Bayesian Filtering

Dieter Fox

Probabilistic Robotics

Key idea: Explicit representation of uncertainty

(using the calculus of probability theory)

- Perception = state estimation
- Control = utility optimization

Bayes Filters: Framework

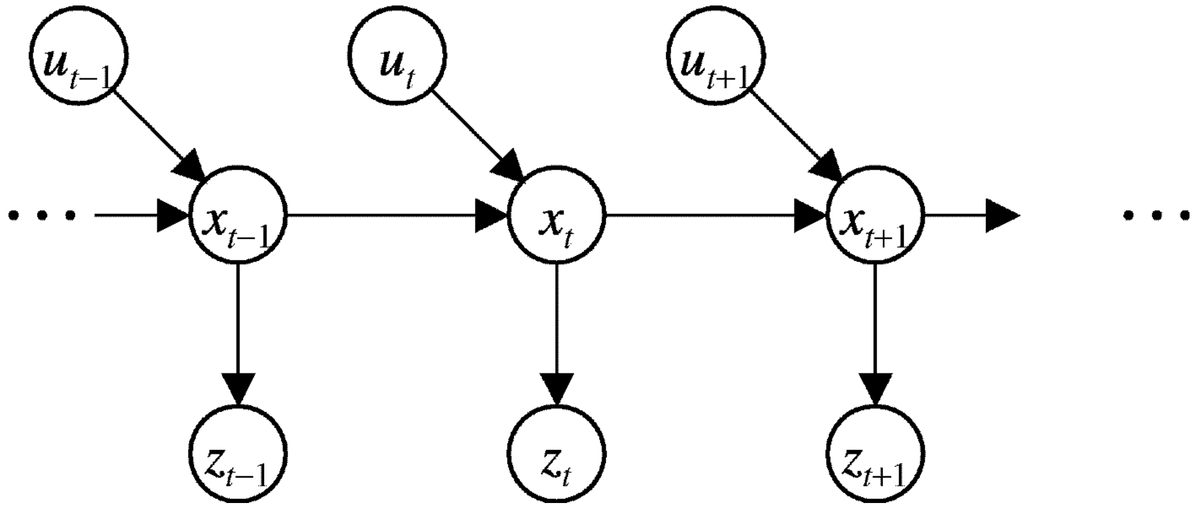
- **Given:**
  - Stream of observations *z* and action data *u:*

$$d_t = \{u_1, z_2 \ldots, u_{t-1}, z_t\}$$

  - Sensor model *P(z|x).*
  - Action model *P(x|u,x')*.
  - Prior probability of the system state *P(x).*
- **Wanted:**
  - Estimate of the state *X* of a dynamical system.
  - The posterior of the state is also called **Belief**:

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Markov Assumption



$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$$
$$p(x_t \mid x_{1:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

Underlying Assumptions

- Static world
- Independent noise
- Perfect model, no approximation errors

Bayes Filters

z = observation
u = action
x = state

$$\boxed{Bel(x_t) =} P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Bayes $$= \eta \; P(z_t \mid x_t, u_1, z_2, \ldots, u_{t-1}) \; P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Markov $$= \eta \; P(z_t \mid x_t) \; P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Total prob. $$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_2, \ldots, u_{t-1}, x_{t-1}) \; P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1}) \; dx_{t-1}$$

Markov $$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1}) \; P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1}) \; dx_{t-1}$$

$$= \boxed{\eta \; P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}}$$

Bayes Filters are Familiar!

$$Bel(x_t) = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ Bel(x_{t-1}) \ dx_{t-1}$$

