Cognitive Robotics

World Models

Hans-Dieter Burkhard
June 2014

Overview

- **Introduction**
- Representations of Environments
- Maps
- Controversies about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Models of Environment: World Model

Robots interact with the environment

based on their belief about the actual world.

The recently available sensor data are sometimes sufficient only for “simple” tasks in “simple environments” by appropriately designed robots (cf. situated robots),

e.g. line following, obstacle avoiding

More complex tasks/environments need memory for

integrating information over time/from different sensors:

Worldmodel as description of actual situation.

Situation in Environment

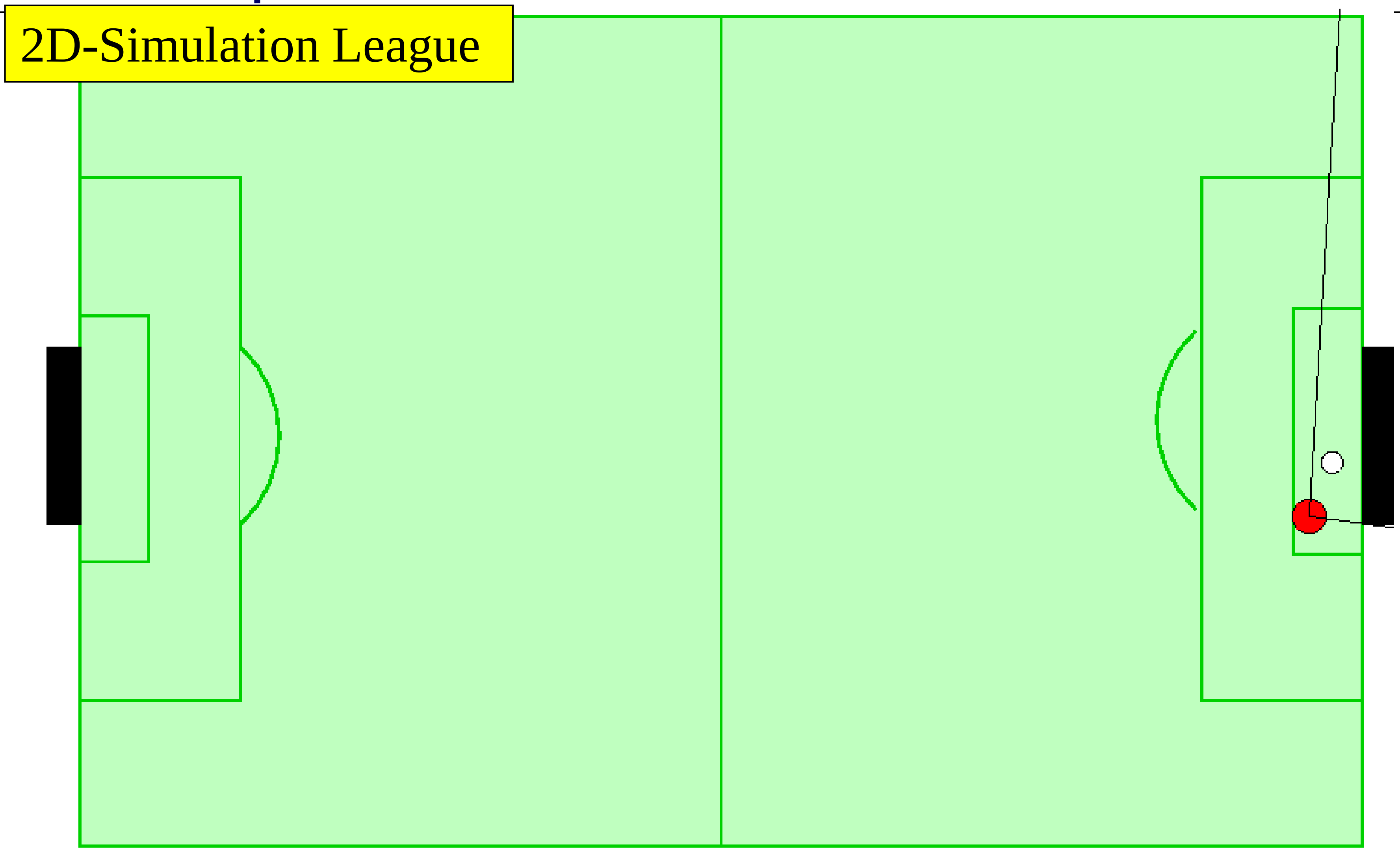
- Own pose and motion (self localization)
- Poses and motions of other objects
- Free space

To be used for

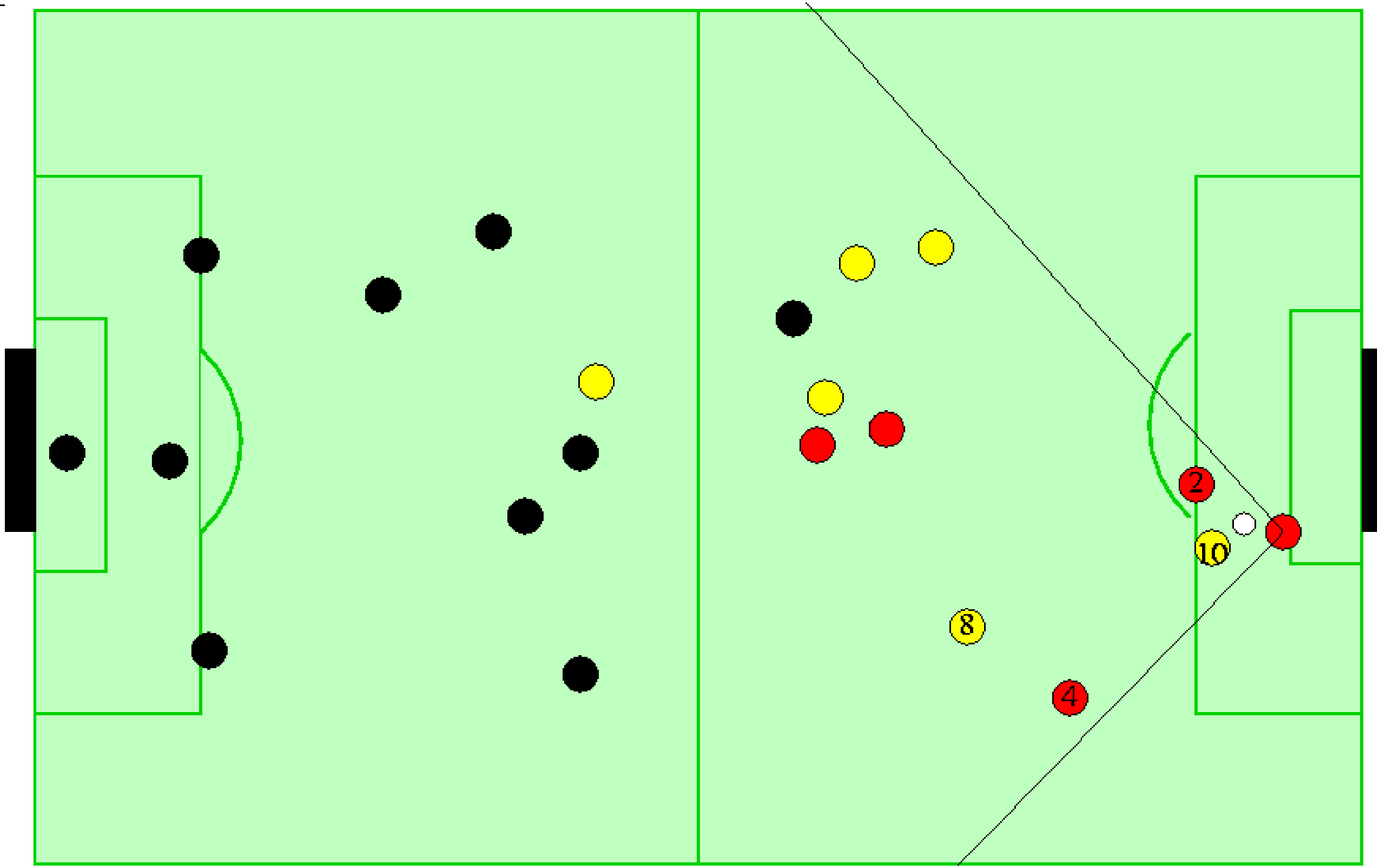
- Navigation
- Path planning
- Mapping (exploration and mapping of the environment)

SLAM = Simultaneous Localization And Mapping

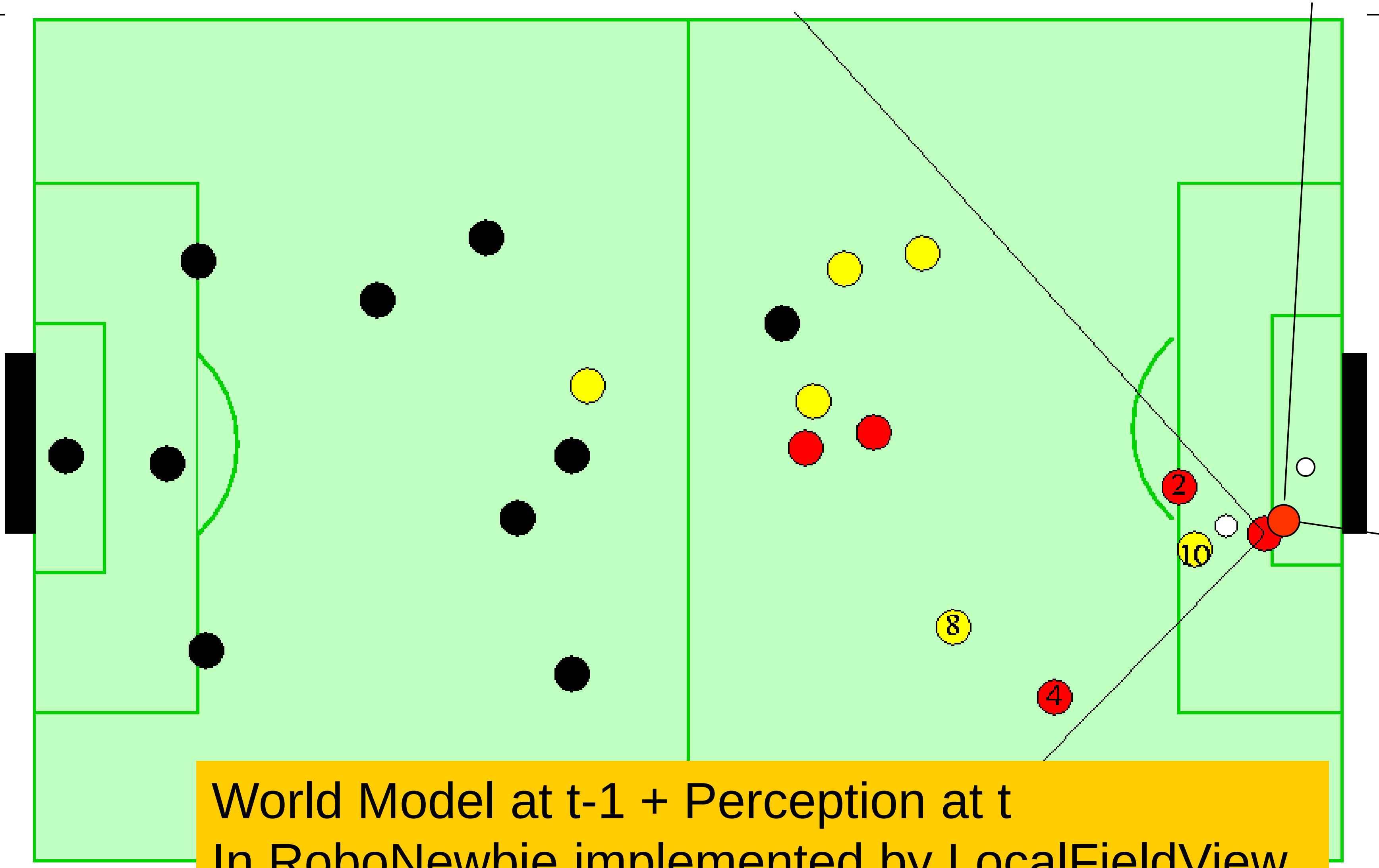
Example: Restricted Observation at Time t



Example: Memory of past situation at Time t-1

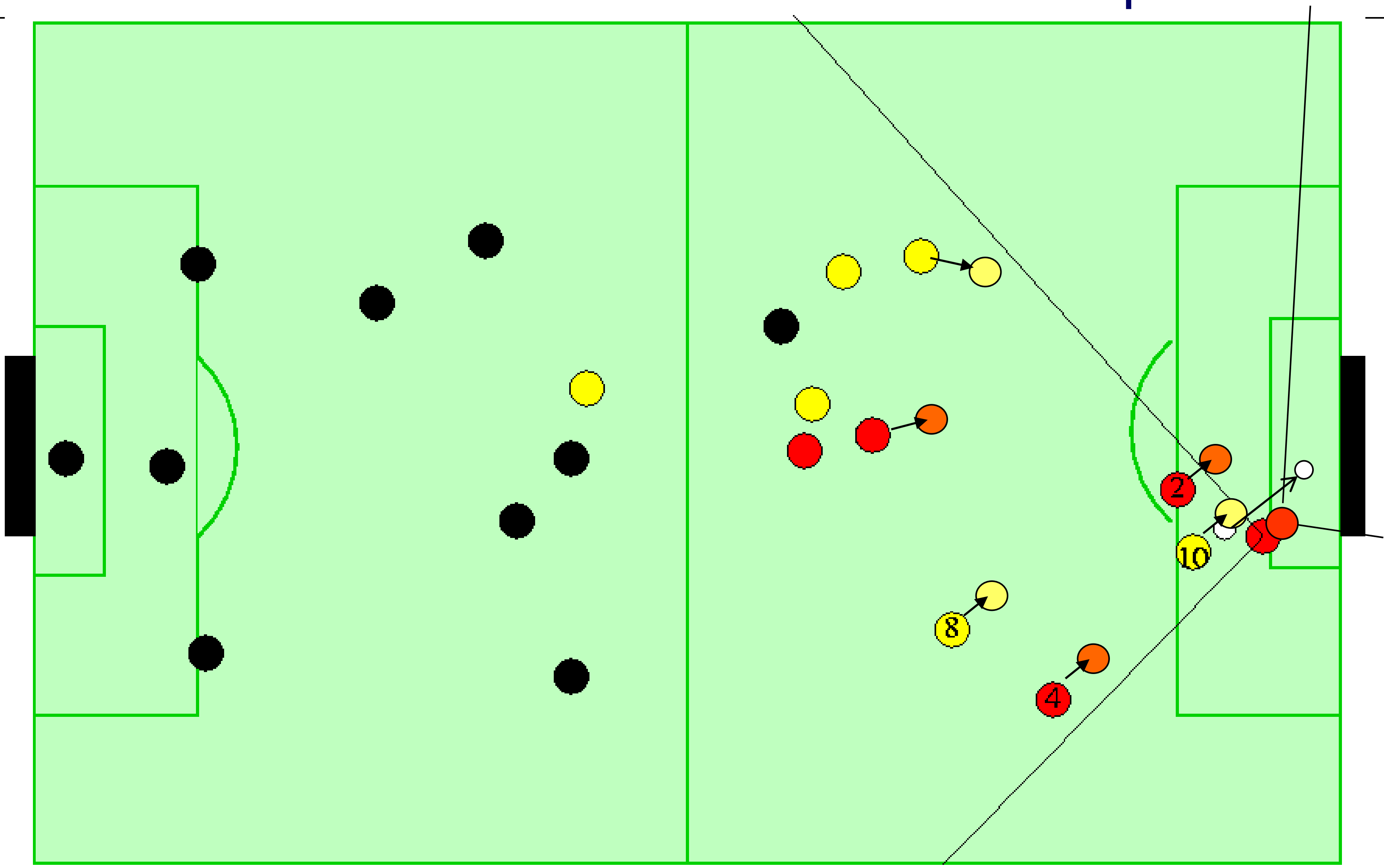


Combination: World Model at Time t



World Model at t-1 + Perception at t
In RoboNewbie implemented by LocalFieldView

Combination: World Model at Time t with Expectations



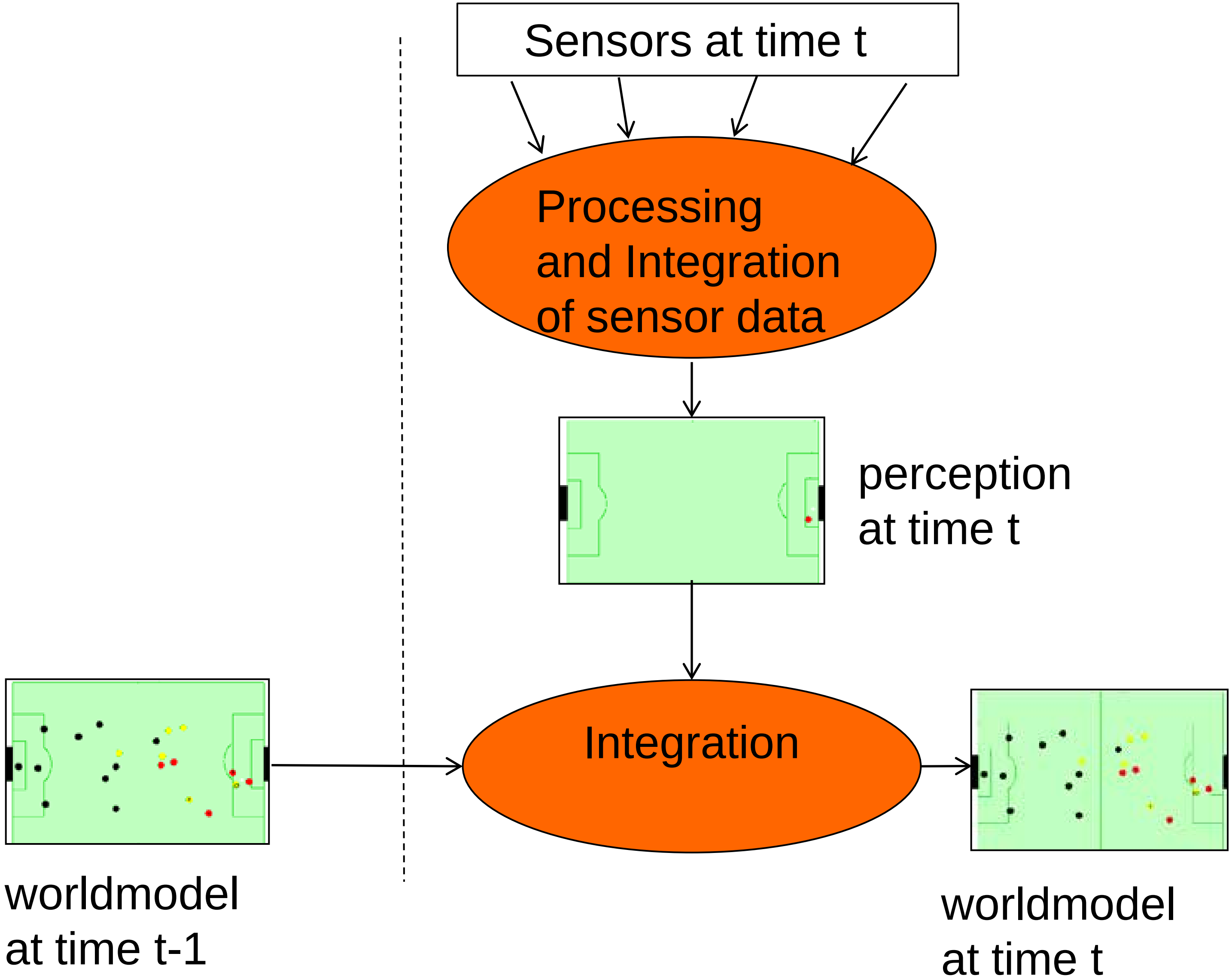
World Model: “**Belief**”

Internal representation of recent environment by given data structures for

- Space
  Objects, poses, ...
- Movements
  Direction, speed, ...
- Actors
  Goals, actions...
- Sociale relations
  Roles, duties ...

„Belief“ instead of „knowledge“:
World model may be incorrect!
(Noisy/incomplete sensor data.)

Update Cycle of World Model



Discrete Modeling: States – Actions

Discrete time points t

x_t : state at time t

u_t : action at time t

z_t : observation (sensor data, percepts …) at time t

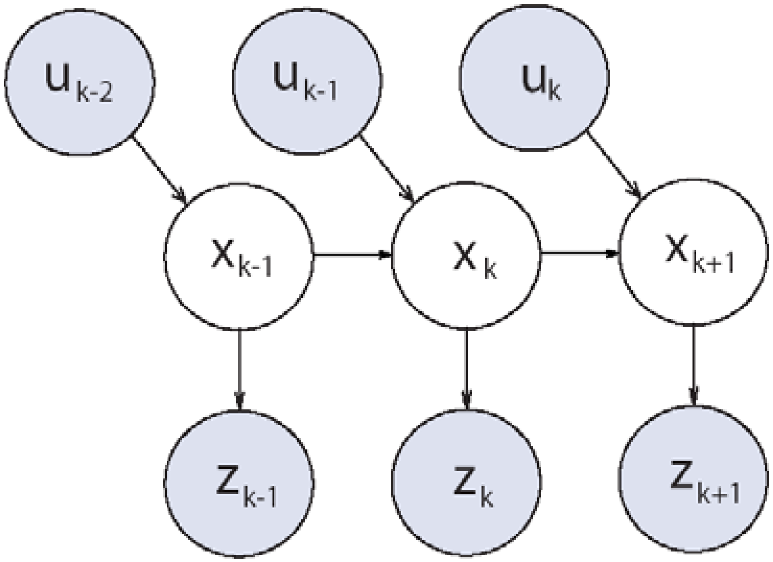
- Transition of states by actions.
- $x_{t+1} = f(x_1, x_2, x_3, \ldots, x_t, u_1, u_2, u_3, \ldots, u_t)$
  (deterministic or stochastic)

- Markov Condition: $x_{t+1} = f(x_t, u_t)$
  (needs appropriately complete descriptions)

Stochastic World

- Uncertainty with respect to states
- Uncertainty regarding results of actions
- Uncertainty regarding observations

Probability distributions for:
x_i state at time i
u_i action at time i
z_i observation at time i



Bayesian Net
(with Markov Condition)

Properties of Environments

- Structure: structured vs. chaotic
- Scaling: discrete vs. continuous
- Dynamics: rapidly changing vs. static
- Definiteness: deterministic/non-determistic/stochastic
- Repeatability: episodic vs. constantly changing
- Observability:
  - complete vs. partial
  - correct vs. uncertain
- Influenceability:
  - complete vs. partial
  - effective vs.ineffective

Properties
may appear differently to the robot.

