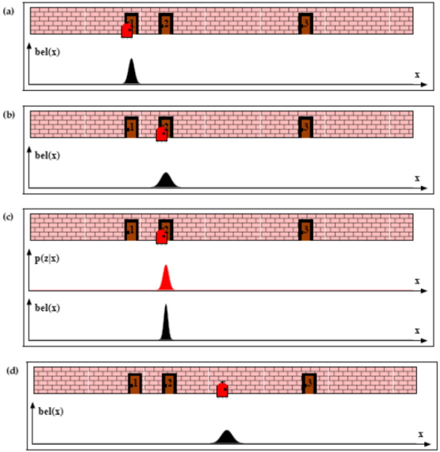
$$\mu_C = \mu_1 + \mu_2 \qquad \sigma_C{}^2 = \sigma_1{}^2 + \sigma_2{}^2$$

- Correction phase

$$\mu_P = \frac{\mu_1.\sigma_2{}^2 + \mu_2.\sigma_1{}^2}{\sigma_1{}^2 + \sigma_2{}^2} \qquad \sigma_P{}^2 = \frac{\sigma_1{}^2.\sigma_2{}^2}{\sigma_1{}^2 + \sigma_2{}^2}$$

Kalman filter localization
Illustration example

(a) Initial belief is a Gaussian distribution
(b) Motion model is applied, increasing uncertainty
(c) Sensor data is integrated, resulting in a variance smaller than variances of belief and sensor model
(d) Motion model is applied, increasing uncertainty



Taken from “Probabilistic robotics”, Thrun, Burgard & Fox.

Kalman filter localization
Extended Kalman filter

- Kalman filters' linear assumption is rarely fulfilled
- Extended Kalman filters (EKF)
  - Assume next state and measurement can be non linear

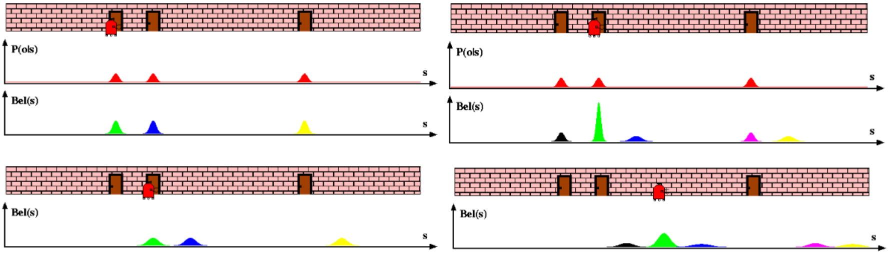
$$x_t = f(u_t, x_{t-1}) + \varepsilon_t$$

$$z_t = h(x_t) + \delta_t$$

- Moreover, instead of matrices F_t and H_t jacobians derived from f and h are used

Kalman filter localization
Multi-Hypothesis Tracking

- Extension to (extended) Kalman filter
- Belief is represented by multiple gaussians



Taken from “Probabilistic robotics”, Thrun, Burgard & Fox.

Gaussian Localization
Summary

- Unimodal Gaussian is a good uncertainty representation for tracking
  - It is not good for global localization
- Not good for hard spatial constraints
  - Close to wall, but not inside wall
  - Unable to process negative information
- Linearization can be an issue
  - depends on degree of nonlinearity
  - depends on degree of uncertainty
- Features must be sufficient and distinguishable
  - Correspondence variables

Grid Localization
Introduction

- Grid decomposition of the pose space
- Can solve the global localization problem
- Not bound to unimodal distributions
- Can process raw sensor measurements

- Uses a histogram filter to represent posterior belief
- Choice of resolution is a key point
  - High resolution $\Rightarrow$ slow computation
  - Low resolution $\Rightarrow$ information loss