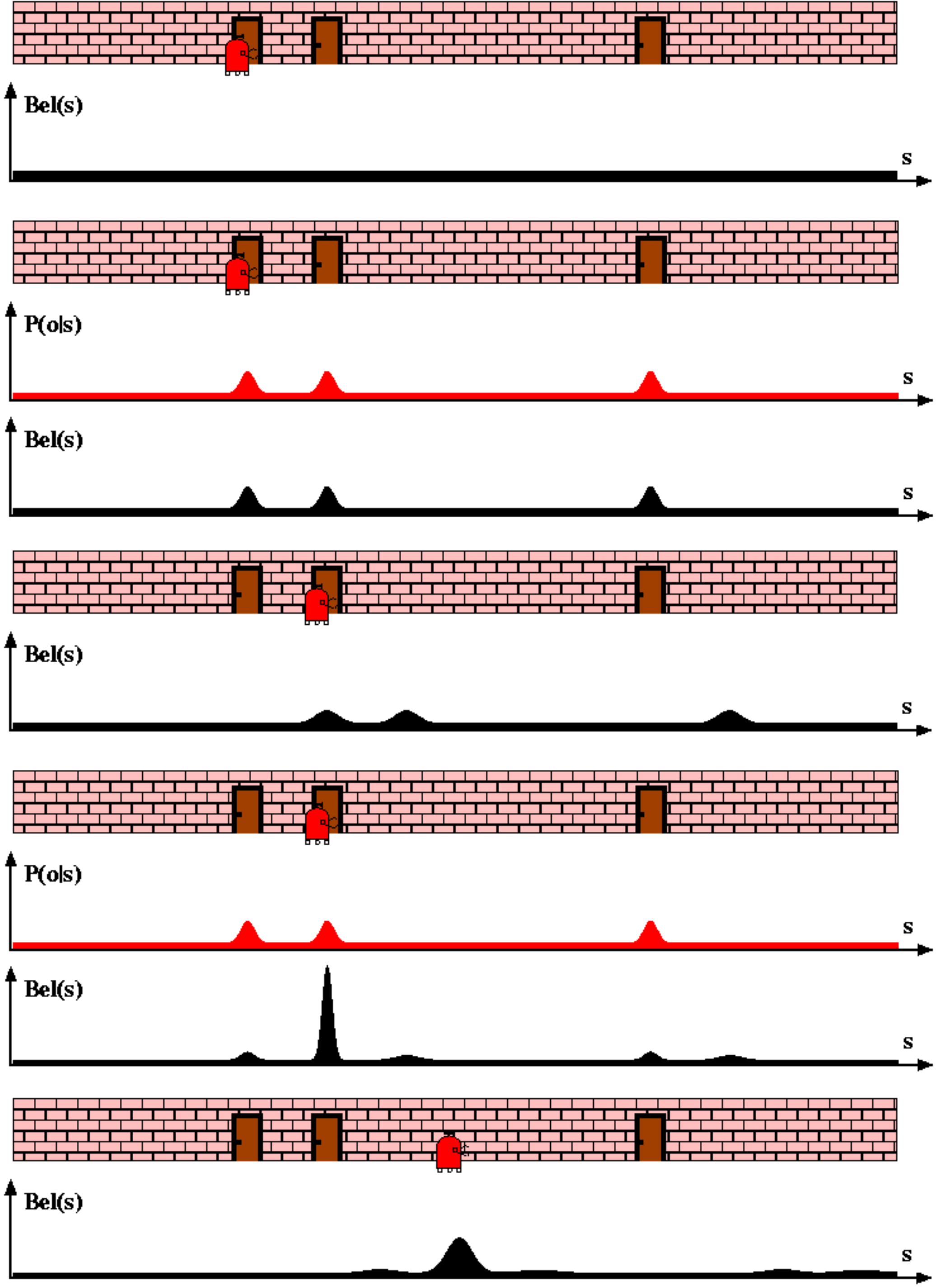
- Kalman filters
- Particle filters
- Hidden Markov models
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

Localization

"Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities." [Cox '91]

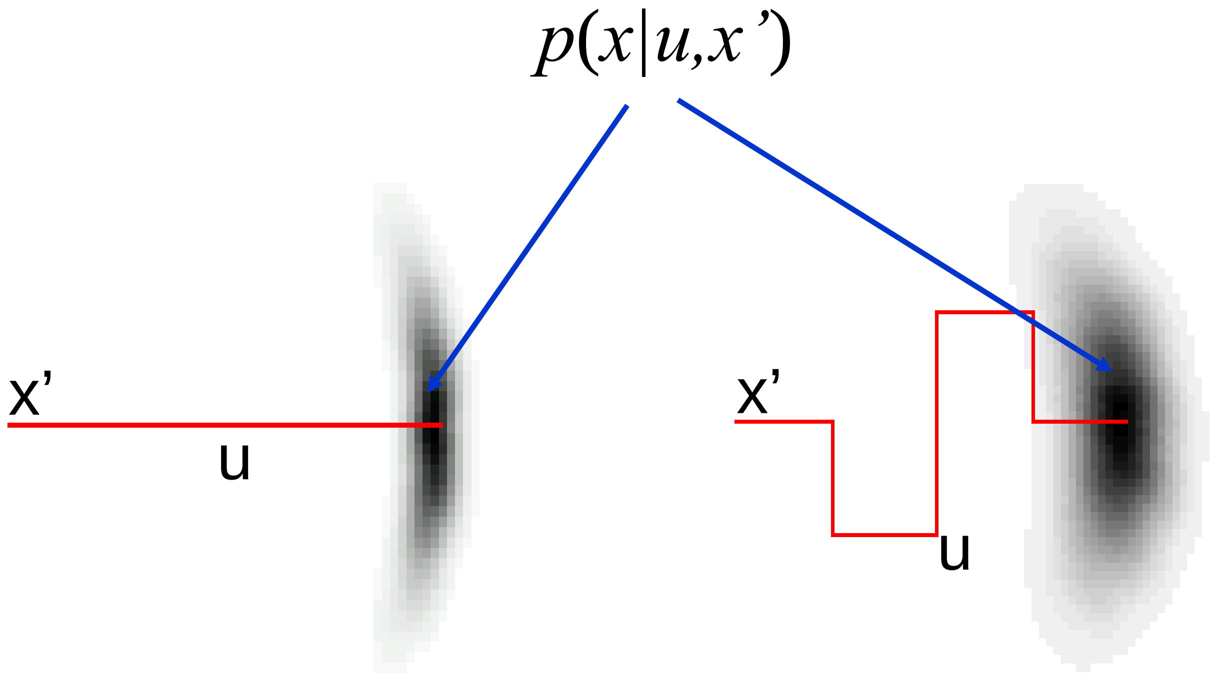
- **Given**
  - Map of the environment.
  - Sequence of sensor measurements.
- **Wanted**
  - Estimate of the robot's position.
- **Problem classes**
  - Position tracking
  - Global localization
  - Kidnapped robot problem (recovery)