They depend on
- *Available ressources:*
  „Bounded Rationality“
- *Design decisions:*
  Data structures

Overview

- Introduction
- **Representations of Environments**
- Maps
- Controversies about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Formalisms for Representation

Quantitative:

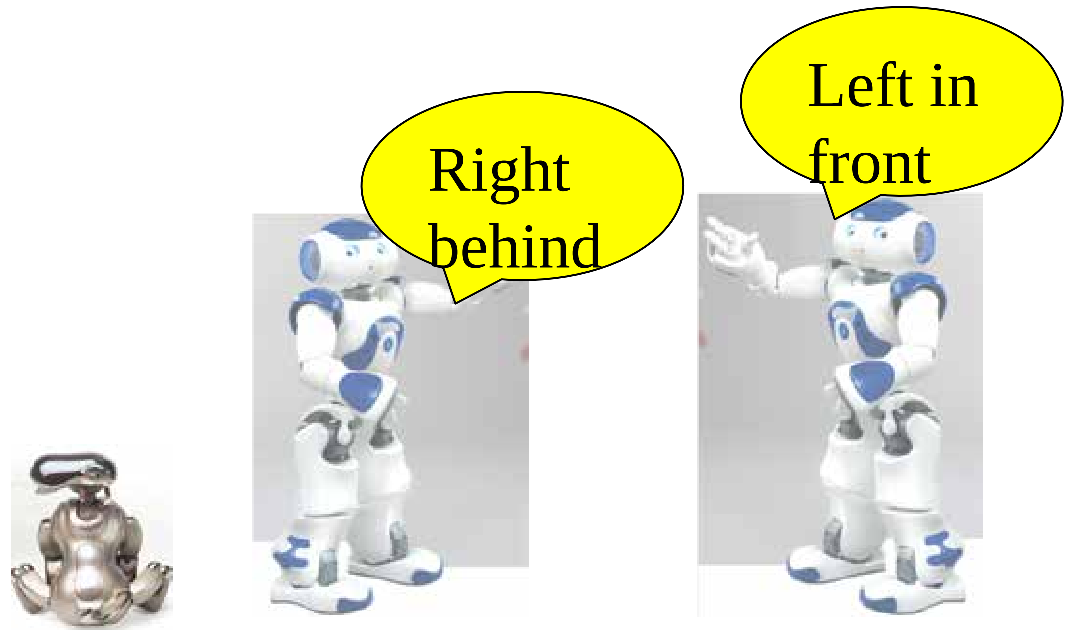
e.g. Cartesian coordinates (x, y), Polar coordinates (r, f)

needs treatment of similarity

Qualitative:

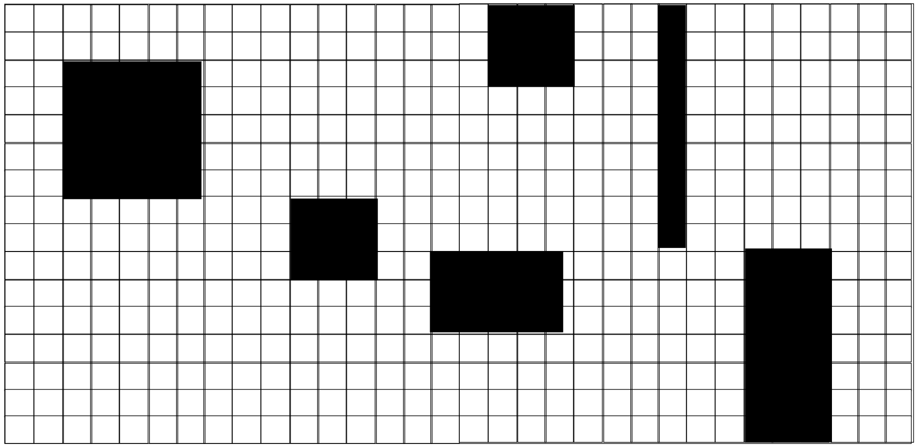
e.g. "right in front”

depends on observer

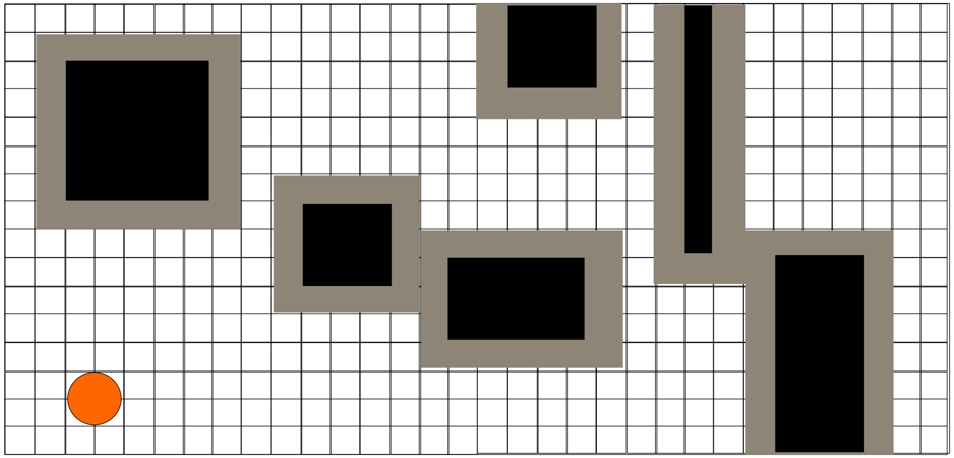


Gridbased Representation

Occupancy Grid

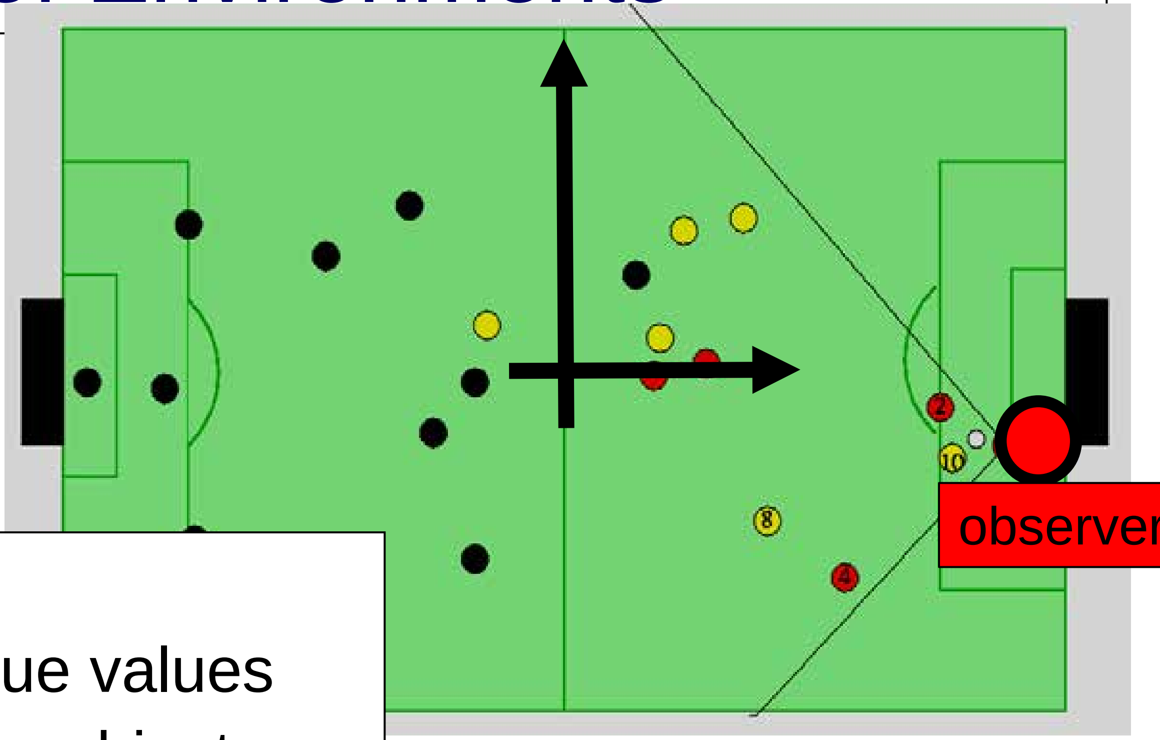


Free space can be described according to size of robot



Potential field methods can be used for path planning.

Representation of Environments



Allocentric

Absolute values
in common system

Advantages:

- Fixed objects with unique values
- Updates only for moving objects
- Common system for all robots

Disadvantages:

- Needs own position
- Inconsistencies by
  error propagation from localization

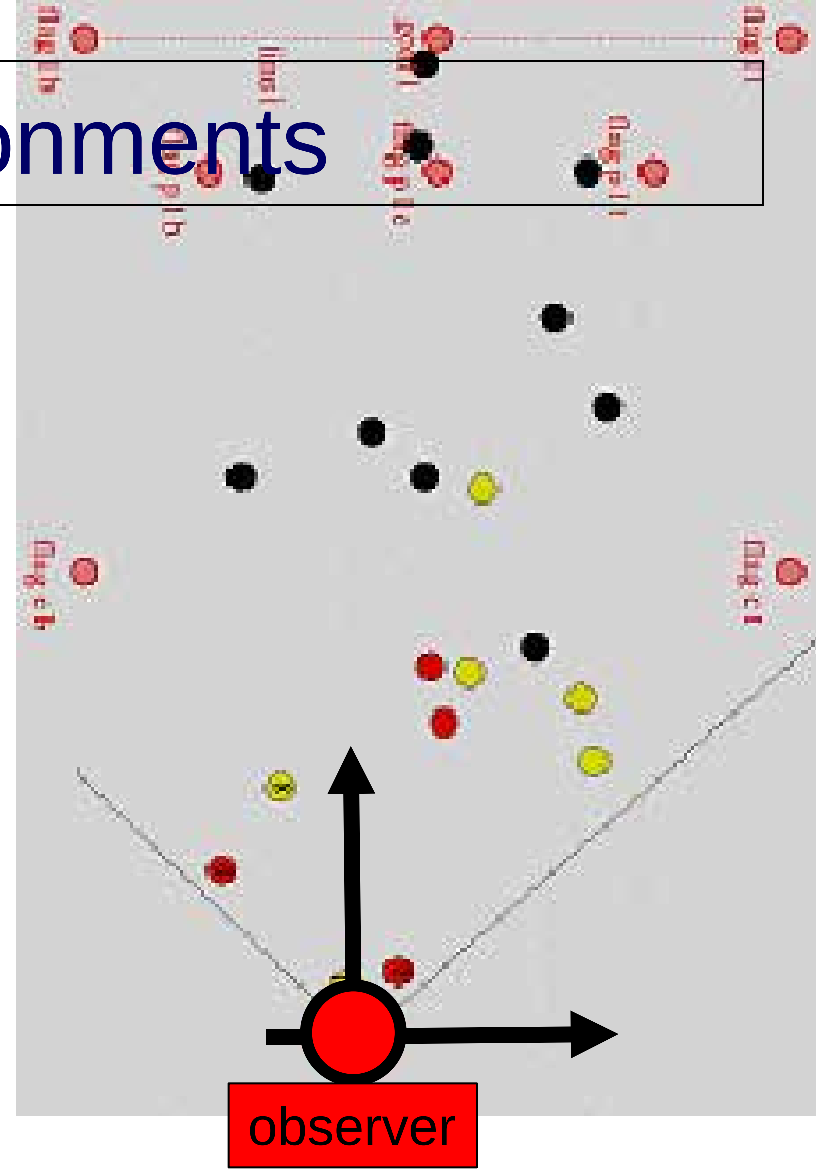
Representation of Environments

Egozentric

Information relative to own position

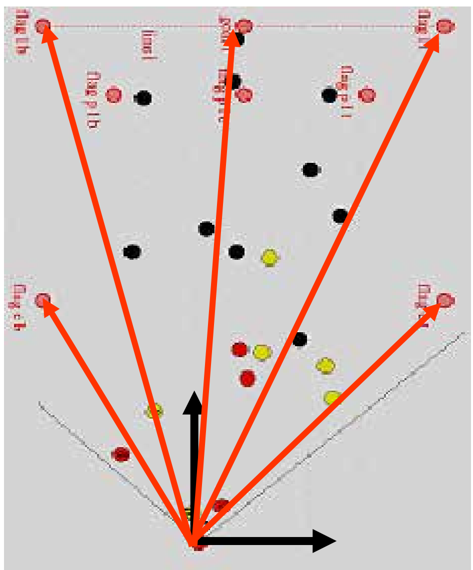
Advantage:
- Consistent with recent perception

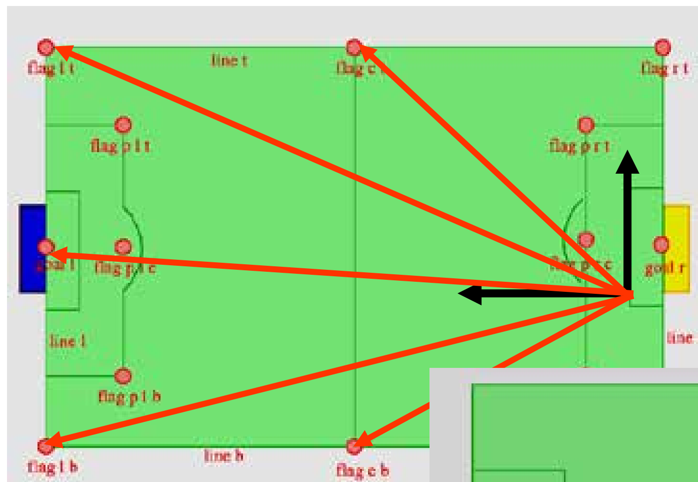
Disadvantage:
- Needs updates of all objects

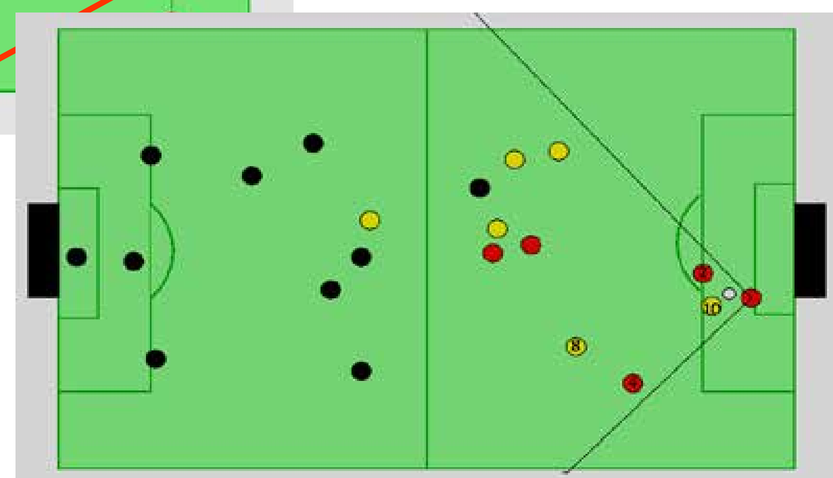


Transformations

Transformations between egocentric and allocentric model using own position in the allocentric model.







Transformations

Information “relative to **own position”**

Egozentric models may have different reference points, e.g.

- Pose of sensor (e.g. camera)
- Pose of robot (torso, foot point, …)

Transformations between coordinates by cinematic chain.

Coordinates in an image are relative to the image,

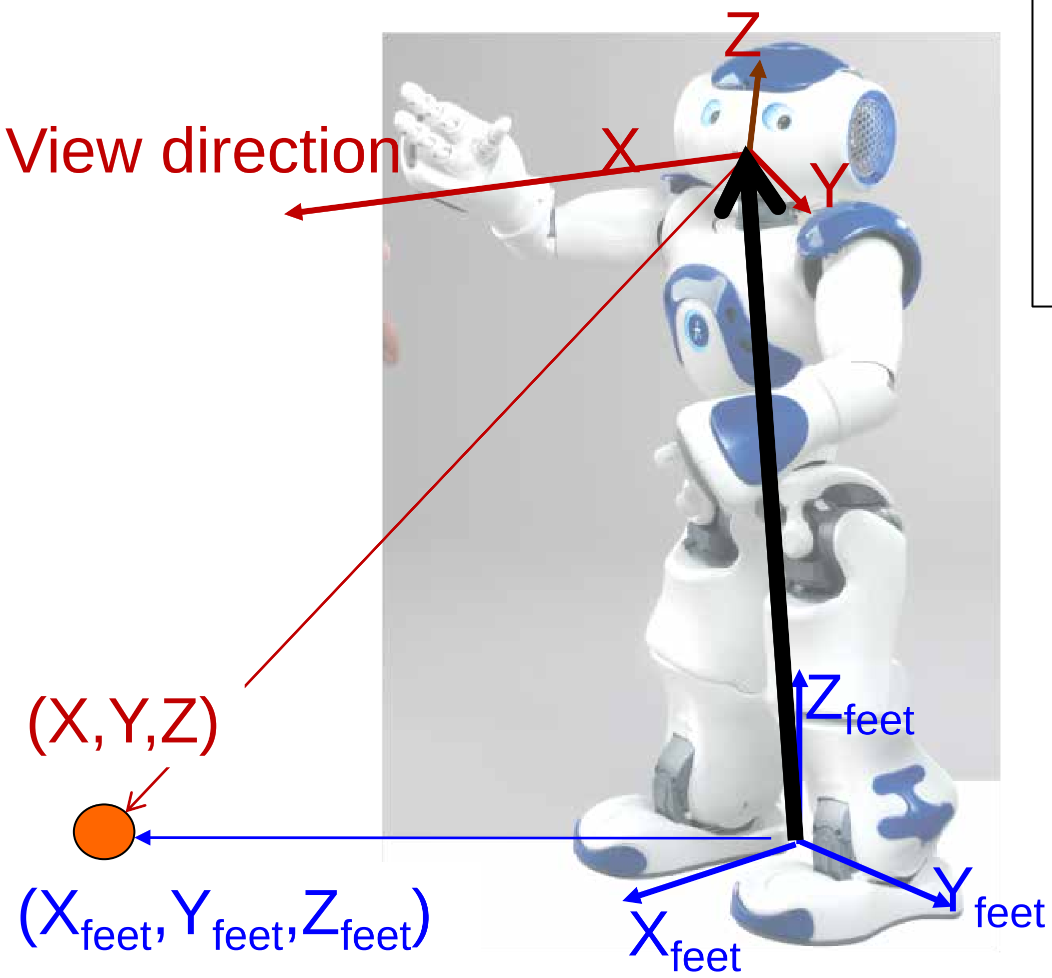
i.e. transformation determined by projection from camera coordinates to image plane.

Ball distance in RoboNewbie class LocalFielView

is distance from camera – not from foot!

Camera height is 54 cm over ground, ball radius is 4,2cm .