- Belief is given by a set of probabilities

$$\text{bel}(x_t) = \{p_{k,t}\}$$

Grid Localization
Algorithm

Algorithm Grid_localization($\{p_{k,t-1}\}, u_t, z_t, m$**):**

for all k *do*

$$\bar{p}_{k,t} = \sum_i p_{i,t-1}\ \mathbf{motion\_model}(\mathrm{mean}(\mathbf{x}_k), u_t, \mathrm{mean}(\mathbf{x}_i))$$

$$p_{k,t} = \eta\ \bar{p}_{k,t}\ \ \mathbf{measurement\_model}(z_t, \mathrm{mean}(\mathbf{x}_k), m)$$

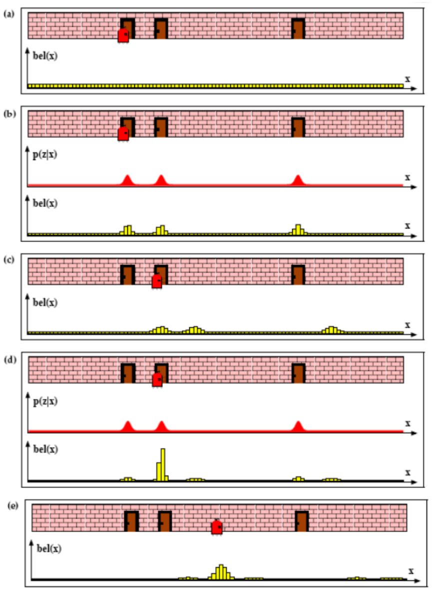
endfor

return $\{p_{k,t}\}$

- $\{p_{k,t\text{-}1}\}$ is the belief at time t-1, u_t the actions at time interval $[t-1, t)$, z_t the measurements at time t, and m the map of the environment
- $\{\overline{p}_{k,t}\}$ is the believe at time t based only on the actions
- $\{p_{k,t}\}$ is the believe at time t based on actions and measurements
- η is a normalization factor (from Bayes filter)

Grid Localization
Illustration example

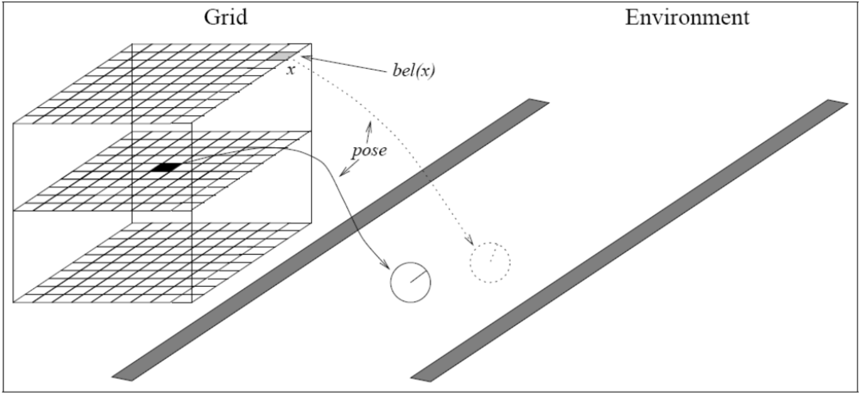
(a) Belief is uniform
(b) First integration of sensor data
  - result is multimodal
(c) Convolution with motion model, shifts and flattens belief
(d) Second integration of sensor data, robot localizes itself
(e) Moving along



Taken from “Probabilistic robotics”, Thrun, Burgard & Fox.

Grid Localization
Example for a 2D pose

- A grid to represent a 2D pose is cubic
  - each plan represents a possible robot orientation



Taken from “Probabilistic robotics”, Thrun, Burgard & Fox.

