Localization

A robot's Navigation Problems

- Where am I? *Localization*
- Where have I been? *Map making*
- Where am I going? *Mission planning*
- What's the best way there? *Path planning*

The Localization Problem

- Detection of the position of a robot relative to its environment, using the information about the environment gathered by the robot.
- The robot typically uses its sensors to gather information about the environment.

Sensor Readings

- Types of sensor readings used in localization:
  - Vision,
  - Sonar distance readings,
  - Laser scanner distance readings,
  - GPS signals,
  - Electronic compass readings,
  - WLAN signal strength,

Pose

- In two dimensions robot's
  - *x, y coordinates and*
  - *heading*

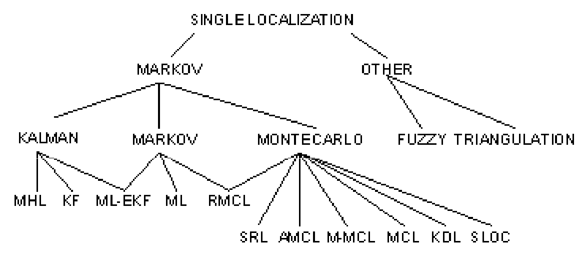
Types of Localization Problems

- **Position (pose) tracking :** Keeping track of the robot's pose using odometry, assuming that the initial position of the robot is known.
- **Global localization :** Finding its location in the environment without using *a priori* information.
- **Kidnapping :** The robot is taken from its current location and carried to another location by teleportation without the robot being aware. The robot should realize this, and relocalize itself in its new position. This is possibly the hardest problem among others.

Active vs Passive Localization

- Active Localization: The robot actively searches its environment to find landmarks to localize itself better, and decrease its localization error.
- Passive localization: The robot is busy with its task, and it runs the localization as a background task, using the observations it gets during its main task implementation.

Classification of Localization Algorithms



[Köse 2006]

Odometry

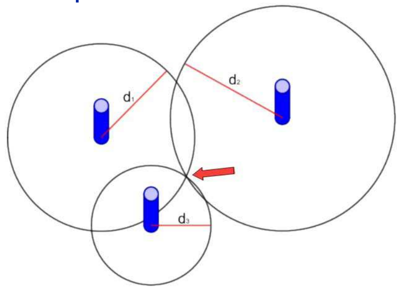
- Odometry is used by mobile robots to estimate their position relative to a starting location.
- Uses data from the rotation of wheels or legs to estimate change in position over time.
- Often very sensitive to error.
- Rapid and accurate data collection, equipment calibration, and processing are required in most cases for odometry to be used effectively.

Triangulation

- The simplest localization method
- Uses the geometric properties of triangles based on the distances and/or angle measurements between the robot and the observed landmarks.

Triangulation using distance and angle measurements

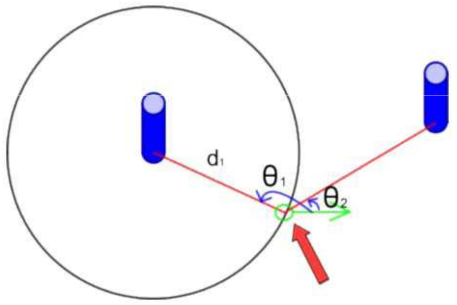
- The position of the robot is calculated using the measured distance between the robot and the multiple reference positions.
- For the position estimation in two dimensions, distance measurements from three non-collinear points are required:



[Çelik 2005]

Triangulation using distance and angle measurements (cont.)

- Two angle measurements and one distance measurement can also be used:

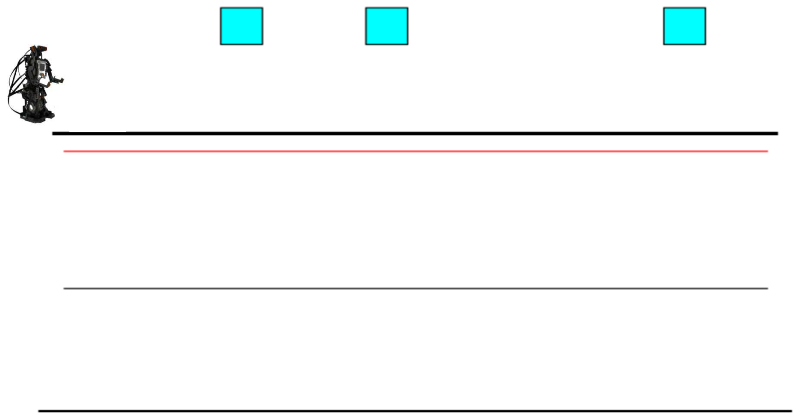


[Çelik 2005]