Transformations

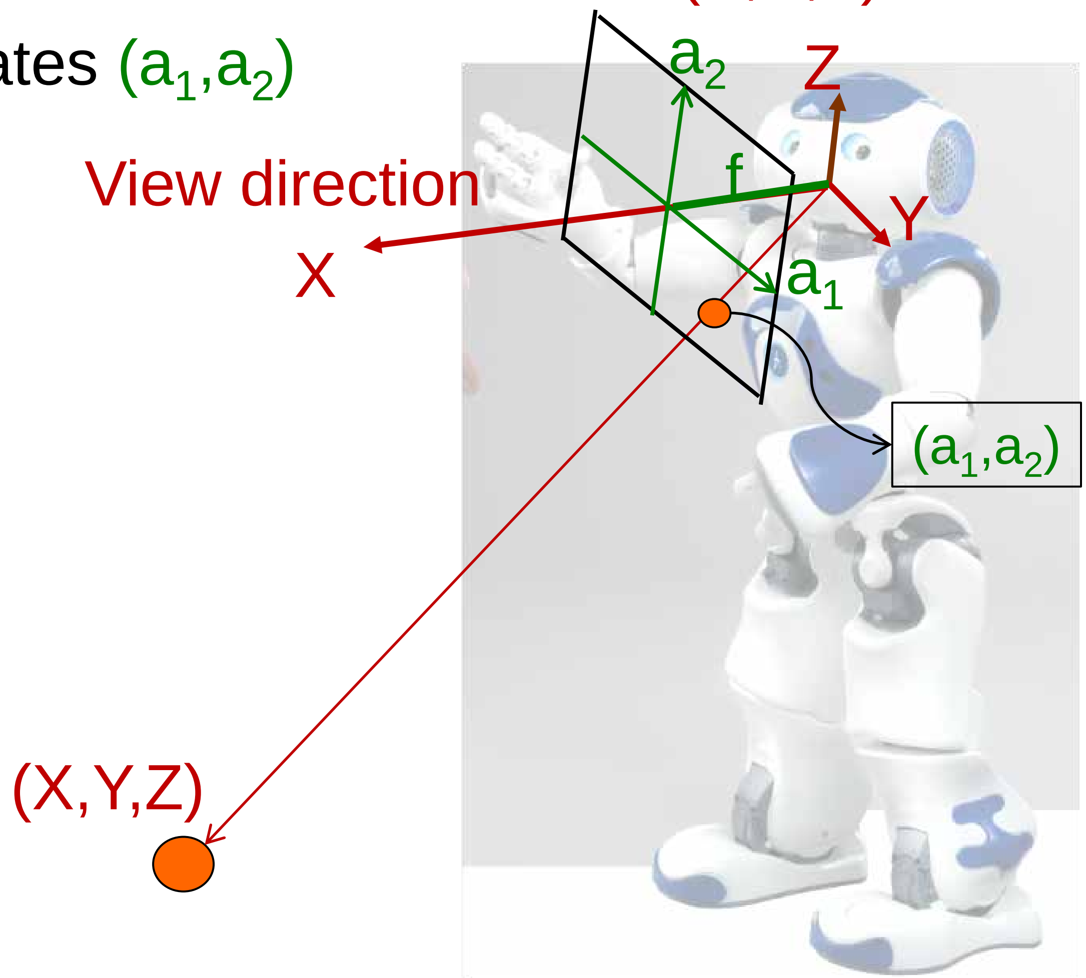


Transformation between coordinates of foot point and camera coordinates by translation to focal point and rotations

Calculation
by cinematic chain:
- translation to neck
- rotation Neck Yaw
- rotation Neck Pitch
- translation to focal point

Projection

Transformation from camera coordinates (X,Y,Z) to image coordinates (a_1,a_2) by projection:



$$a_1 = (f/X)\,Y$$

$$a_2 = (f/X)\,Z$$

Problems with Inconsistent Informations/Models

Problems with different models: Example

Robot beliefs to perceive the goal right in front.

By calculation of own position (e.g. using odometry), the robot beliefs the goal to be left of the robot.

Usually,
it is not clear which belief is correct.

Overview

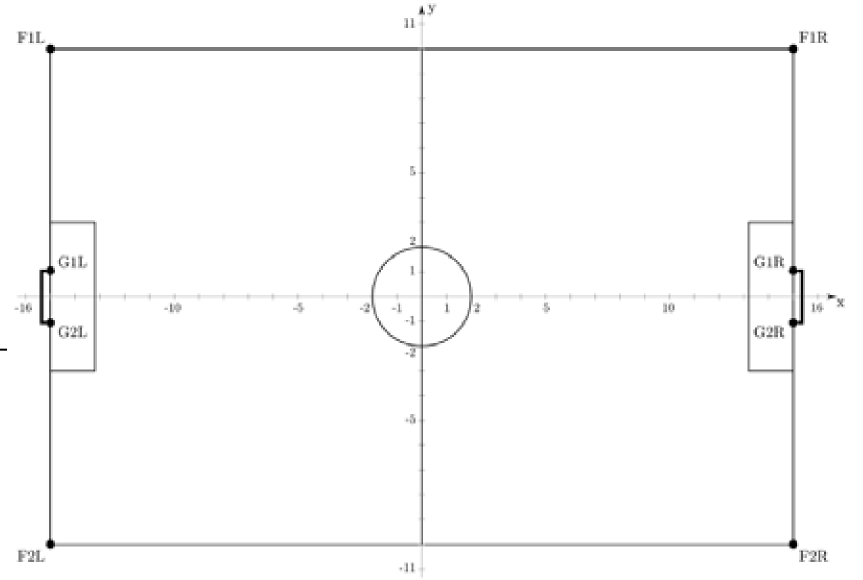
- Introduction
- Representations of Environments
- **Maps**
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Landmarks and Maps

- „Natural landmarks“
- „Artificial landmarks“ with specific coding
- „Active landmarks“ send signals

Knowledge about landmarks by maps

- Position
- Size
- Appearance ...



Map Related Problems

Correspondence problems:

- Matching of sensor data with objects in the map
- Tracking of objects:

  Matching of objects at consecutive time points

Self-Localization: Own pose relative to a given map

- by matching of observed objects with map
- by dead reckoning

Dead Reckoning

Consecutive poses are estimated by

- Initial pose and
- Actions or sensory data

  e.g. inertial system, odometry, …

Problems:

- Drift problems
- „Kidnapped Robot Problem“:

  Dead Reckoning not useful if robot has lost its position.

SLAM

= Simultaneous Localization and Mapping

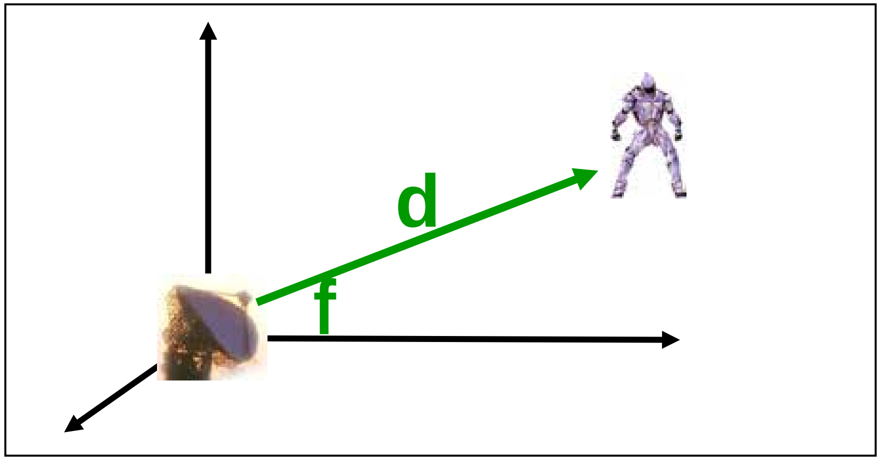
Autonomous Map Building

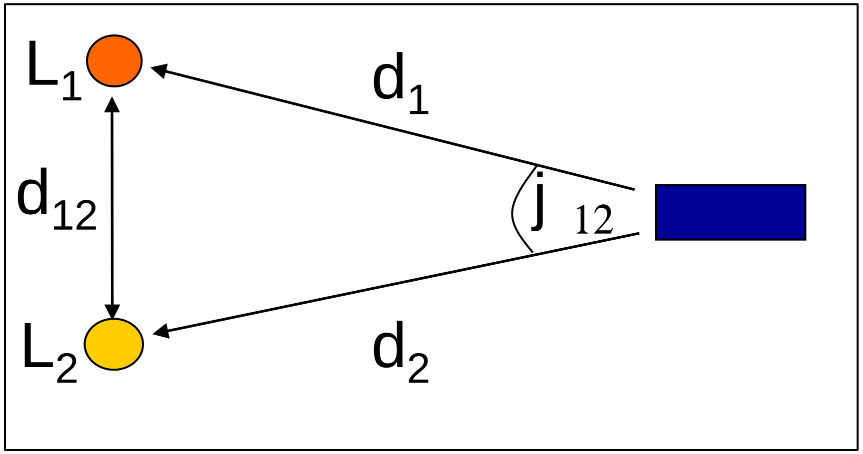
- Local environment of observer, e.g.
  - positions of recently seen landmarks relatively to observer
  - local occupancy grid
- Global map using:
  - dead reckoning
  - landmarks recognized from different poses

Geometrical Methods for (Self)Localization

Useful Data

- Landmarks L_i with known poses/positions
- Distance d_{12} between landmarks
- Angle f_i in which robot sees L_i
- Angle j_{12} between landmarks from robot position
- Distance d_i robot/landmark





Trigonometry, Trilateration

Calculations by triangles

- Trilateration: distances
- Triangulation: angles

Theorems on congruences

Sine-, Cosine-, Tangens-Formulas

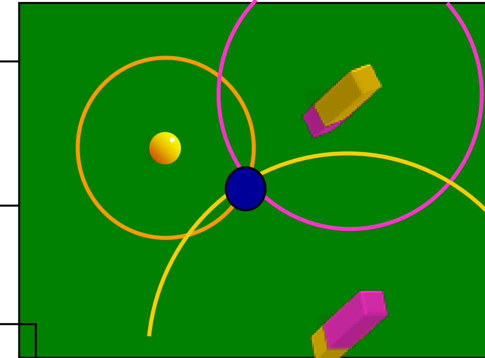
Problems

- Erroneous Measurements: distances, angles,…
- Wrong landmarks (missing ones, ghost images …)
- Bad use of formulas (e.g. sine around 0)

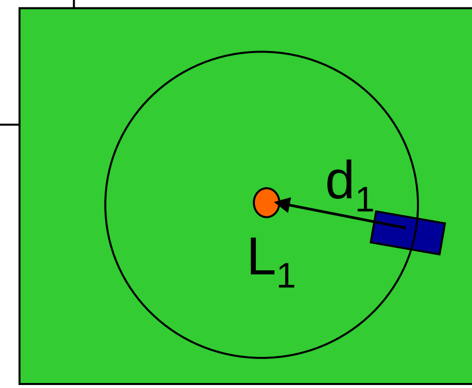
Exploitation of Redundancies:

Integration of data from different sources

Position by Intersecting Cycles

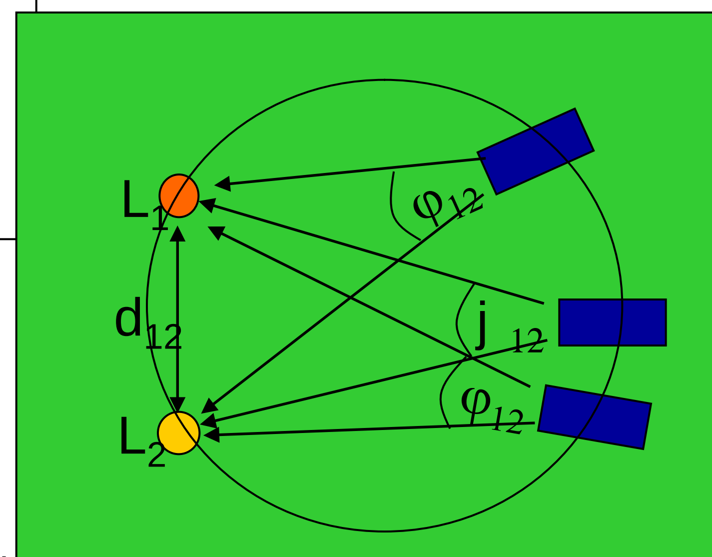


Robot position on circle with distance d_i around landmark L_i



Robot position on circle with angle j_{12} for two landmarks

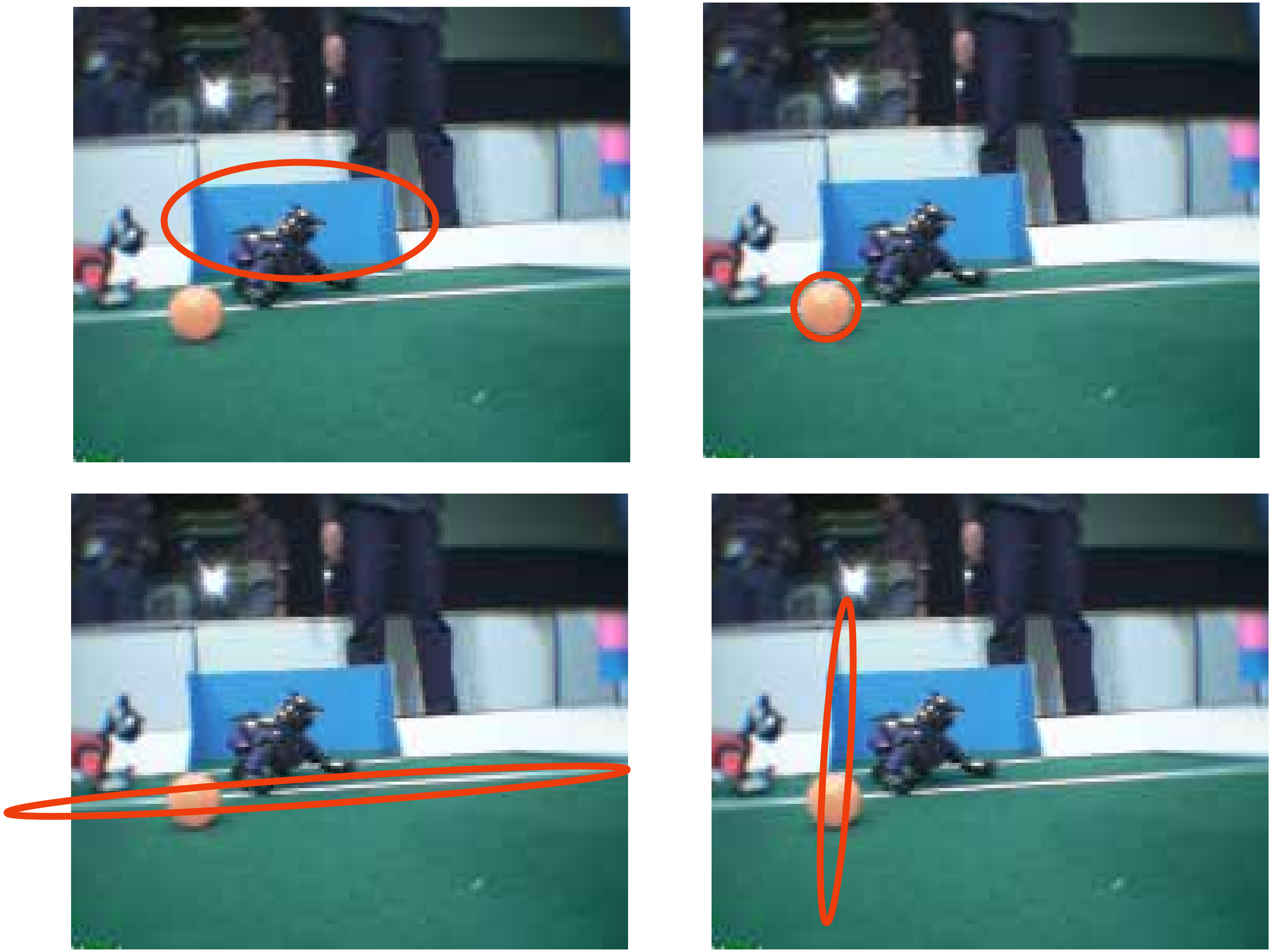
(„Peripherie Angle Theorem“)



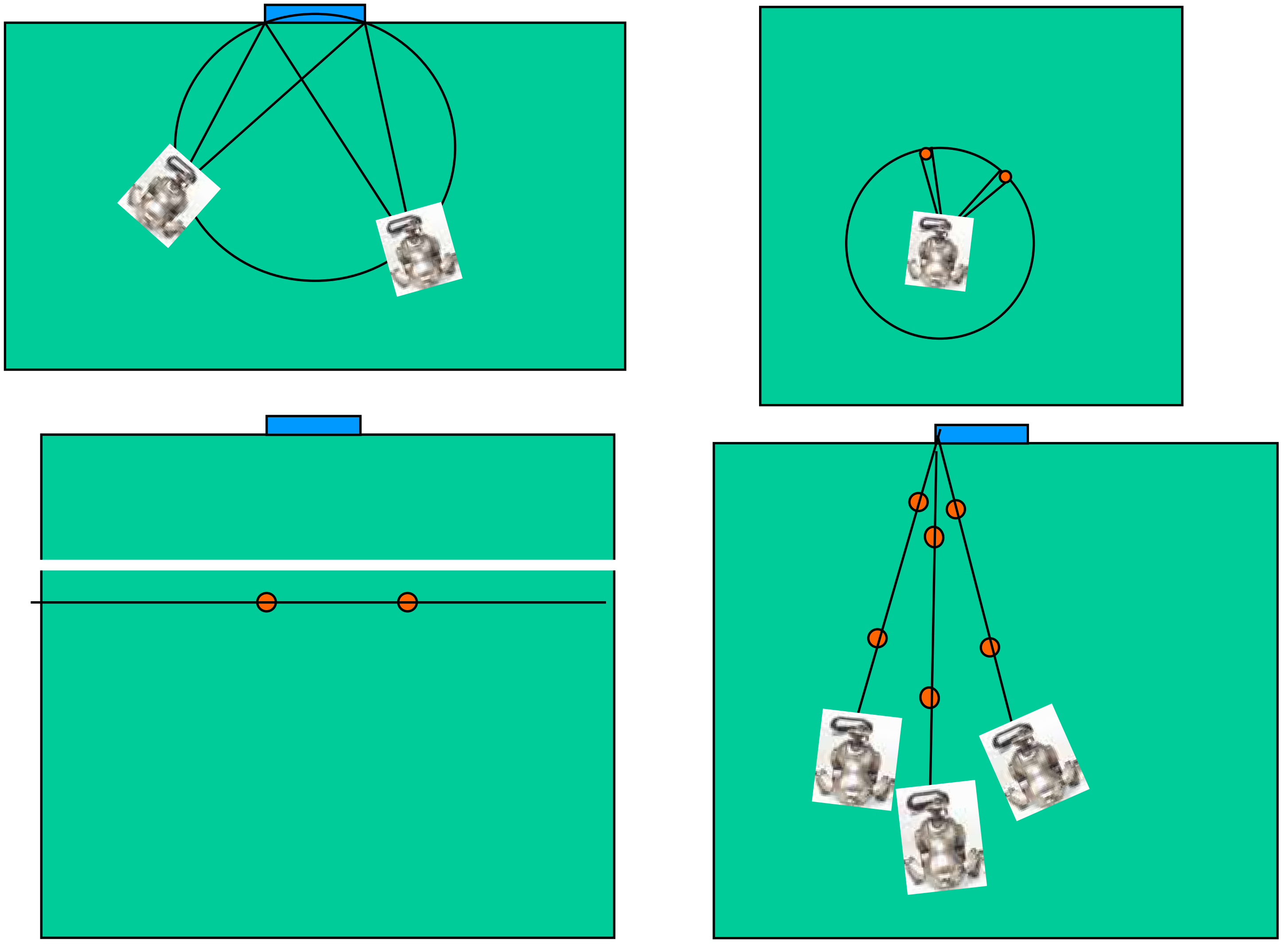
Constraints by Geometrical Relations

Where am I ?
Where is the ball ?