Monte Carlo localization
Introduction

- Based on particle filter algorithm
  - Using appropriate probabilistic motion and perceptual models
- Can solve the global localization problem
- Not bound to unimodal distributions

- The belief is a set of particles
  - each particle represents a pose
- Measurement is used to determine the importance weight of particles
- Weights are used to influence a random selection of particles
  - Heavier particles are more likely to be selected

Monte Carlo localization
Algorithm

Algorithm MCL(X_{t-1}, u_t, z_t, m):

$\overline{X}_t = X_t = \emptyset$

for $i = 1$ to M **do**

$x_t^{[i]} =$ **sample_motion_model** $(u_t, x_{t-1}^{[i]}, m)$

$\omega_t^{[i]} =$ **sample_mesurement_model** $(z_t, x_t^{[i]}, m)$

$\overline{X}_t = \overline{X}_t + \left\langle x_t^{[i]}, \omega_t^{[i]} \right\rangle$

end for

for $i = 1$ to M **do**

draw $x_t^{[i]}$ with probability $\propto \omega_t^{[i]}$

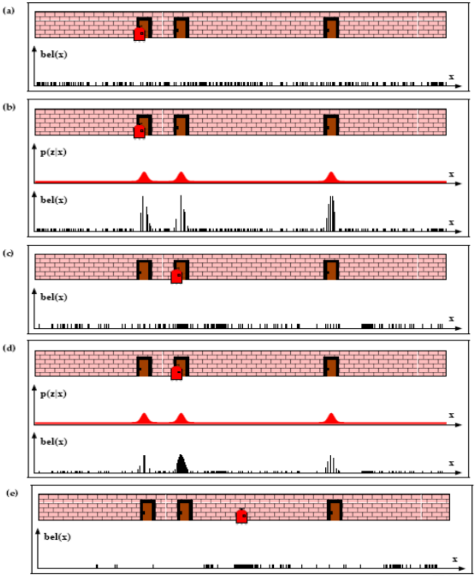
$X_t = X_t + x_t^{[i]}$

end for

return X_t

Monte Carlo localization
Example

(a) Pose particles drawn at random and uniformly
(b) Importance factor assigned to each particle
  - set of particles hasn't changed
(c) After resampling and incorporating robot motion
(d) New measurement assigns new importance factors
(e) New resampling and motion



Taken from "Probabilistic robotics", Thrun, Burgard & Fox.

Monte Carlo localization
Example (2)

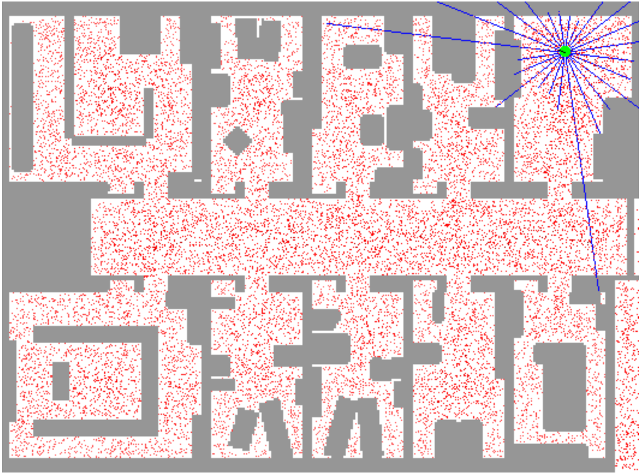


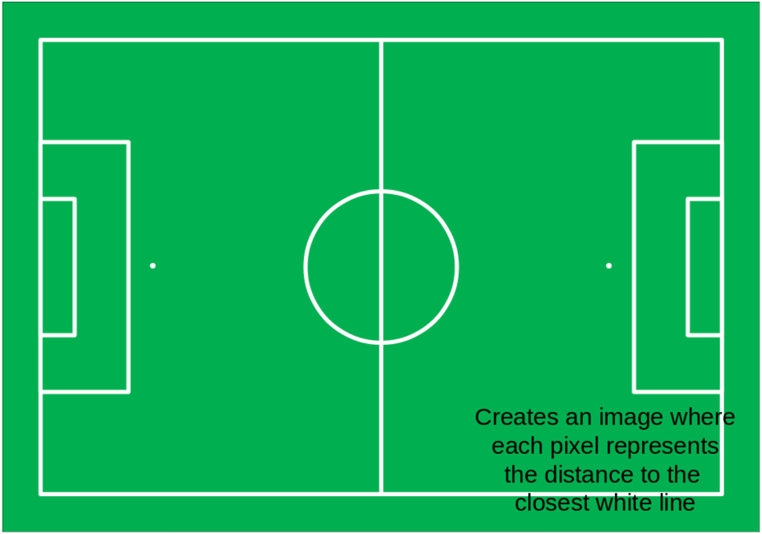
Image source https://rse-lab.cs.washington.edu/projects/mcl