Probabilistic Localization

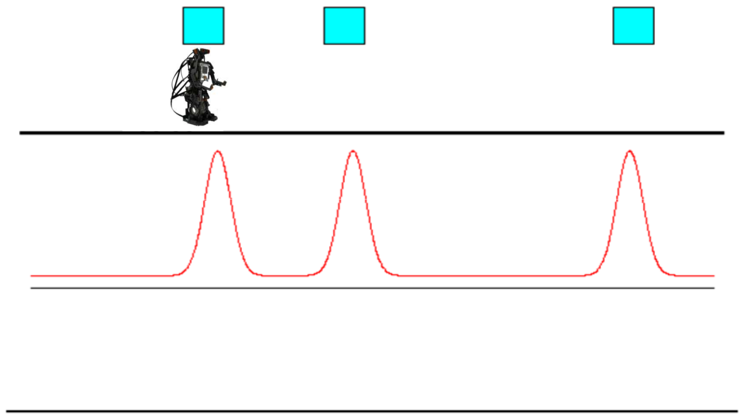
- The robot's location is represented as a probability density over possible robot locations or poses.
- **Belief state:**
  - Represents the robot's belief in its current position.
  - It summarizes all the information the robot has about its current position.
  - It is updated for movements of the robot and for information from sensors.

Probabilistic Localization Example



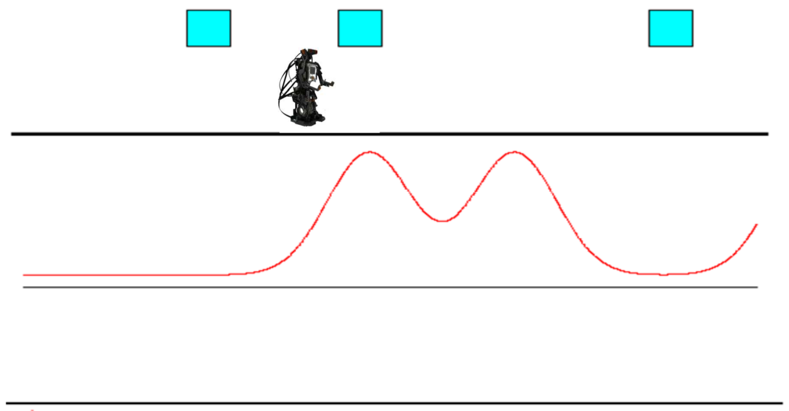
Initially, the robot does not know where it is.

Probabilistic Localization Example (cont.)



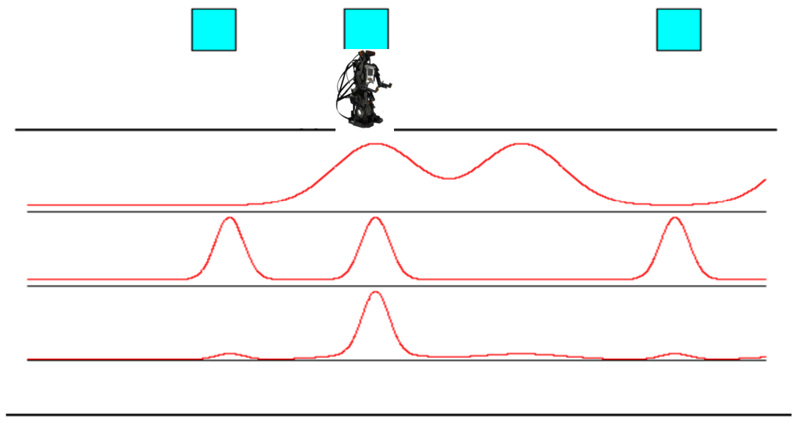
The robot observes a landmark.

Probabilistic Localization Example (cont.)



The robot moves.

Probabilistic Localization Example (cont.)



The robot observes another landmark.

Markov Localization Method

- It does not make any distribution assumptions, and allows any kind of distribution to be used.
- It uses odometry measurements and perceptional measurements to estimate the current position of the robot.
- The robot maintains a belief over its position which is denoted as *$Bel^{(t)}(L)$ at time t.*
- The ML method uses a histogram filter for posterior belief.