Constraints by Geometrical Relations

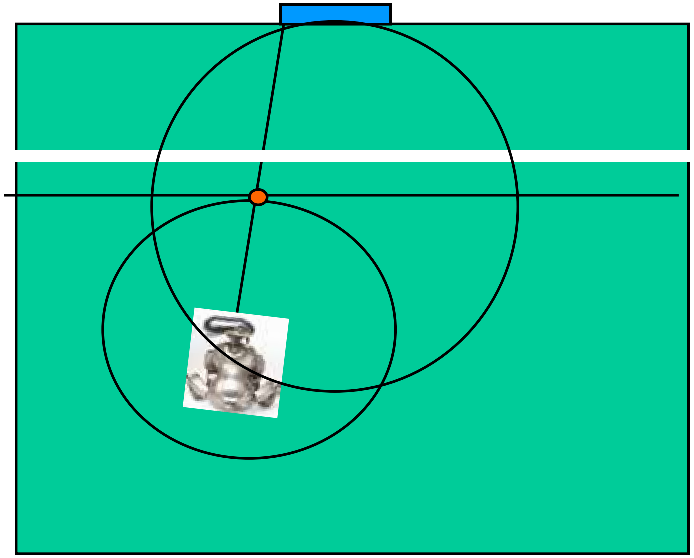


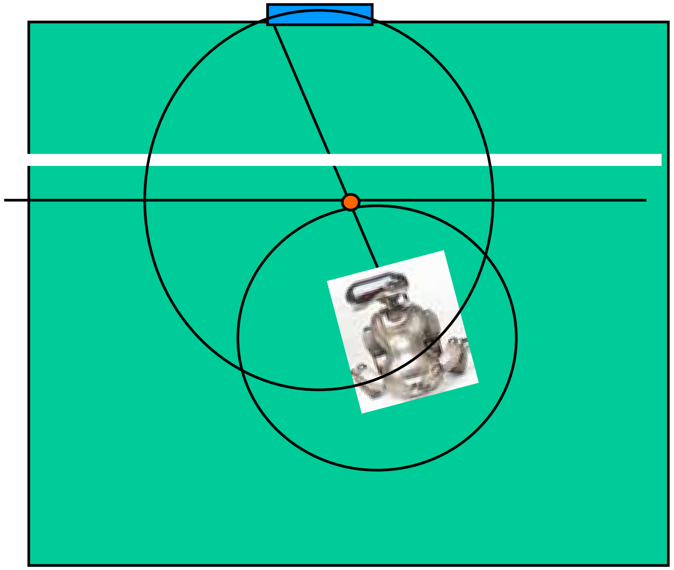
Constraints by Geometrical Relations



Constraints by Geometrical Relations

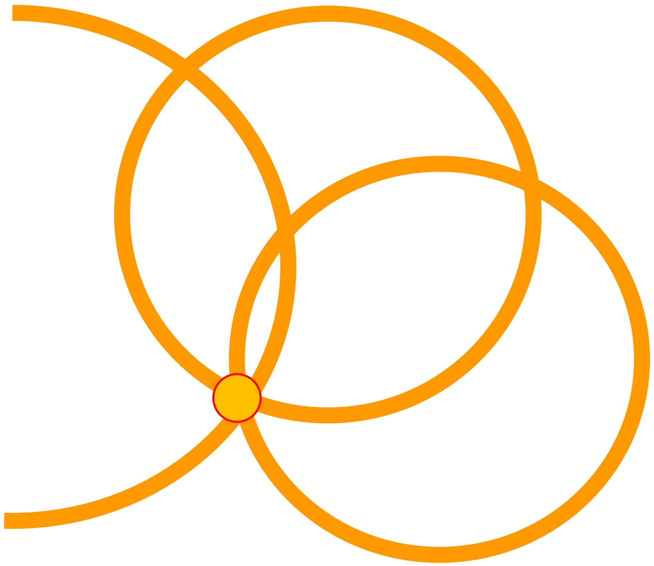
Combination yields 2 possible positions

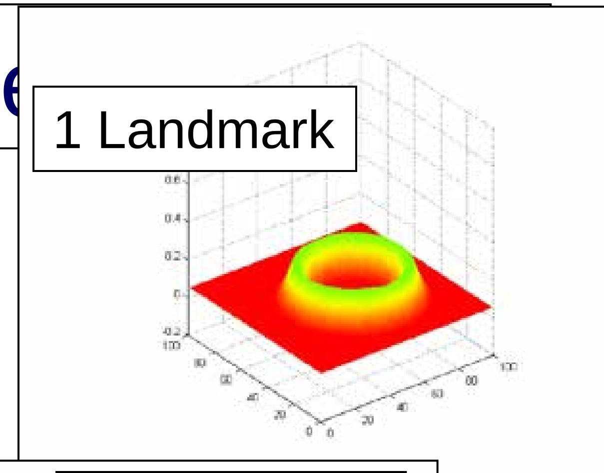


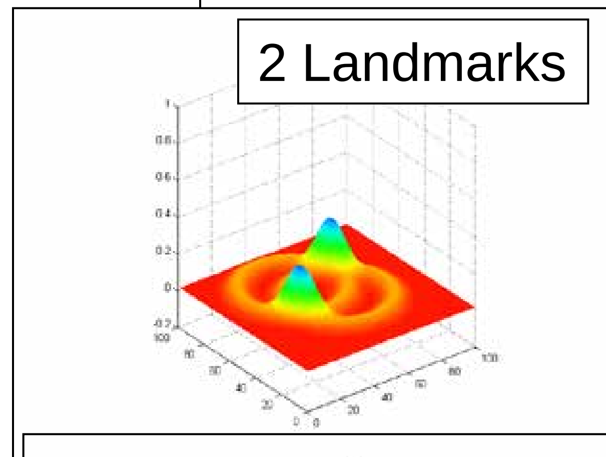


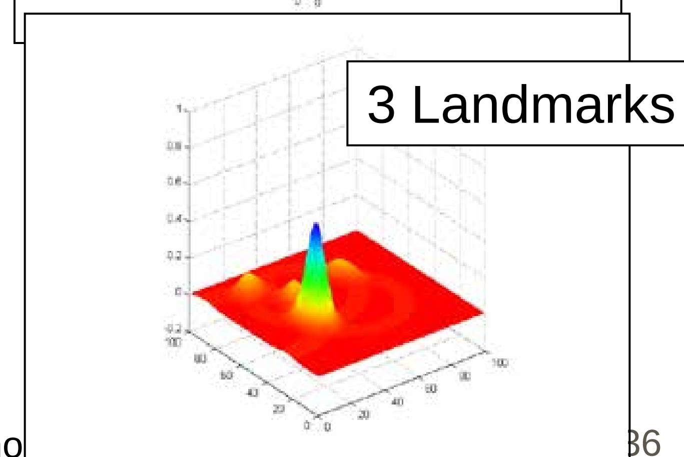
Constraints by Geometrical Re

Because of noise, constraints are fuzzy



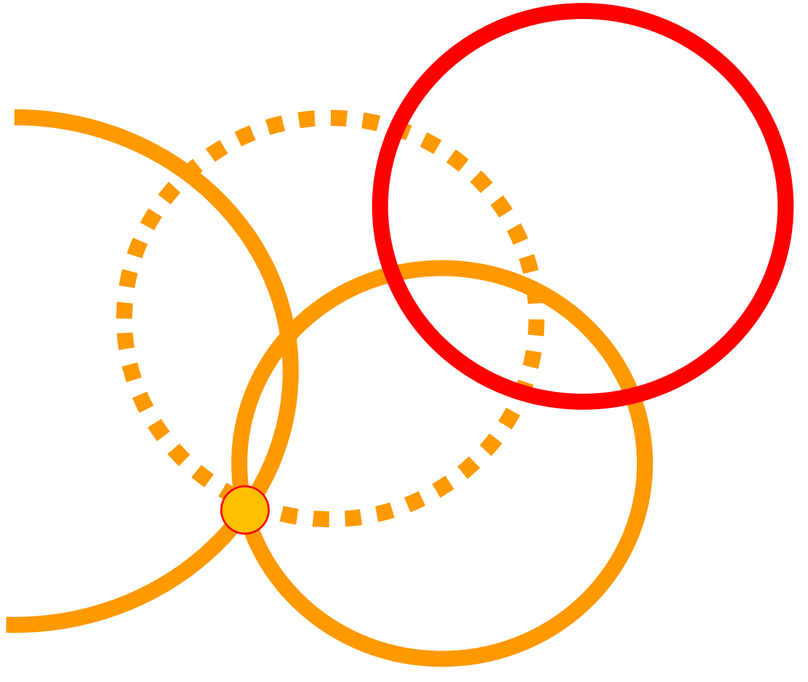


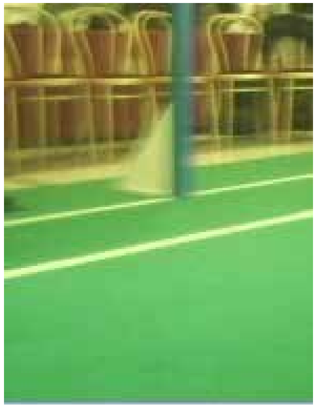




Constraints by Geometrical Relations

Wrong landmarks/odometry may lead to inconsistencies





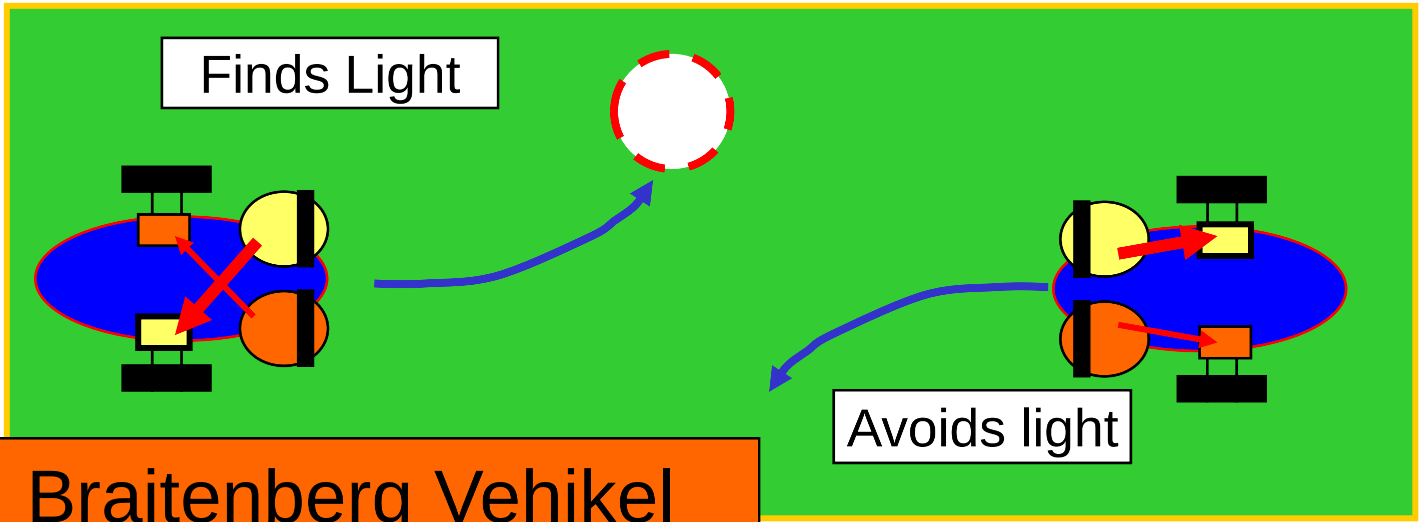
Overview

- Introduction
- Representations of Environments
- Maps
- **Controverses about World Models**
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Controverses: How Much World Model?

Physical Symbol System Hypothesis
or Physical Grounding Hypothesis?

„Ant at the beach finds her path without maps.“



Braitenberg Vehikel

Physical Symbol System Hypothesis

"A physical symbol system has the necessary and sufficient means for intelligent action.“

Newell/Simon: "Computer Science as Empirical Inquiry: Symbols and Search“

GOFAI= „good old fashioned AI“

Needs:

- Complete Descriptions of the Worlds
- Algorithms for actions

Many critics
(Dreyfus, Searle, Penrose, ..., Brooks, Maes, Pfeiffer...)

Physical Grounding Hypothesis

This hypothesis states that to build a system that is intelligent it is necessary to have its representations grounded in the physical world. Our experience with this approach is that once this commitment is made, the need for traditional symbolic representations fades entirely. The key observation is that the world is its own best model. It is always exactly up to date. It always contains every detail there is to be known. The trick is to sense it appropriately and often enough.

To build a system based on the physical grounding hypothesis it is necessary to connect it to the world via a set of sensors and actuators. Typed input and output are no longer of interest. They are not physically grounded.

R.A. Brooks: Elephants Don´t Play Chess

Physical Grounding Hypothesis

This hypot … ystem that is intelligent
it is neces … ons grounded in the
physical w … s approach is that once
this comm … traditional symbolic
representa … observation is that the
world is its … s exactly up to date. It
always contains every detail there is to be known. **The trick is to sense it appropriately and often enough.**

New Problem

To build a system based on the … s
it is necessary to connect it to t …
and actuators. Typed input and …
interest. They are not physically …

R.A. Brooks: Elephants Don´t Play Chess

But: To bring the Beer from the basement, the robot should have an idea about the location etc...

Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- **Formal Descriptions of World Models**
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Knowledge Representation in AI

Explicit Knowledge:

Facts, rules, … *(rules of chess)*

Implicit Knowledge:

Deduced from explicit knowledge *(how to play chess)*

Description by (machine processible) formal systems, e.g.:

- symbols for sensor data (e.g. pixels)
- symbols for landmarks
- symbols for relations (e.g. distances)
- methods for calculations (e.g. image interpretation)

Syntax and Semantics

Semantics: Meaning of symbols, meaning of sensor signals?

- by conventions
- by real world as common reference system

Natural systems: experience, teaching

Symbol Grounding

Technical systems: formal semantics, algorithms

Interpretation of sensor data by programs
as
image, force, motion, words, …

General Properties of Knowledge

Imprecise knowledge („*between 3 and 4 cm*“)

Uncertain knowledge („*possibly 3 cm*“)

Probability Theory *P(d=3cm) = 60%*

Modale Logics *possible(d=3cm) = true*

Vague knowledge („very near“)

Multivalued Logics *truth-value(d=3cm) = 0,7*

Fuzzy-Theory *(d=3cm)Î “very near” = 70%*

Combinations: „probably very near“

General Properties of Knowledge

Imprecise knowledge

can be treated by alternatives (“or”) resp. intervals.

Uncertain and vague knowledge by related theories:

	Certain, reliabe	*Uncertain, unreliable*
Crisp	Logics, Set Theory, Algebra, Analysis, ...	Probability Theory, Statistics, Modal Logics, Decision Thory ...
Vague *(„fuzzy“)*	Multi-Valued Logics Fuzzy Theory ...	

Probability Theory

for environments with properties like

- Definiteness: deterministic/non-determistic/**stochastic**
- Observability:
  - complete vs. **partial**
  - correct vs. **uncertain ("noise")**
- Influenceability:
  - complete vs. **partial**
  - effective vs. **ineffective**

Properties may appear differently to the robot. They depend on available ressources: „Bounded Rationality"

Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- **Descriptions of Other Actors**
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Representation of Other Actors

Actors: humans, robots, …

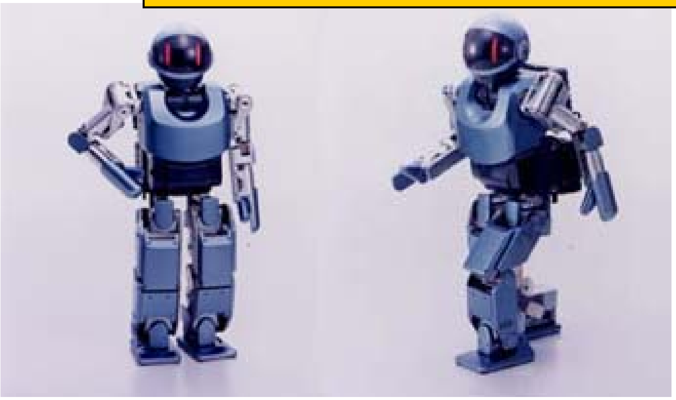
General properties:

- Morphology
- Sensors, Actuators, …
- Skills
- Roles

Situational properties

- Pose
- Bodily states: Control parameters, Sensor values, Energy,…
- Mental states: Belief, Duties, Intentions, Plans, Emotions,…

Philosophical Problems:
– Consciousness
– Free Will



Representation of Other Actors

Representation of mental states

(Belief, Duties, Intentions, Plans, Emotions,…)

e.g. by Modal and Temporal Logics

IF BELIEF (Willie, task = bring water)
AND CAN (Willie, task = bring water)
THEN BELIEF (Opa, LATER(Oma, will have water))

Modal Logical Representation

$$\text{HOLDS}(\mathsf{offside\text{-}punishable}(p), \langle max(s_j, s_m), min(e_l, e_m)\rangle) \Leftrightarrow$$

$$\exists j, k, l, m, p_2 :$$

$$\text{OCCUR}(\mathsf{kick}(p_2), j) \wedge \text{HOLDS}(\mathsf{offside\text{-}position}(p), k) \wedge$$

$$\text{HOLDS}(\mathsf{ball\text{-}free}, l) \wedge \text{HOLDS}(\mathsf{approaching}(p, ball), m) \wedge$$

$$starts(j, l) \wedge in(j, k) \wedge contemporary(l, m) \wedge \mathsf{team}(p) = \mathsf{team}(p_2).$$

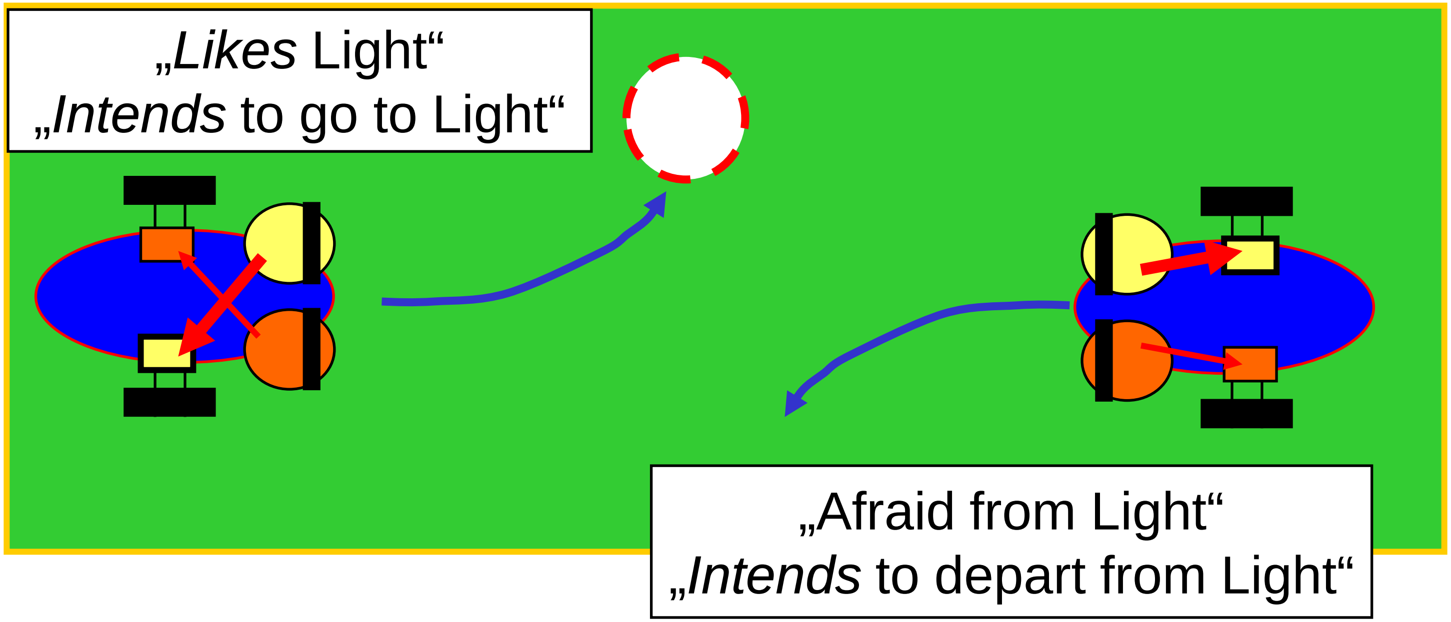
starts, in, contemporary
denote relations between
temporal intervals

(PhD thesis Andrea Miene - Bremen, 2003)

Representation of Other Actors

Ascription or reality?

Is it helpful for modeling and reasoning?



Experiments in 2D-League RoboCup

Observation and classification of opponent teams:

- Player parameters (size, force, energy,…)
- Skills (e.g. dribbling)
- Actions (e.g. passing behavior)
- Strategic behavior (e.g. offside trap)

Coach agent can observe complete field:

- Online observation/classification

Logfiles can be analyzed

- Offline mining

Problem:
Opponent team behavior depends on own team behavior.

Experiment: Behavior Recognition

Behavior to be recognized:

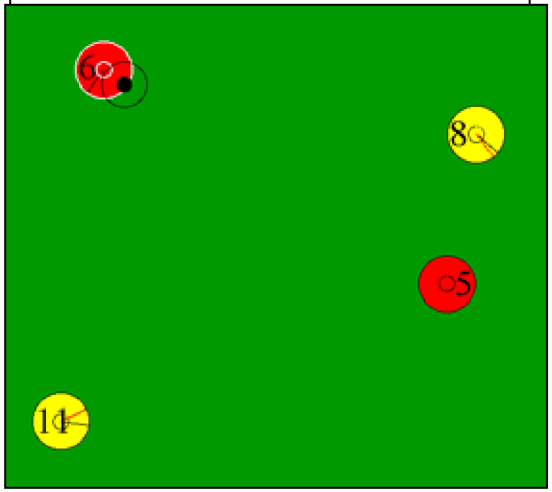
Which pass is performed under which conditions?

Collection of “cases” from logfiles (by analysis over time):

Case description by

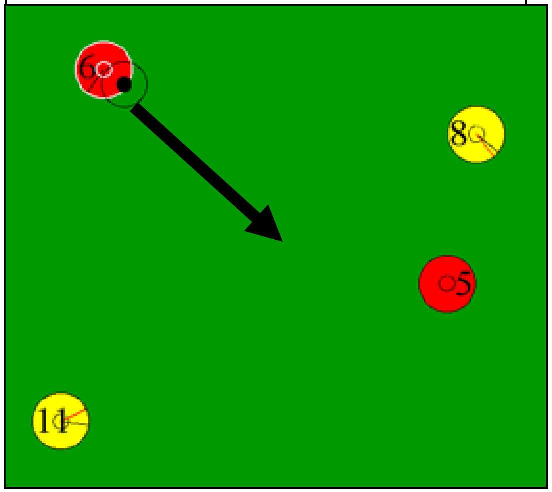
PhD thesis
Jan Wendler

Trigger
(part of situation)



+

Behavior
(part of situation)



Experiment: Behavior Recognition

Hypothesis of Case Based Reasoning (CBR):
Similar trigger leads to similar behavior.

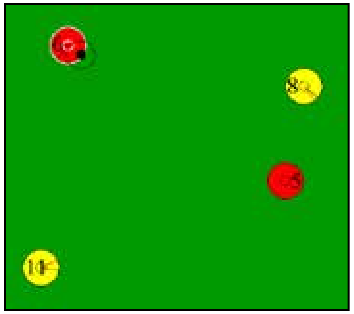
Application of Case Based Reasoning (CBR)
for a given situation with known trigger:

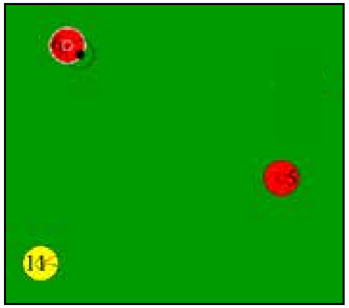
1. Search for cases with similar triggers in your collected cases.
2. Adapt behavior of found cases.
3. The adapted behavior is expected.

Needs:

- Collection of cases.
- Similarity measure.
- Retrieval methods.
- Adaptation methods.