Download image; it is an animated gif

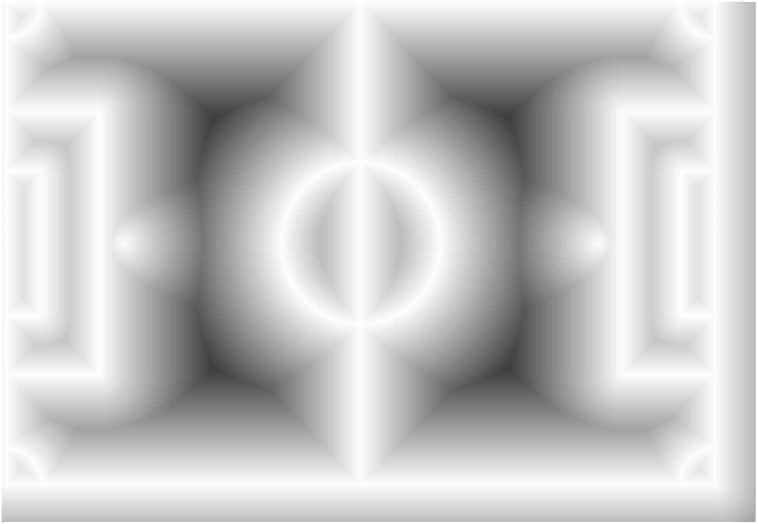
Localization in CAMBADA
Approach

- Based on Tribots localization
- Uses white lines seen by the robot
  - captured using an omni camera
- A correction map converts pixels to real distances
  - this map is constructed in a calibration phase
- A distance map of the field is used to correct robot pose
  - this map is constructed in advance and kept in a lookup table