Markov Localization

- General Remarks
  - Coarse
  - Very robust
  - No distribution assumption so more flexible but more costly
  - High computational complexity
  - Fast recovery from kidnapping

Monte Carlo Localization

- Monte Carlo Localization (MCL) is a version of ML that relies on sample-based representation and the sampling/importance re-sampling algorithm for belief propagation
- Beliefs are represented by a set of K weighed samples (particles) which are of type *$((x, y, \theta), w)$, where $w \geq 0$* is a non-negative numerical weighting factor such that the sum of all w is one.
- *The* weighting factors are called *importance weights.*
- *Odometric and sensory updates are* similar to ML. They are performed within the prediction and correction steps.

The MCL Algorithm

```
procedure MCL(S, a, o)
1: for i = 0 to n do
2:    draw by replacement random sample l from S
      according to $w_1..w_n$ (resampling)
3:    draw sample l p(l|a, l)(sampling)
4:    calculate w=p(l|o) (importance sampling)
5:    add (l, w) to S
6: end for
7: normalize the importance factors w in S
8: return S
```

The MCL Algorithm

- MCL represents the probability distribution of location *l* by a sample set.
- The samples are located such that, the population size of samples is proportional to the probability of the robot being in that location and the surroundings.
- After several steps, the samples converge to some place in the environment.
  - The position of the robot is found by taking the average of the positions of the samples, and
  - The standard deviation gives the uncertainty of the robot.

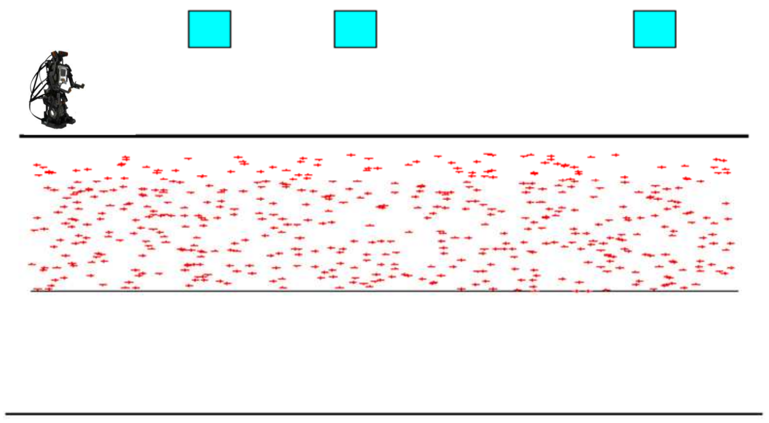
MCL and Kidnapping

- The original MCL algorithm suffers in case of kidnapping.
- Several MCL extensions were proposed to overcome this problem.
- These extensions add new samples to the sample set from different parts of the field, and differ in the number of new samples and when to add these samples.
- Some of these methods are
  - Sensor Resetting Localization (SRL),
  - Mixture MCL (Mix-MCL),
  - Adaptive MCL (A-MCL), and
  - Kullback-Leibler Distance (KLD)-Sampling

Monte Carlo Localization

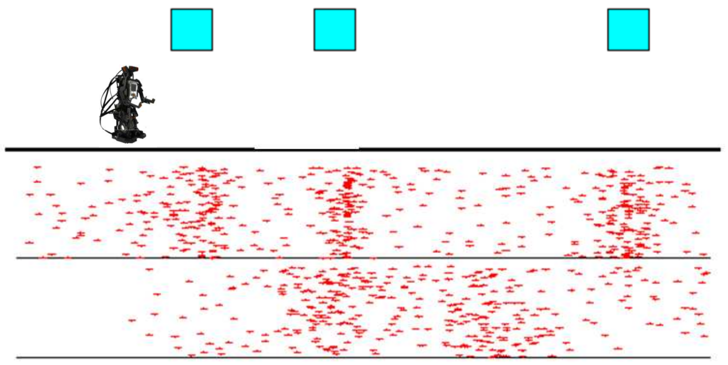
- General Remarks
  - Fails in case of kidnapping
  - Accuracy and robustness depend on number of samples
  - High number of samples ->less bias
  - Less computational complexity than ML
  - Less memory consumption than ML->depends on the number of samples
  - Not as fast as ML in convergence