Experiment: Similarity of Triggers





Triggers T^1, T^2 described by attributes for:

- Distances
- Directions
- Players

Similarities for attributes a_i : $sim_i(a^1_i , a^2_i)$

Similarity of Triggers by weighted sum:

$$Trigger\_sim(T^1,T^2) = \Sigma_{i=1,\ldots,n} \; w_i * sim_i(a^1_i , a^2_i)$$

weights w_i by experiments

Experiment: Evaluation

Accuracy of prognosis was less than 50%

Passing player

- has limited view
- may decide regarding not only actual local environment (e.g. depending on team communication, mental state ,…)

Representation of „Social Relations“

Robots to
- to other robots
- to humans

Multi Robot Systems
- Coordination (protocols)
- Communication (protocols, languages)
- Cooperation

Organization (structures, hierarchies, ...)
- roles (permanently)
- tasks (temporarily)

Social attributes
- Responsibilities, duties, ...
- Cooperativeness, altruism, …
- Trustworthiness, reliability, believability, …

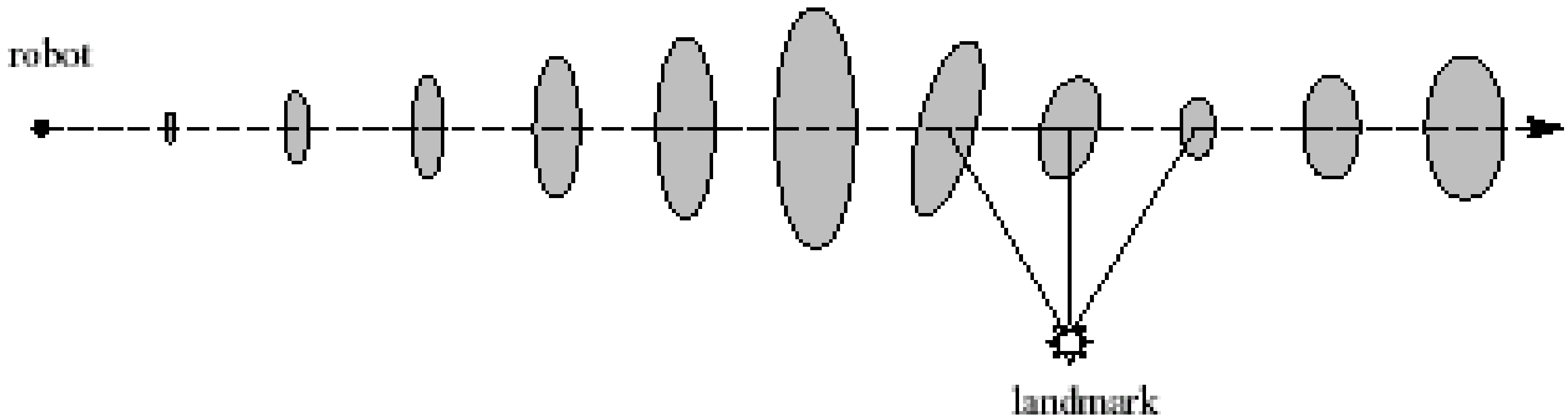
Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- **Probabilistic Methods: Bayes Filter**
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- SLAM

Data Integration/Fusion

„Fusion“, „Integration“ of information

e.g. from motion (odometry) and observation



Need methods for combination:

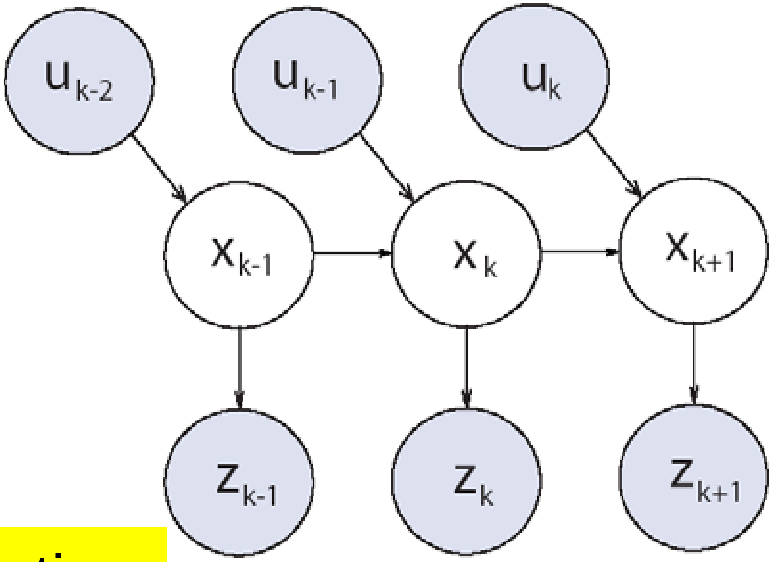
- Calculations of the desired values
- Estimation of reliability

Stochastic World

- Uncertainties with respect to world state
- Uncertainties regarding results of actions
- Uncertainty regarding observations

x_i state at time i
u_i action at time i
z_i observation at time i

Bayesian Net



Observations provide new information

Bayesian Model

Probability of state x_{t+1} after actions $u_1,..,u_t$ observations $z_1,..,z_{t+1}$:

Bel (x_{t+1}) („Belief that world is in state x_{t+1})

$= P(x_{t+1} \mid u_1,..,u_t , z_1,..,z_{t+1})$

$= a \times P(z_{t+1} \mid x_{t+1}, z_1,..,z_t , u_1,..,u_t) \times P(x_{t+1} \mid z_1,..,z_t , u_1,..,u_t)$

(by Bayesian formula)

$= a \times P(z_{t+1} \mid x_{t+1}) \times P(x_{t+1} \mid z_1,..,z_t , u_1,..,u_t)$

(by Markov condition)

$= a \times P(z_{t+1} \mid x_{t+1}) \times$ **bel (x_{t+1})**

where **bel (x_{t+1})** is the
Probability of state x_{t+1} after actions $u_1,..,u_t$ and old observations $z_1,..,z_t$

Bayesian Model

Probability of state x_{t+1} after actions $u_1,..,u_t$ and old observations $z_1,..,z_t$

bel (x_{t+1})

$= P(x_{t+1} \mid z_1,..,z_t , u_1,..,u_t)$

$= \int P(x_{t+1} \mid x_t , z_1,..,z_t , u_1,..,u_t) \times P(x_t \mid z_1,..,z_t , u_1,..,u_t)\, dx_t$

by using total probability

$= \int P(x_{t+1} \mid x_t ,u_t) \times P(x_t \mid z_1,..,z_t , u_1,..,u_t)\, dx_t$

by Markov Condition

$= \int P(x_{t+1} \mid x_t ,u_t) \times P(x_t \mid z_1,..,z_t , u_1,..,u_{t-1})\, dx_t$

by assuming x_t independent from u_t

$= \int P(x_{t+1} \mid x_t ,u_t) \times \mathbf{Bel(x_t)}\, dx_t$

Bayesian Model

Recursion formula

$Bel(x_{t+1}) = \alpha \cdot \int P(z_{t+1} \mid x_{t+1}) \cdot P(x_{t+1} \mid x_t, u_t) \cdot Bel(x_t)\, dx_t$

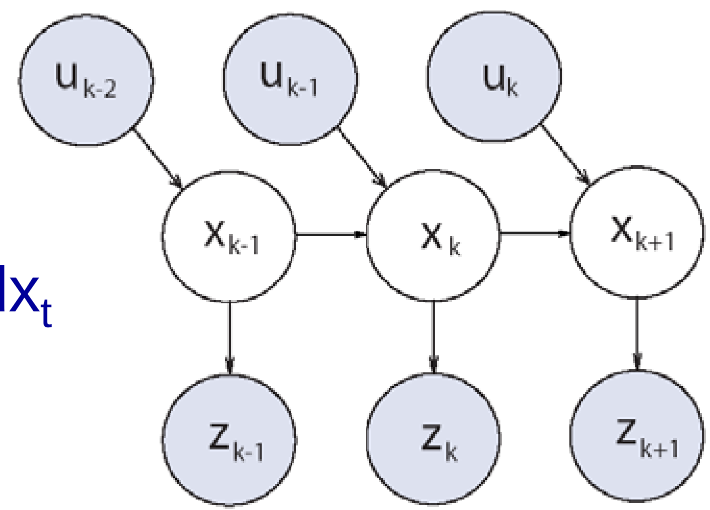
„Bayes Filter“:

Start with initial belief $Bel(x_0)$

Update by

a) „Motion model“ $P(x_{t+1} \mid x_t, u_t)$

$$bel(x_{t+1}) = \int P(x_{t+1} \mid x_t, u_t) \cdot Bel(x_t)\, dx_t$$



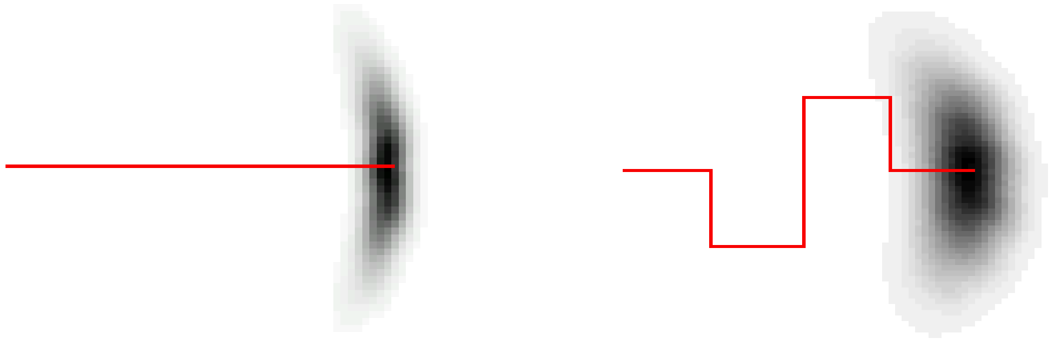
b) „Sensor model“ $P(z_{t+1} \mid x_{t+1})$

$$Bel(x_{t+1}) = \alpha \cdot P(z_{t+1} \mid x_{t+1}) \cdot bel(x_{t+1})$$

Motion Model

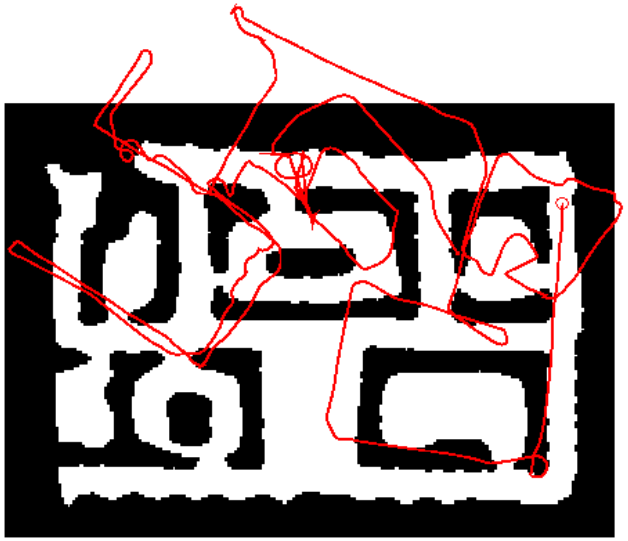
Tutorial: Thrun 2000

Motion model $P(x_{t+1} \mid x_t, u_t)$



Problems with control errors and odometry errors (especially for angles)
e.g. by

- Lack of traction
- Hardware problems
- Obstacles (other robots)

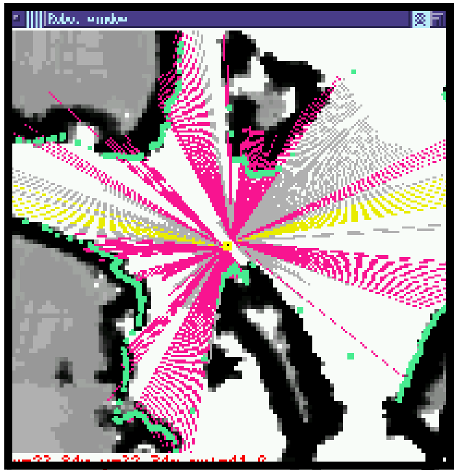


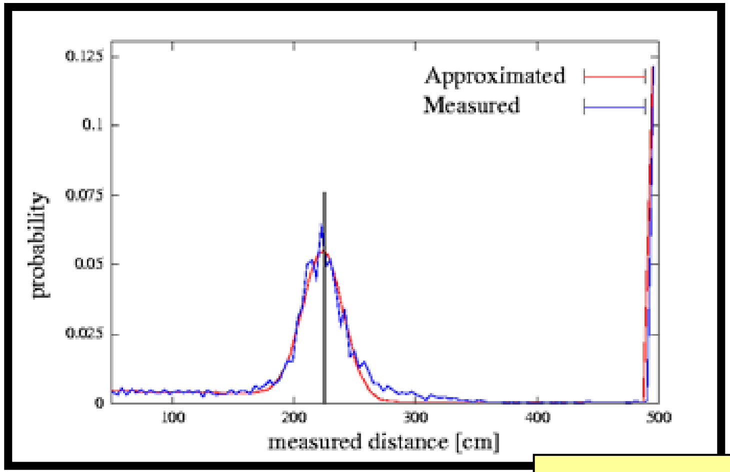
Sensor Model

Sensor model: $P(z_t \mid x_t)$

Correspondence of observation z_t with state x_t

(e.g. observed distances/angles for landmarks)

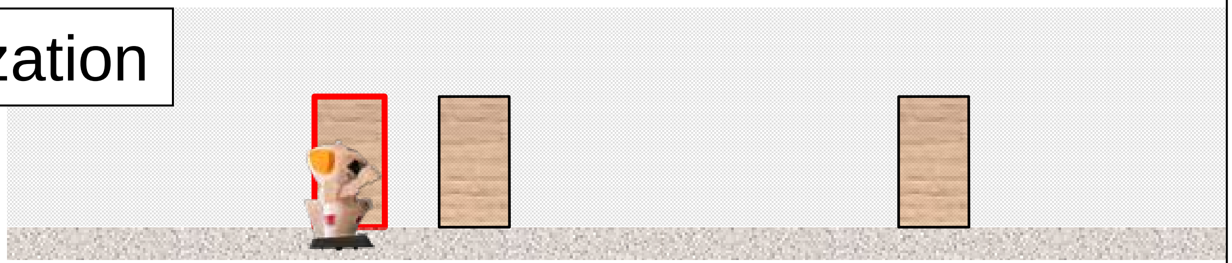




From Tutorial:
Thrun 2000

Bayes Estimation for Several Hypothesis

Localization



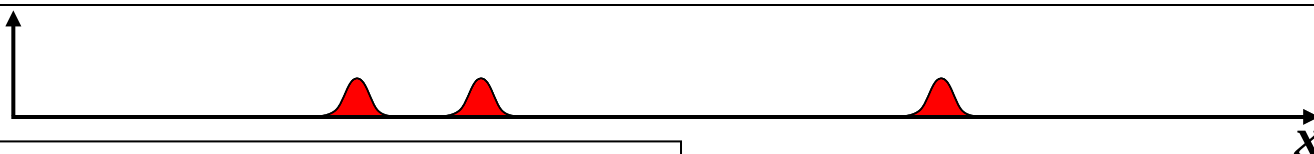
Example by S. Thrun, Images by J. Hoffmann

Initial estimation equally distributed:

$Bel(x_0)$
$= bel(x_0)$

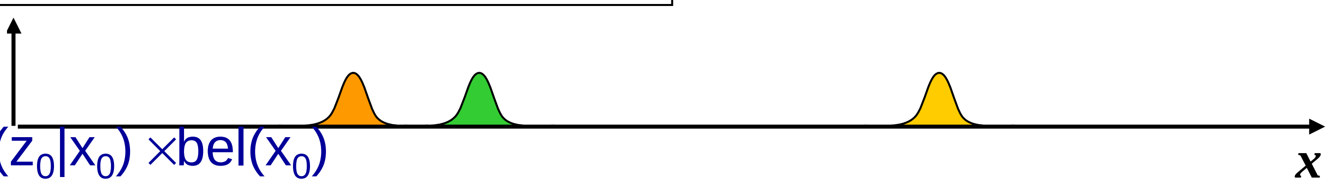
Observation: Robot sees a door, but does not know which one:

$P(z_0|x_0)$



Estimation after observation:

$Bel(x_1)$
$= a \times P(z_0|x_0) \times bel(x_0)$

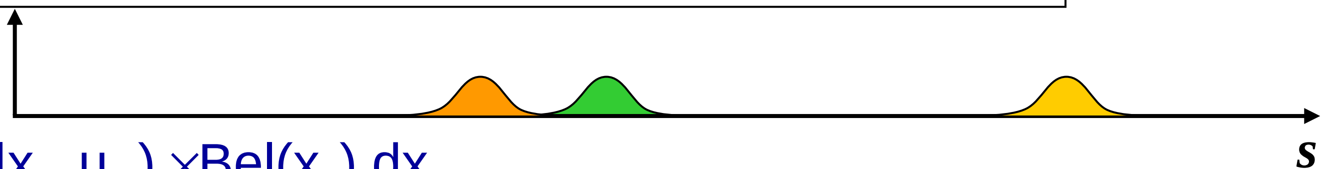


Bayes Estimation for Several Hypothesis

Robot moves:

New estimation according to action model:

bel (x_1)



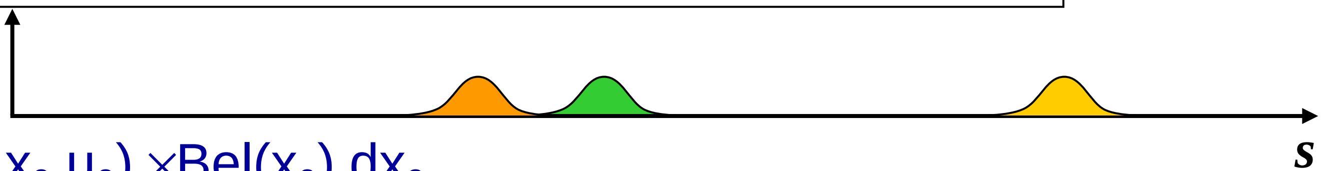
$= \int P(x_1 | x_0, u_0) \cdot Bel(x_0)\, dx_0$

Bayes Estimation for Several Hypothesis

Robot moves:

New estimation according to action model:

bel (x_1)



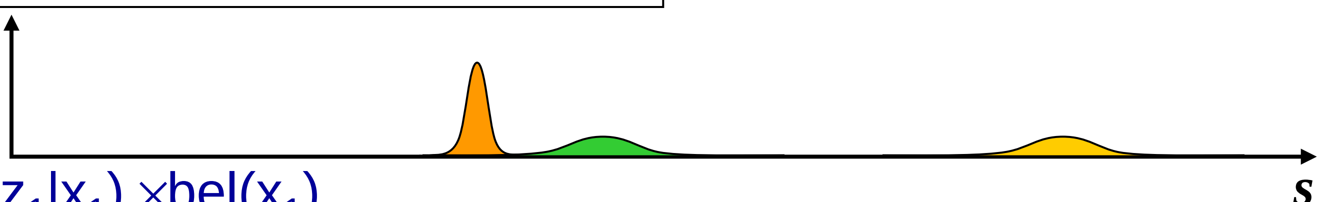
$= \int P(x_1|x_0,u_0) \times Bel(x_0)\, dx_0$

Observation: Robot sees a door, but does not know which one:

$P(z_1|x_1)$

Estimation after observation:

$Bel(x_1)$



$= \alpha \times P(z_1|x_1) \times bel(x_1)$

Bayes Filter: „Markov-localization“

Initialisation: $Bel(x_0)$

(initial position estimation)

A-priori estimation after action u_t :

$$bel\ (x_{t+1}) = \int P(x_{t+1} \mid x_t, u_t) \times Bel(x_t)\ dx_t$$

A-posteriori estimation after observation z_{t+1} :

$$Bel\ (x_{t+1}) = \alpha \times P(z_{t+1} \mid x_{t+1}) \times bel(x_{t+1})$$

with incrementally calculated

normalisation factor $\alpha = 1 / \int Bel(x_t)\ dx_t$

Bayes Filter: „Markov-localization“

Different methods for complex calculations of

$$bel\ (x_{t+1}) = \text{ò}P(x_{t+1}\ |\ x_t\ ,u_t\ )\ Bel(x_t\ )\ dx_t$$
$$Bel\ (x_{t+1}) = a\ P(\ z_{t+1}\ |\ x_{t+1})\ bel(x_{t+1}\ )$$