Localization in CAMBADA
Building the field LUT

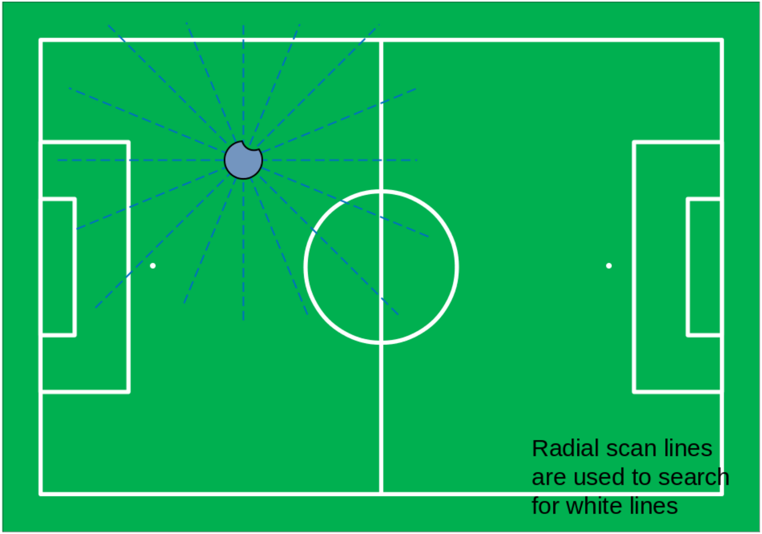


Localization in CAMBADA
Building the field LUT

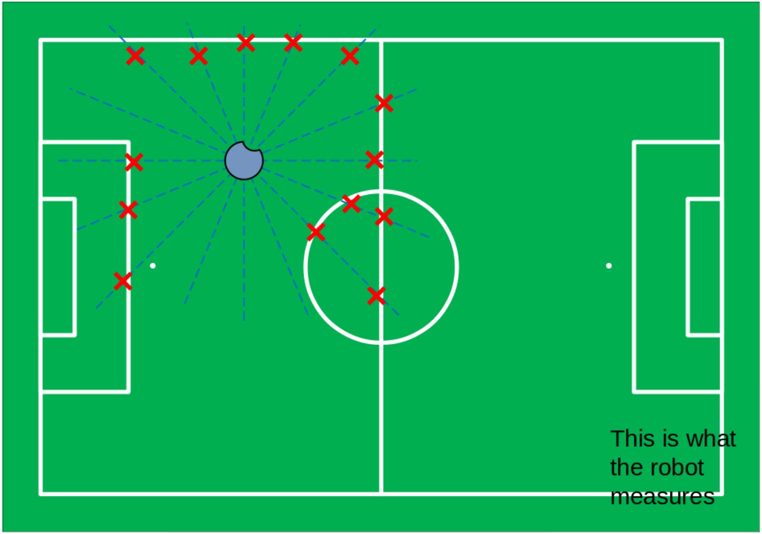


Localization in CAMBADA

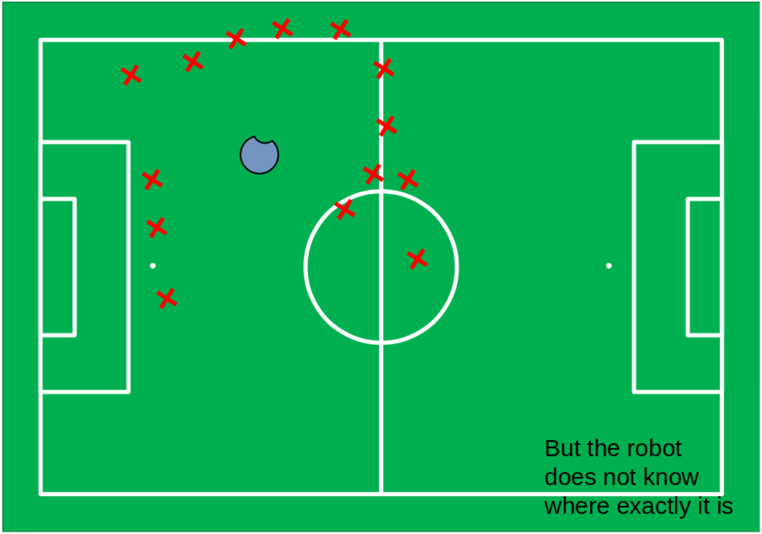
Getting visual lines, real pose



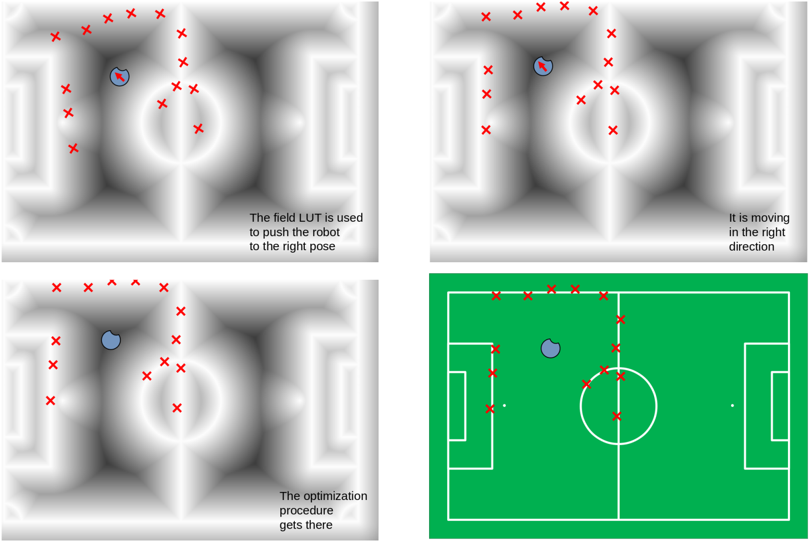
Localization in CAMBADA
Getting visual lines, real pose