MCL Steps



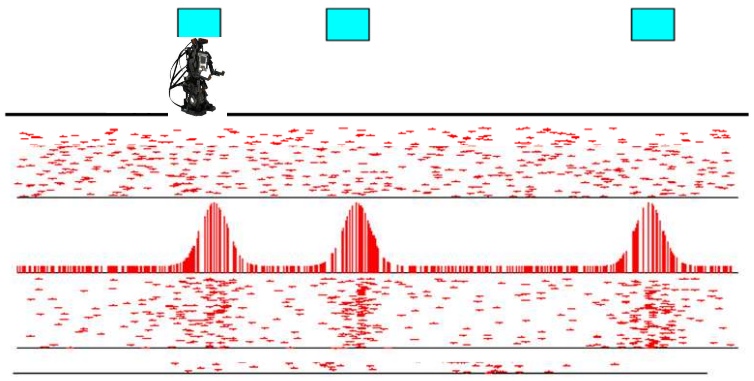
The robot is initially lost

MCL Steps (cont.)



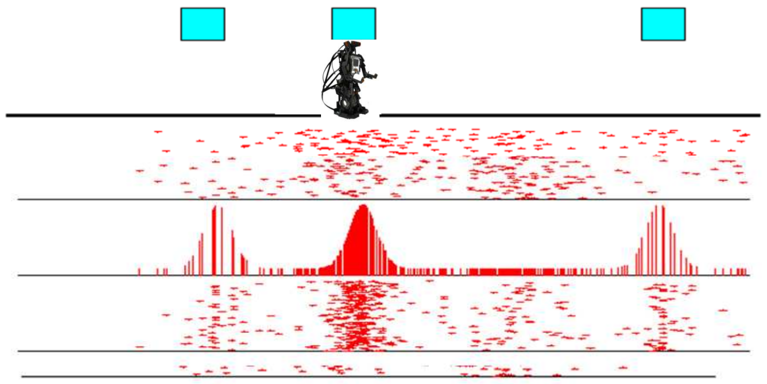
The robot moves to right.

MCL Steps (cont.)



The robot observes a landmark.

MCL Steps (cont.)

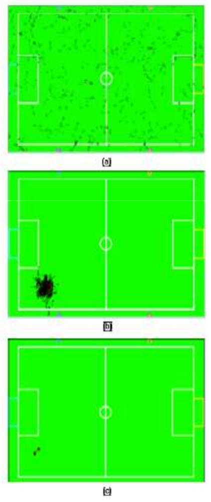


The robot keeps moving to the right, observes another landmark.

Robocup Aibo Field Setup

Robocup Nao Field Setup

MCL Application



[Kaplan et al 2005]

Kalman Filter Method

- It implements belief computation for continuous states.
- It is similar to Markov Localization.
- This filter integrates uncertainty into computations by making the assumption of Gaussian distributions to represent all densities including positions, odometric and sensory measurements.
- Position estimates are updated by odometry and sensing alternately using the property that Gaussian distributions can be combined using multiplication.

R-MCL (Reverse MCL) method

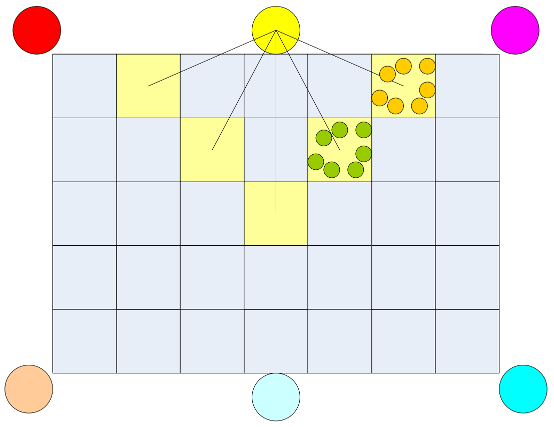
- ML is a robust grid based method.
  - Due to large grid cell sizes it is not very accurate.
- MCL is not so robust and converges more slowly than ML
  - Too many samples are used to get accurate result-expensive
- Why not combine these two methods?

R-MCL Algorithm

Algorithm 1 The R-MCL Algorithm

```
1: if $bool\_ML==TRUE$ then
2:    $ML\_update$
3:    if $ML\_number\_of\_grid\_cells\_in\_max\_grid\_array < Th_{ML}$ then
4:       $MCL\_init(ML\_samples)$
5:       $bool\_ML=FALSE$
6:    end if
7: else
8:    $MCL\_update$
9:    $MCL\_init(ML\_samples)$
10:   if $MCL\_lost()==TRUE$ then
11:      $ML\_reset()$
12:      $bool\_ML=TRUE$
13:   end if
14: end if
```

R-MCL



[Köse 2006]