- Grid based (å instead of ò)
- Kalman Filter

  for Gauß-distributions and linear models

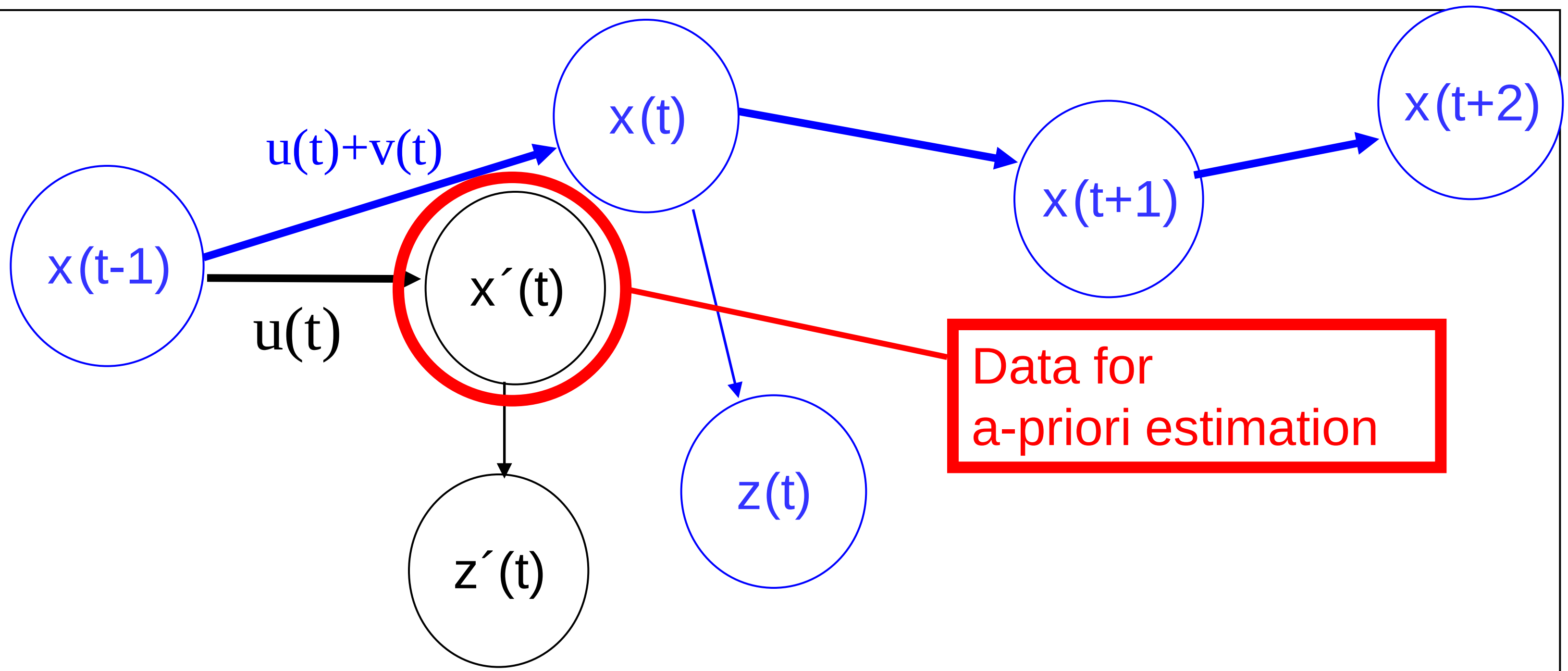
  (extensions for more general situations)
- Monte Carlo Filter/Particle Filter/Importance Sampling

  approximation by weighted examples (particles)

Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- **Data Fusion/Integration**
- Kalman Filter
- Particle Filter
- SLAM

Data Integration/Fusion: Model

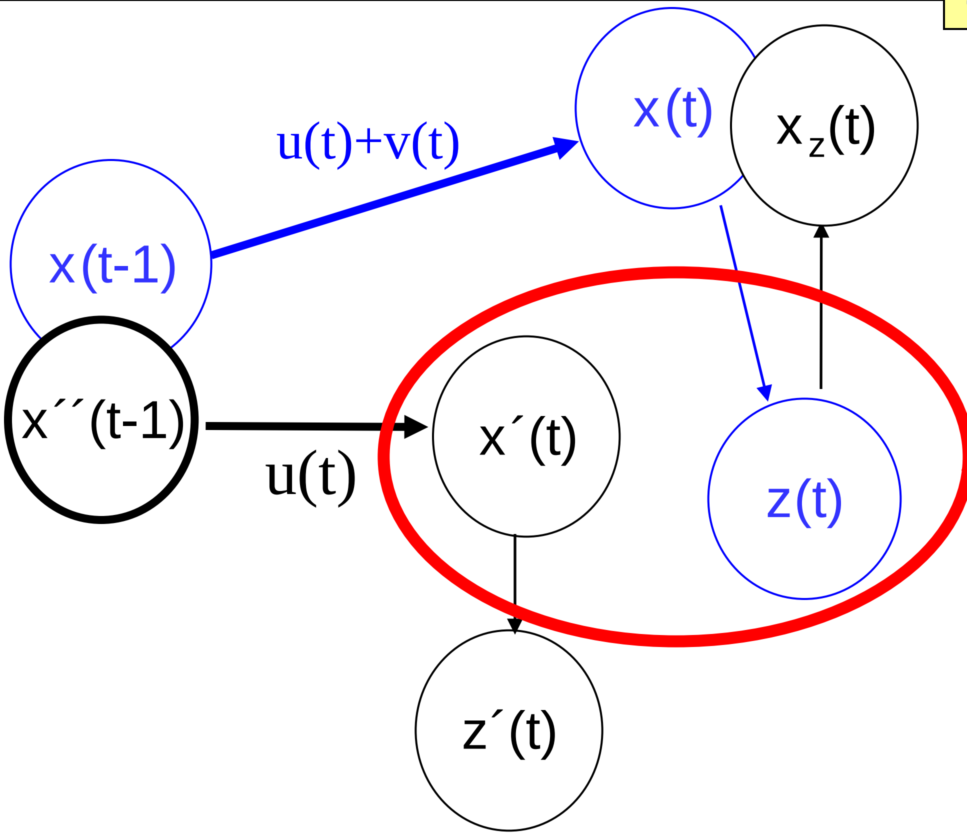


x´(t) : expected state after action u(t)
z´(t) : expected sensor information

x(t) actual state: action u(t) with noise v(t),
z(t) actual sensor information with noise w(t)

Idea of Integration/Fusion

$x(t)$ actual state
$z(t)$ actual sensor information



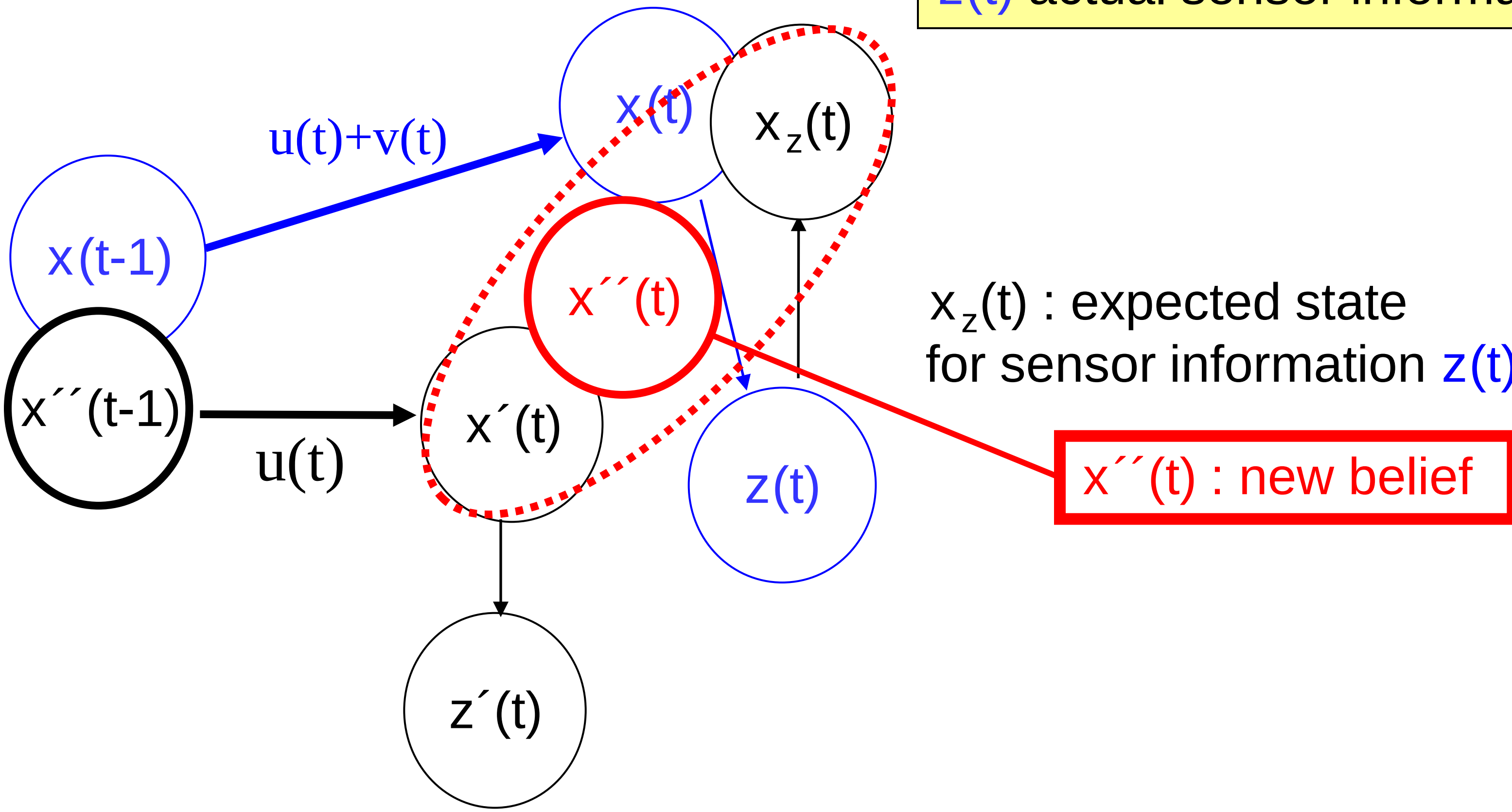
$x_z(t)$: expected state
for sensor information $z(t)$

Data for
a-posteriori estimation

$x´(t)$: expected state after motion $u(t)$ in beliefed state $x´´(t)$
$z´(t)$: expected sensor information

Idea of Integration/Fusion

x(t) actual state
z(t) actual sensor information



x´(t) : expected state after motion u(t) in state x´´(t)
z´(t) : expected sensor information

Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- **Kalman Filter**
- Particle Filter
- SLAM

Kalman filter

Needs special conditions:

Linear models:

- action model $x(t) = \mathbf{A} \times x(t-1) + v(t)$
- sensor model $z(t) = \mathbf{H} \times x(t) \quad + w(t)$

  with matrices $\mathbf{A}$, $\mathbf{H}$
- Normally distributed error (Gaussian distributions)

  estimation for $x(t)$: $P(X=x) = N(m_x, S_x) \times (x)$

  estimation for $z(t)$: $P(Z=z) = N(m_z, S_z) \times (x)$

  with mean m and covariance matrices S
- Normally distributed noise $v(t)$, $w(t)$

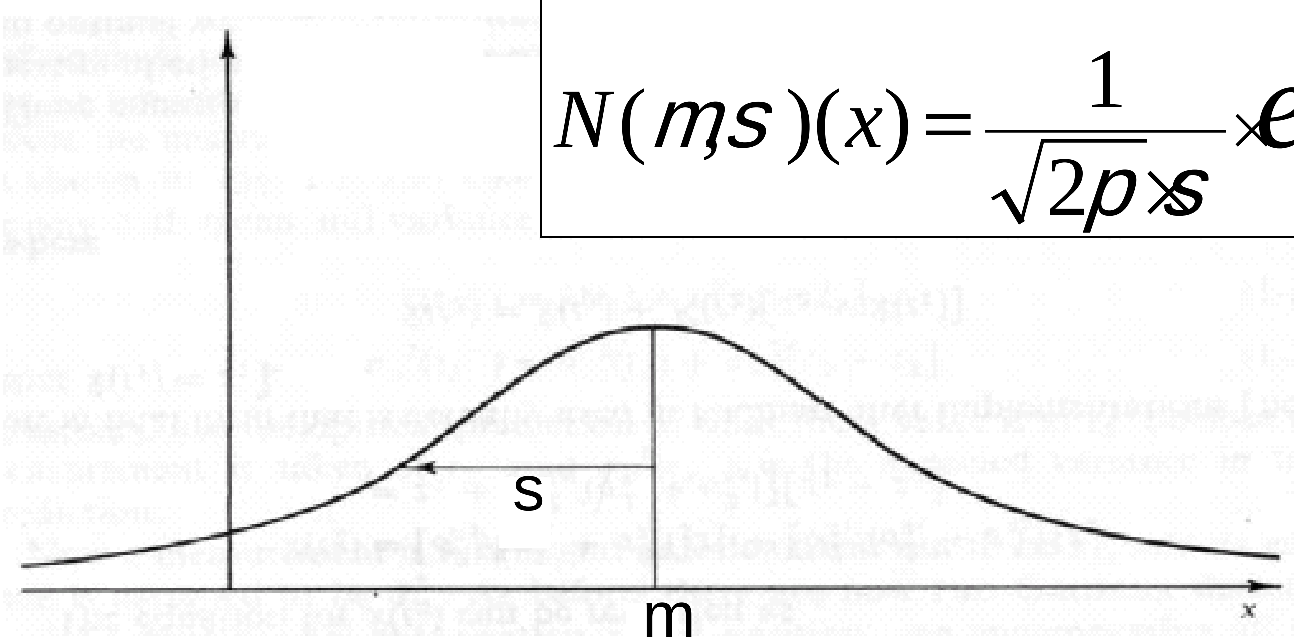
Gaussian Distributions

Gaussian Distributions (Normal Distribution N)

are determined by mean und variance

For one Variable X
with mean mund variance s^2:

$$N(m,s)(x) = \frac{1}{\sqrt{2p} \times s} \times e^{-\frac{1}{2}\frac{(x-m)^2}{s^2}}$$



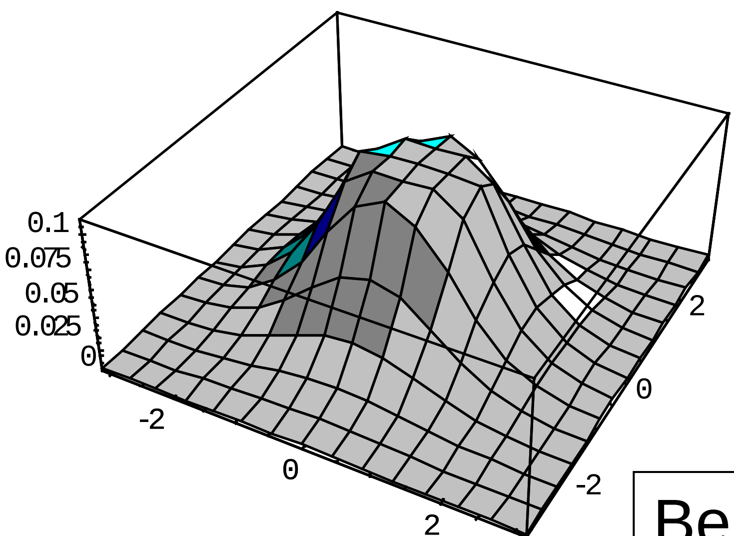
Gaussian Distributions

For n-dimensional vector X
with mean vector **μ** and covariance matrix *Σ*:

$$N(\mu,\Sigma)(x) = a \times e^{-\frac{1}{2}((x-\mu)^{T}\Sigma^{-1}(x-\mu))}$$

$$a = \frac{1}{\sqrt{(2\pi)^{n} * \det(\Sigma)}}$$

x



Bell curve

Kalman filter

$$N(\mu,\Sigma)(x) = a \cdot e^{-\frac{1}{2}((x-\mu)^T \Sigma^{-1}(x-\mu))}$$

Mean μ is the estimation, covariance matrix Σ is the error

A-priori-estimation by action model:

$\mu_{x'}(t) = x'(t) = A \cdot x''(t-1) + v(t)$ Error: $\Sigma_{x'}(t)$

In practice, these errors are usually determined by estimation.

A-posteriori estimation

$\mu_{x''}(t) = x''(t) = x'(t)+c \cdot (x_z(t)-x'(t))$, Error: $\Sigma_{x''}(t)$

with $c \in [0,1]$ determined

using sensor model: $z(t) = H \cdot x(t) + w(t)$

μ_z is the measurement, Error: $\Sigma_z(t)$

In practice, these errors can usually be measured.

Gaussian Distributions

Under the assumption of linear models, i.e.

actions model $x(t) = \mathbf{A} \times x(t\text{-}1) + v(t)$

sensor model $z(t) = \mathbf{H} \times x(t) + w(t)$

with matrices **A**, **H** and Gaussian error $v(t)$, $w(t)$

Gaussian Distributions for $x(t\text{-}1)$

lead to Gaussian Distributions for $x(t)$ und $z(t)$:

$$P(x(t) \mid x(t\text{-}1)) = N(A \times x(t\text{-}1), \boldsymbol{S}_x(t)) (x(t))$$

$$P(z(t) \mid x(t)) = N(H \times x(t), \boldsymbol{S}_z(t)) (z(t))$$

Kalman filter

Calculation of actual distributions from previous distributions:

a-priori-estimation by action model:

$$\mathbf{m}_{x'}(t) = x'(t) = \mathbf{A} \times x''(t-1)$$

$$S_{x'}(t) = \mathbf{A} \times S_{x''}(t-1) \times \mathbf{A}^T + \mathbf{Q}$$

Q is covariance matrix for Gaussian noise $v(t)$

(**Q** could also depend on time).

Kalman filter

a-posteriori estimation by observation model

Uses difference $z(t) - \mathbf{H} \times x´(t)$ („Innovation") between actual observation $z(t)$ and expected observation $\mathbf{H} \times x´(t)$.

$$\mathbf{m}_{x´´}(t) = x´´(t) = x´(t) + \mathbf{K}(t) \times (z(t) - \mathbf{H} \times x´(t))$$

$$\boldsymbol{S}_{x´´}(t) = (\mathbf{I} - \mathbf{K}(t) \times \mathbf{H}) \times \boldsymbol{S}_{x´}(t)$$

Conversion to position and weighting by „Kalman-Gain":

$$\mathbf{K}(t) = \boldsymbol{S}_{x´}(t) \times \mathbf{H}^T / (\mathbf{H} \times \boldsymbol{S}_{x´}(t) \times \mathbf{H}^T + \mathbf{R})$$

using covariance matrix $\mathbf{R}$ for sensor-noise $w(t)$

Kalman filter

Recursive procedure:

First estimation by action model

Second estimation by observation model:

$$x'(0)\ ,\ \ \boldsymbol{S}_{x''}(0)$$

$$x'(t) = \mathbf{A} \times x''(t-1)$$

$$\boldsymbol{S}_{x'}(t) = \mathbf{A} \times \boldsymbol{S}_{x''}(t-1) \times \mathbf{A}^T + \mathbf{Q}$$

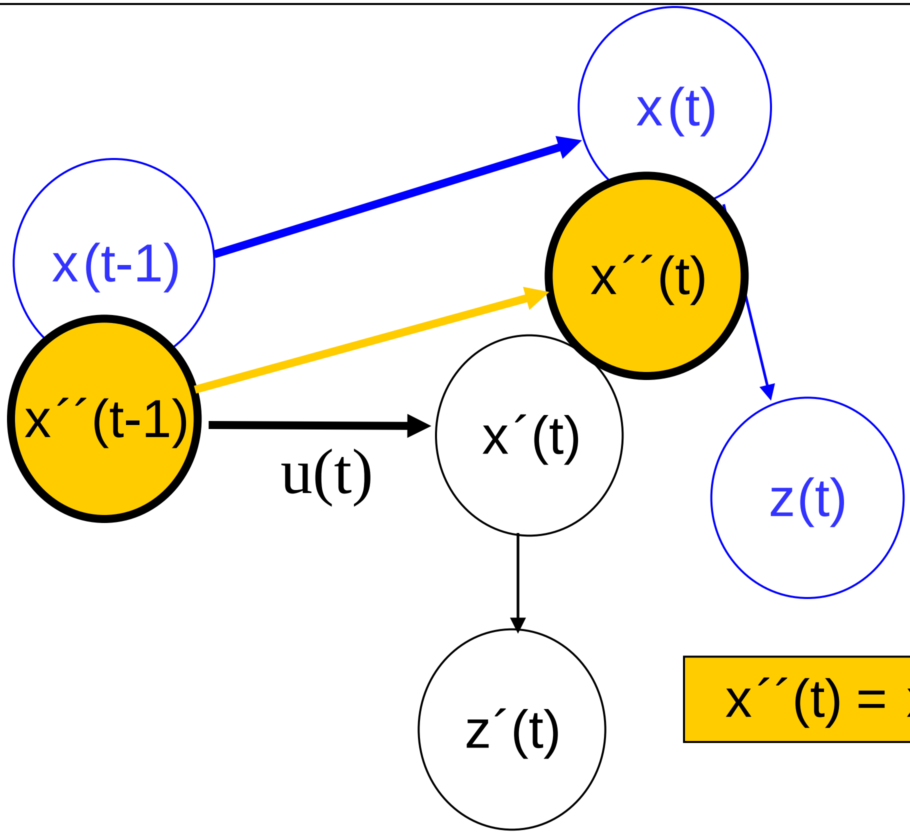
$$\mathbf{K}(t) = \boldsymbol{S}_{x'}(t) \times \mathbf{H}^T \ \mathbf{/} \ (\ \mathbf{H} \times \boldsymbol{S}_{x'}(t) \times \mathbf{H}^T + \mathbf{R}\ )$$

$$x''(t) = x'(t) + \mathbf{K}(t) \times (z(t) - \mathbf{H} \times x'(t)\ )$$

$$\boldsymbol{S}_{x''}(t) = (\ \mathbf{I} - \mathbf{K}(t) \times \mathbf{H}\ ) \times \boldsymbol{S}_{x'}(t)$$