Localization in CAMBADA
Lines in estimated pose



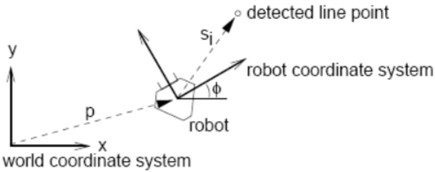
Localization in CAMBADA
Correcting pose



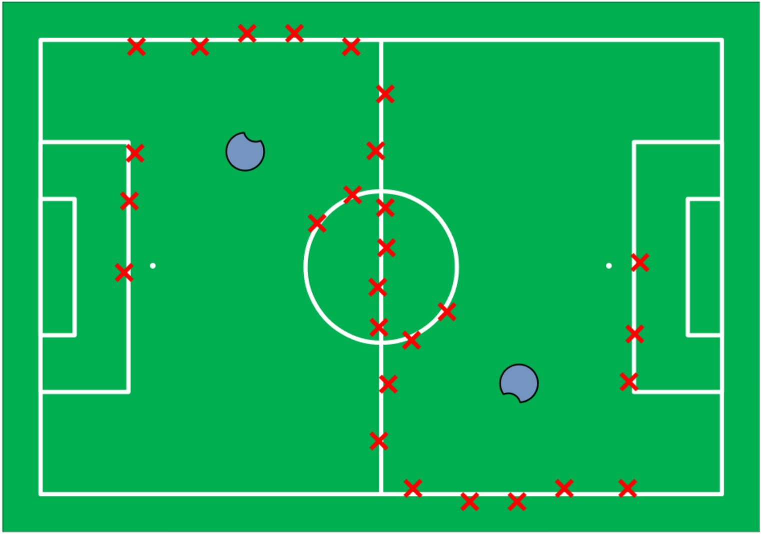
Localization in CAMBADA
Error function

$$\underset{\boldsymbol{p},\phi}{minimize} \quad E := \sum_{i=1}^{n} err(d(\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i}))$$

- p and θ are the position and heading
- s_i is the position of a detected white line
- Mapping `d()` gives the distance from a point in the field to the closest white line

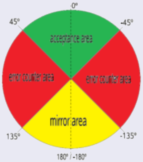


Localization in CAMBADA
Symmetric position problem



Localization in CAMBADA
Tracking

- Robot optimizes previous position (updated with odometry) and also 4 positions with:
  - fixed offsets of 60cm in xx and yy positive and negative dirs
  - small random heading offset
  - The optimized position with the smallest error is taken as the best estimate
- Detection of symmetric position
  - Compass based, if possible
    - compass divided into 4 regions



- Detection of lost condition
  - Compass based, if possible
  - Forces global localization algorithm

Localization in CAMBADA
Global localization

- A grid of trial points is used as candidate position for optimization
  - Grid spans one half of the field
  - Resolution of 1m over xx and yy
- Initial heading may be:
  - Based on compass (allows use without human intervention)
  - Fixed, ex: robot oriented towards positive xx (for fatidic fields)
- Optimized position with smallest error is chosen
- A set of 4 neighbors of smallest error position (using 40cm offsets) are still checked for better precision

Bibliography

- “Probabilistic Robotics”, Sebastian Thrun, Wolfram Burgard, Dieter Fox, MIT Press, Cambridge, Massachusetts, London England, 2005.
- “Calculating the perfect match: An efficient and accurate approach for robot self-localisation”, Martin Lauer, Sascha Lange and Martin Riedmiller, RoboCup 2005: Robot Soccer World Cup IX, LNCS.
- “The Robotics Primer”, Maja J. Mataric, The MIT Press