Kalman filter

x(t) actual state
z(t) actual sensor information

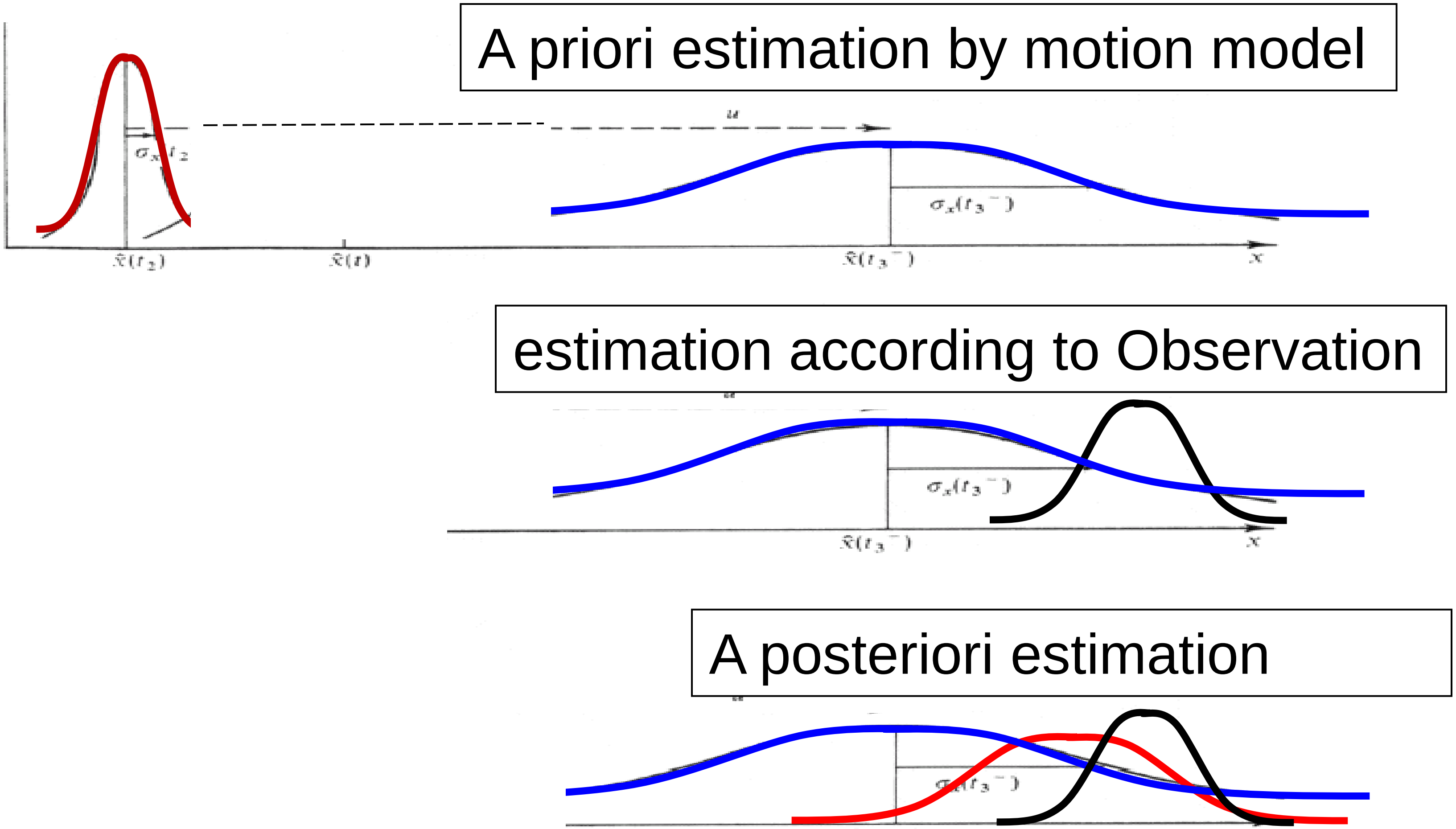


x´´(t) : new hypothesis

x´´(t) = x´(t) + **K**(t) ×(z(t) – **H** x´(t))

x´(t) : expected state after u(t) starting in x´´(t)
z´(t) : expected sensor information

Kalman Filter: 1D-Example



Kalman filter

Assumptions:

Linear Model

Gaussian error

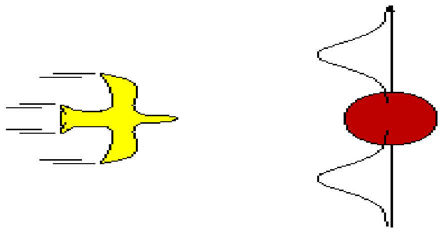
Only one hypothesis

Extensions:

- Extended Kalman Filter

  Linearization by 1. derivation of models
- Unscented Kalman Filter

  Linearization by linear regression of models
- Parallel Kalman Filters



Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- **Particle Filter**
- SLAM

Monte Carlo Filter/Particle Filter

Update after action u and observation z by repeated calculations.

A distribution is represented by “particles” $[s_{t+1}^{(i)}, w_{t+1}^{(i)}]$

with state $s_{t+1}^{(i)}$ and weight $w_{t+1}^{(i)}$

Update method:

1) Choice of examples $s_t^{(i)}$ with probability $w_t^{(i)}$

2) New examples $[s_{t+1}^{(i)}, w_{t+1}^{(i)}]$

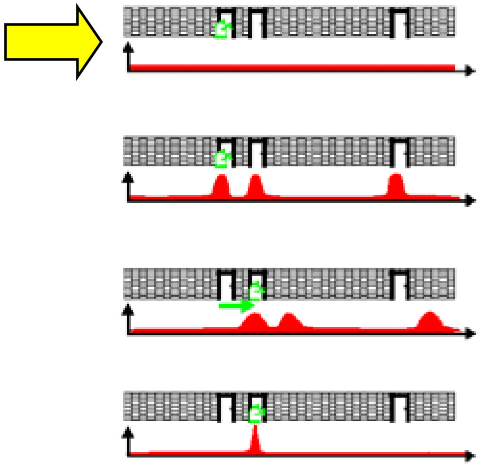
$s_{t+1}^{(i)}$ by **action model** $P(x_{t+1}^{(i)} \mid u, x_t^{(i)})$

$w_{t+1}^{(i)}$ by **sensor model** $w_{t+1}^{(i)} := b \times P(z|x_{t+1}^{(i)})$

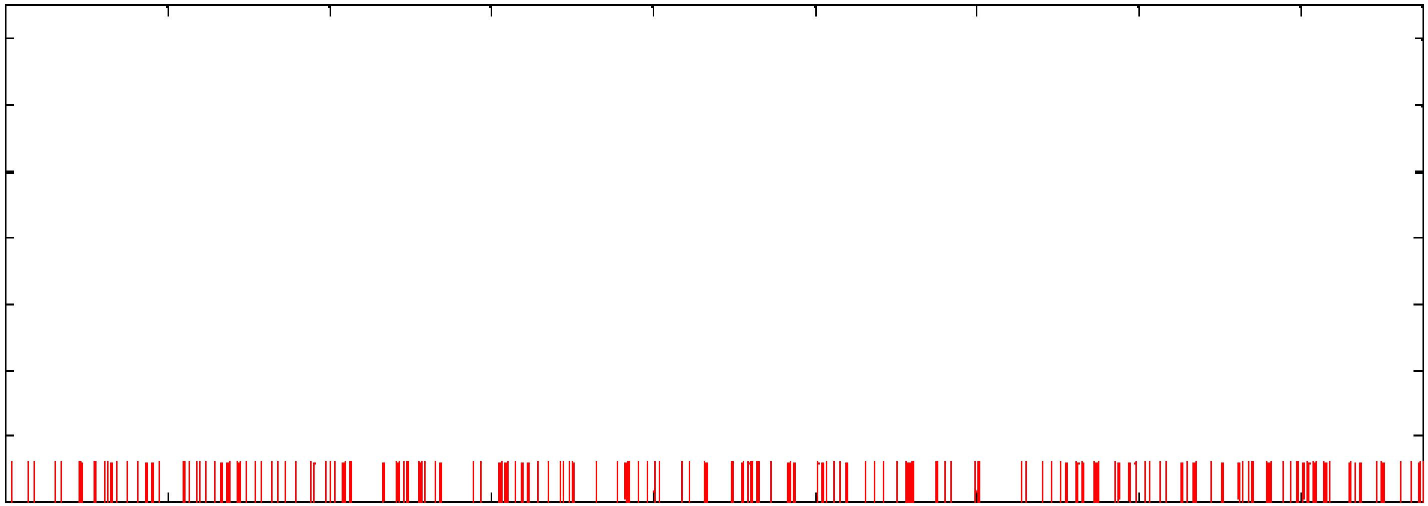
(normalization $b = (S\, w_{t+1}^{(i)})^{-1}$)

Replacement of some particles by randomly chosen new examples (for kidnapping problem)

Monte Carlo Localization (MCL)

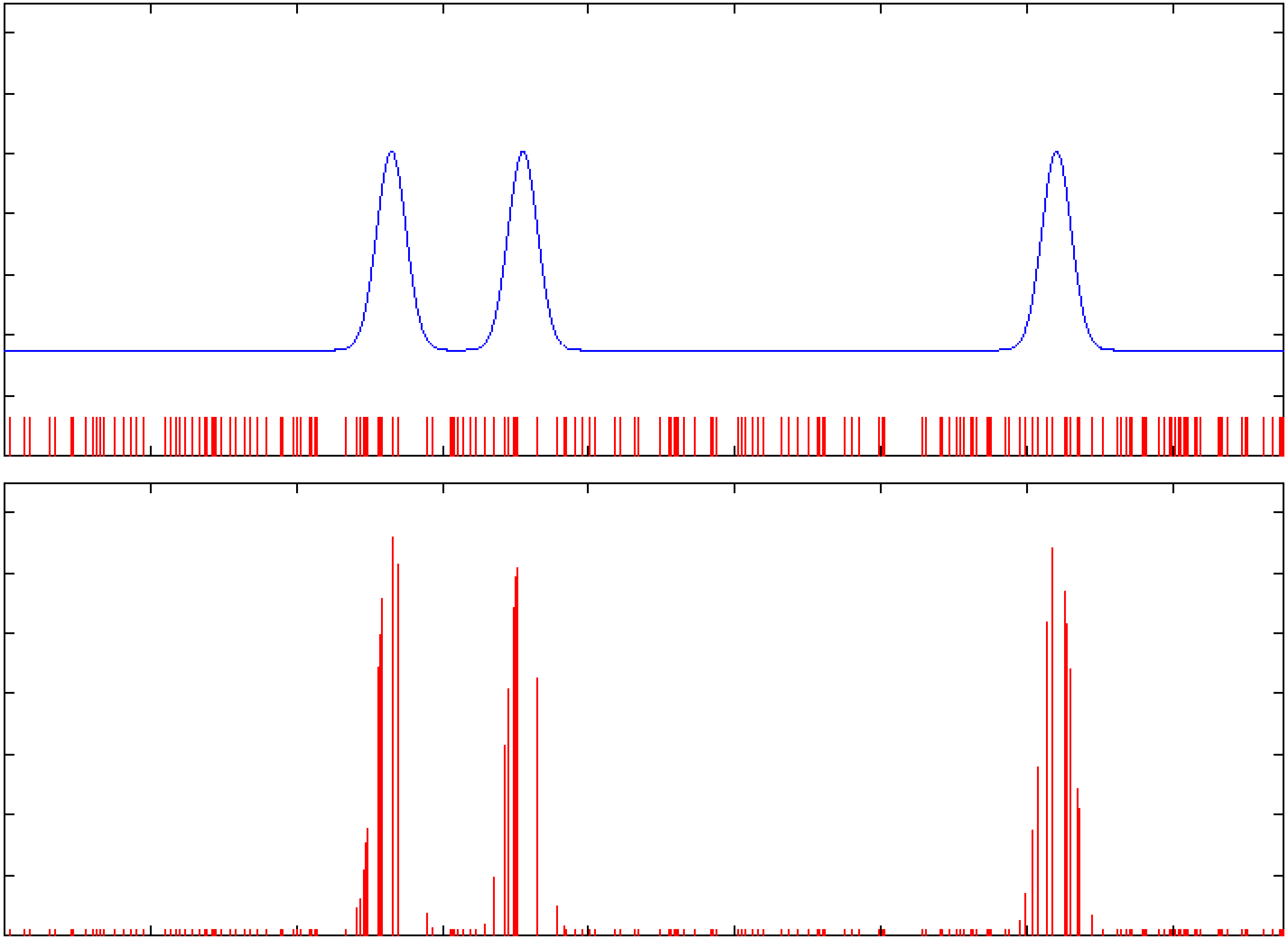


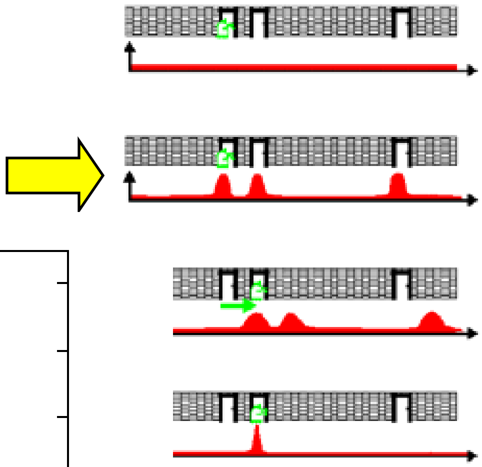
Adapted from Tutorial: Thrun 2000



MCL: Importance Sampling

$w_{t+1}^{(i)} := b \times P(z|x_{t+1}^{(i)})$

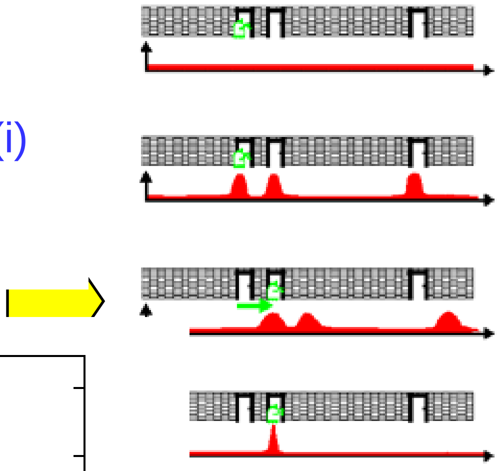


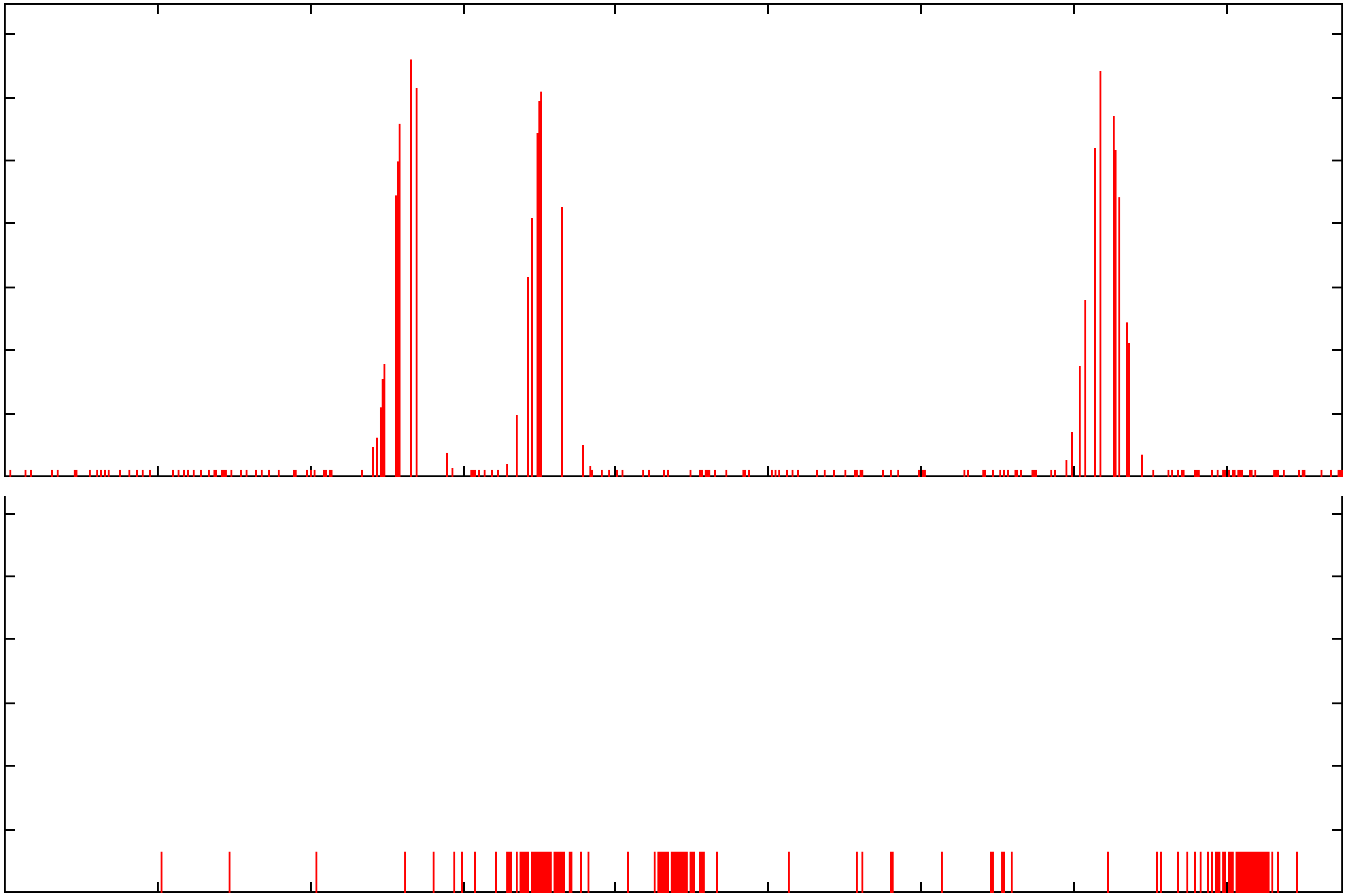


MCL: Robot Motion

Choice of examples $s_t^{(i)}$ with probability $w_t^{(i)}$

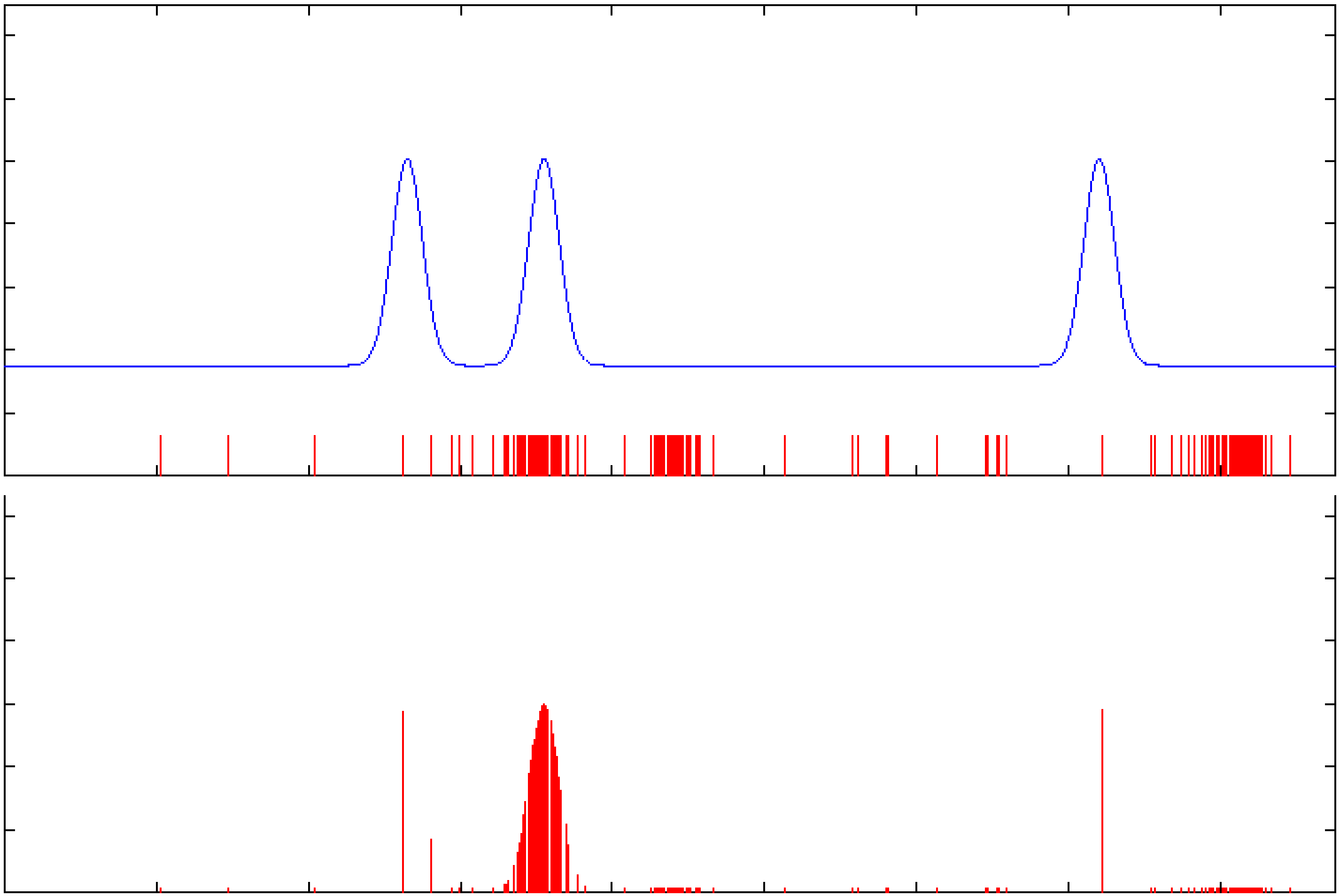
$s_{t+1}^{(i)}$ by **action model** $P\,(x_{t+1}^{(i)}\,|\,u,\,x_t^{(i)}\,)$

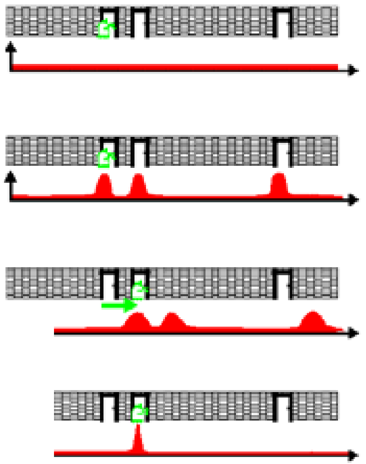




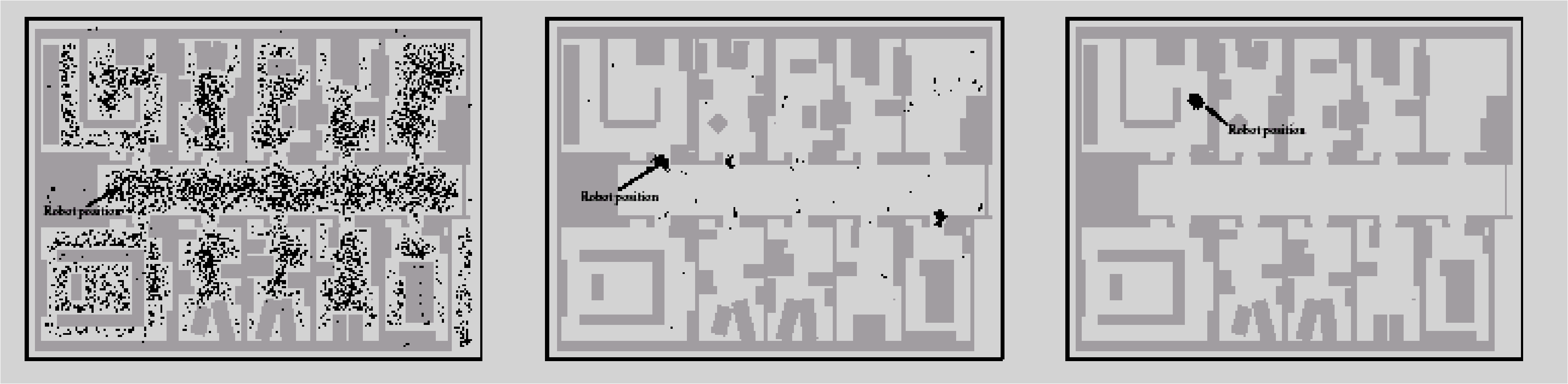
MCL: Importance Sampling

$$w_{t+1}^{(i)} := b \times P(z|x_{t+1}^{(i)})$$

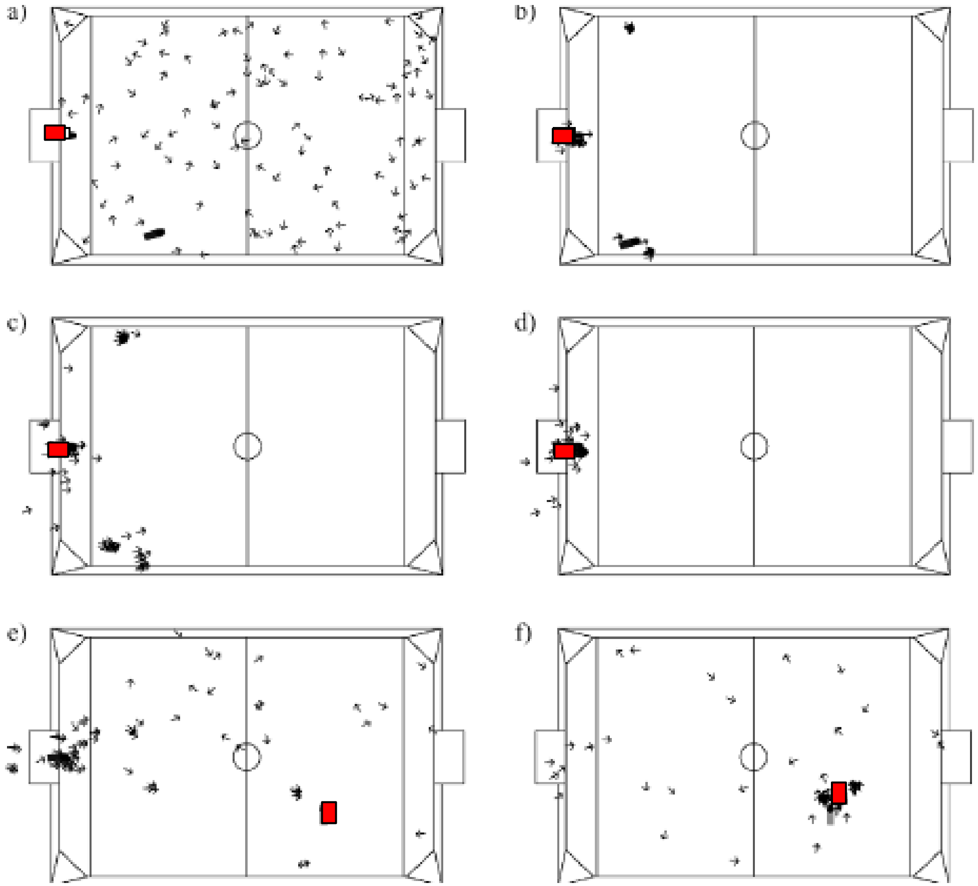




Monte Carlo Filter/Particle Filter



Monte Carlo Filter/Particle Filter

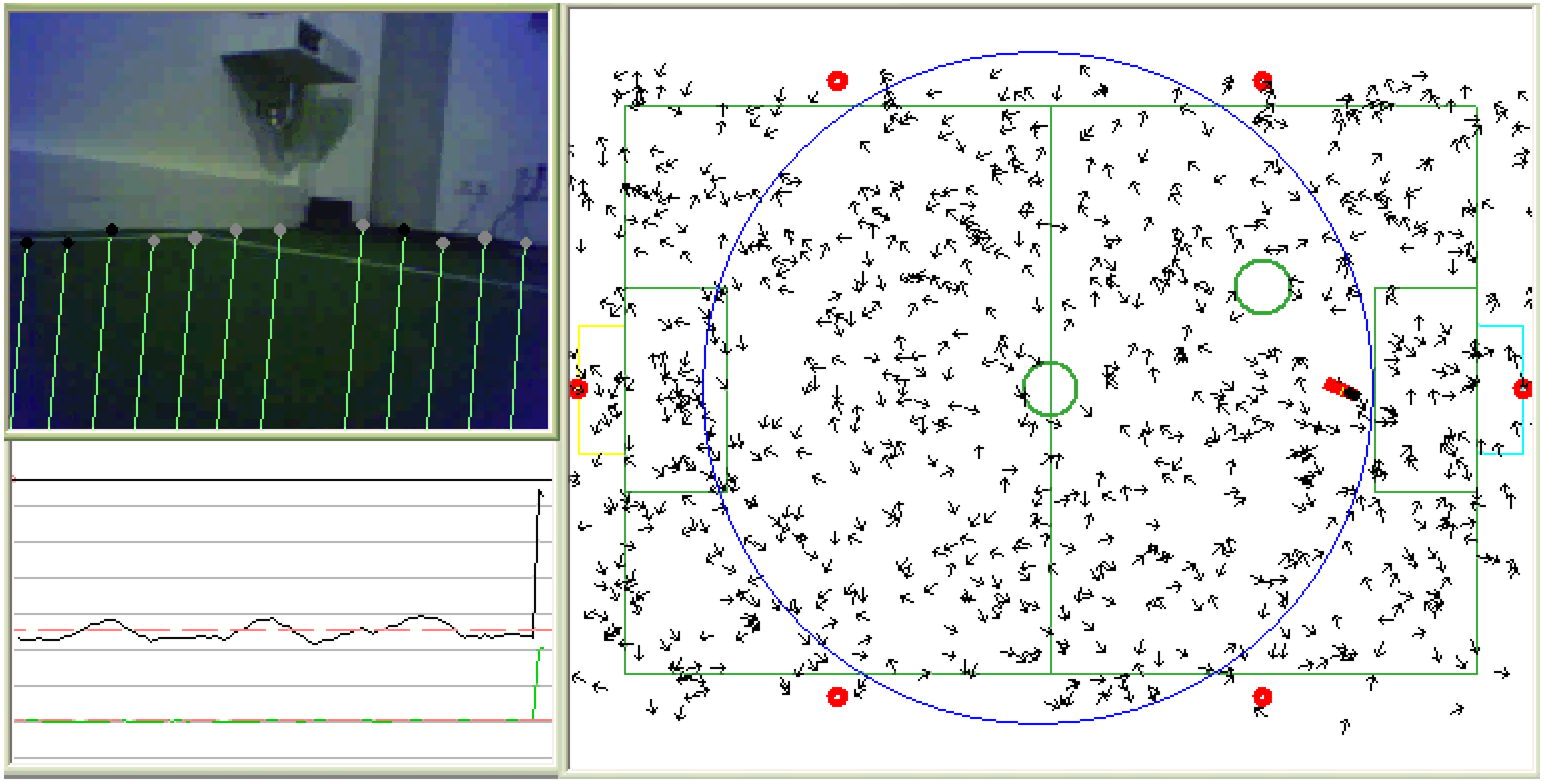


Robot in left goal (figures a-d):
Particle distribution after 1, 8, 14, 40 cycles.

„Kidnapped robot“ (figures e, f):
New stabilization after 13 cycles

(GermanTeam Report 2004)

Monte Carlo Filter/Particle Filter



Monte Carlo Filter/Particle Filter

Can handle several hypothesis (several clusters of particles)

Any-time-Algorithm (number of particles).

Treatment of „Kidnapped Robot Problem“:

- Particles near robot increase:
  - better evaluation (consistent observations),
  - i.e. higher probability for new particles.
- Particles far from robot decrease
  - (inconsistent observations).

Cluster move from old pose to correct new pose.

Re-stabilization time depends on number of random particles:
Trade-off between persistency and adaptability!

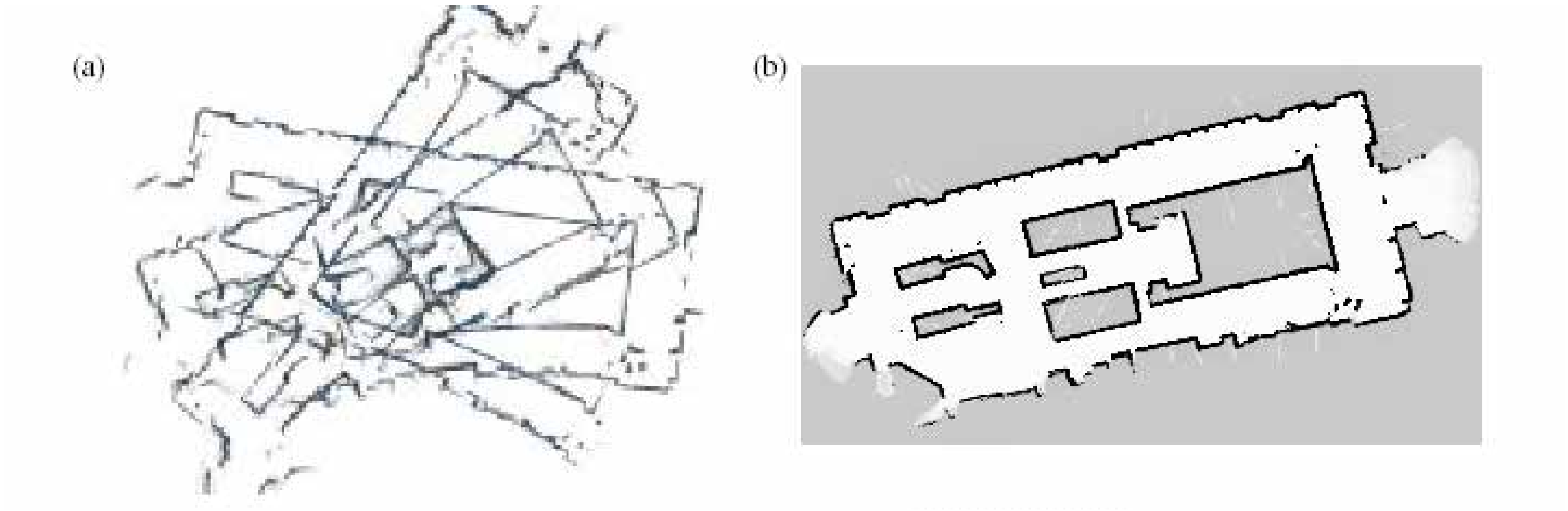
Overview

- Introduction
- Representations of Environments
- Maps
- Controverses about World Models
- Formal Descriptions of World Models
- Descriptions of Other Actors
- Probabilistic Methods: Bayes Filter
- Data Fusion/Integration
- Kalman Filter
- Particle Filter
- **SLAM**

Mapping

SLAM = Simultaneous Localization and Mapping

Robot builts a map using correspondences of positions, motions, landmarks, sensor data, …



Mapping

Useful information:

- Known positions and directions
- Odometry (or control commands)
- Landmarks (correspondence problems to be solved)
- Common knowledge (e.g. about buildings)

Probabilistic Mapping: Kalmanfilter

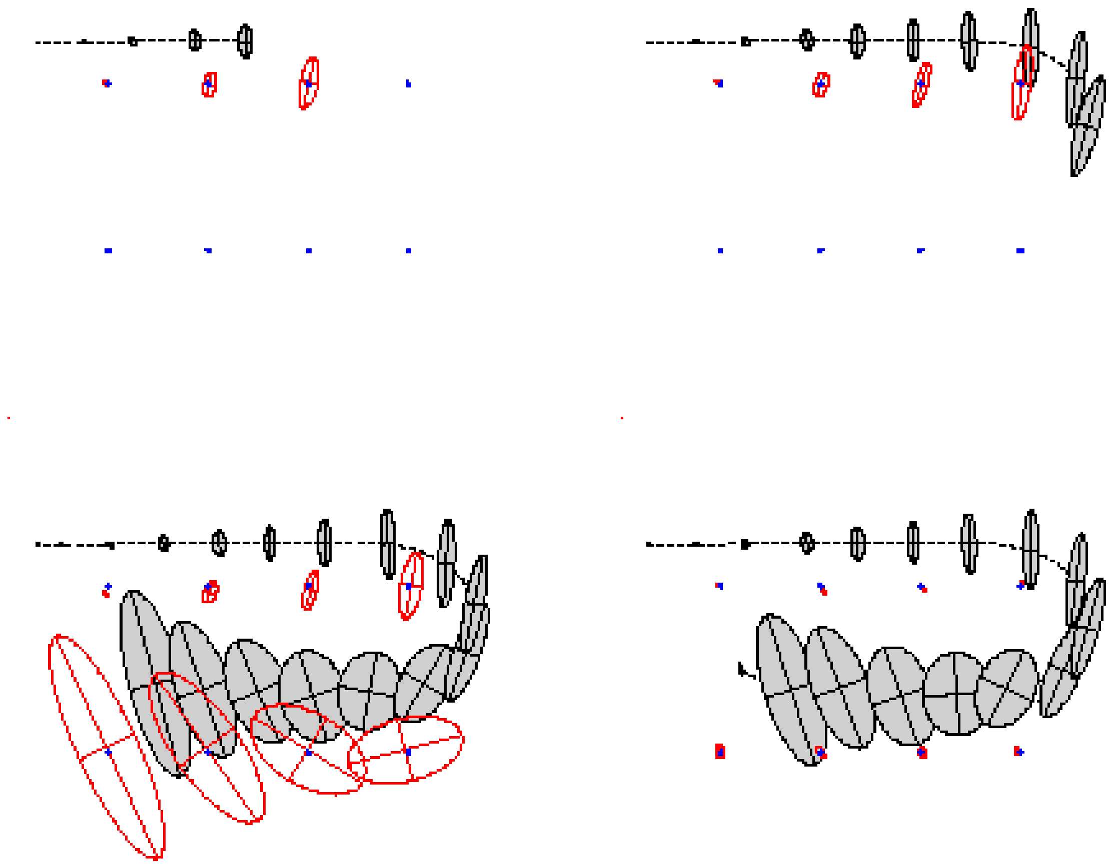
Additional Variables for positions of landmarks.

Actualization of a Gauß-distribution with

2n+3 dimensional mean *m* for estimation of own pose (3) and positions of n landmarks (each 2)

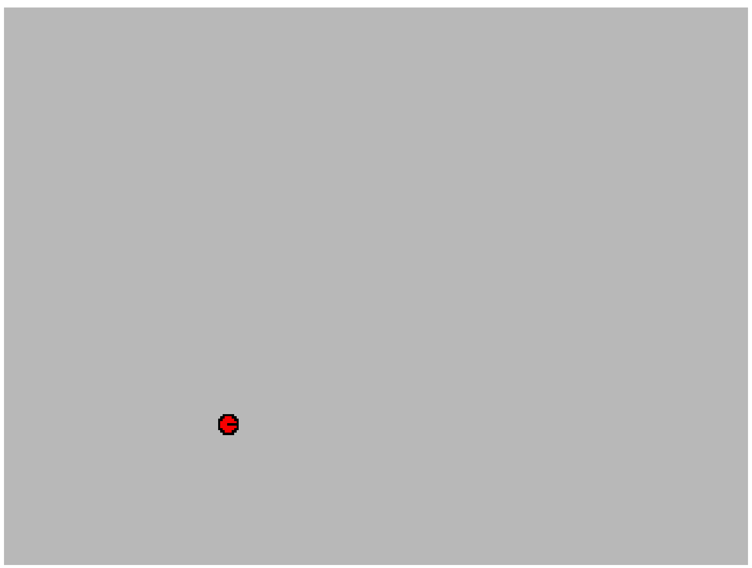
(2n+3)x(2n+3) dimensional covariance matrix *S* for error estimation

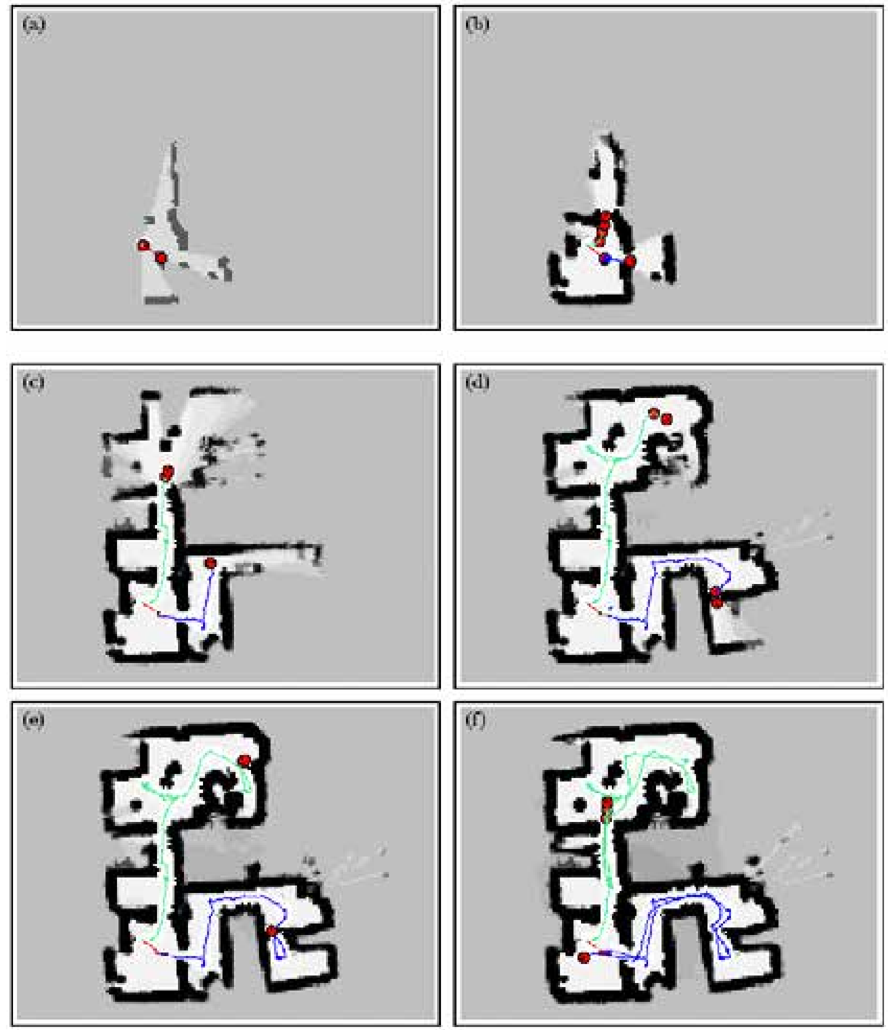
Probabilistic Mapping: Kalmanfilter



Probabilistic Mapping: Kalmanfilter

cooperative mapping by several robots
(Sebastian Thrun, CMU)





Probabilistic Mapping: Cooperating Robots

(Sebastian Thrun, CMU)