Bayes Filters for Robot Localization



Probabilistic Kinematics

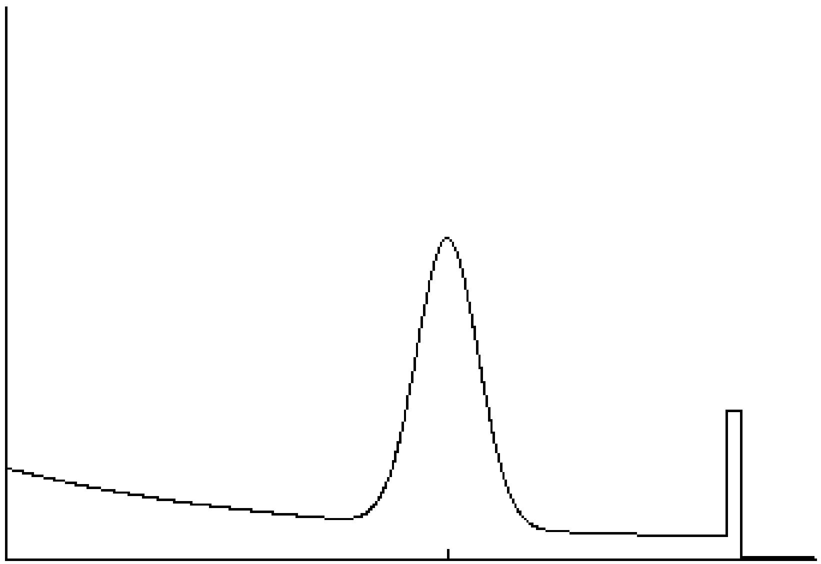
- Odometry information is inherently noisy.



Proximity Measurement

- Measurement can be caused by ...
  - a known obstacle.
  - cross-talk.
  - an unexpected obstacle (people, furniture, ...).
  - missing all obstacles (total reflection, glass, ...).
- Noise is due to uncertainty ...
  - in measuring distance to known obstacle.
  - in position of known obstacles.
  - in position of additional obstacles.
  - whether obstacle is missed.

Mixture Density

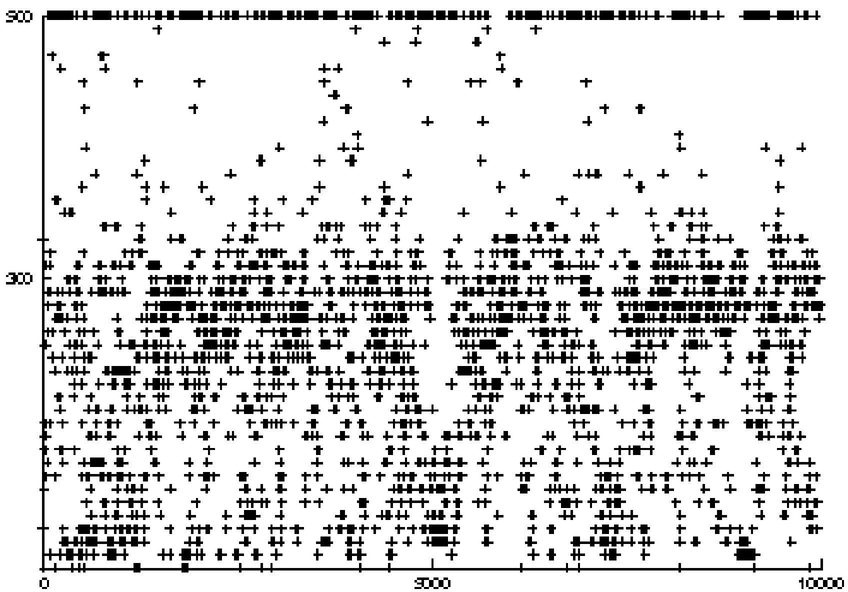


$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\text{max}}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

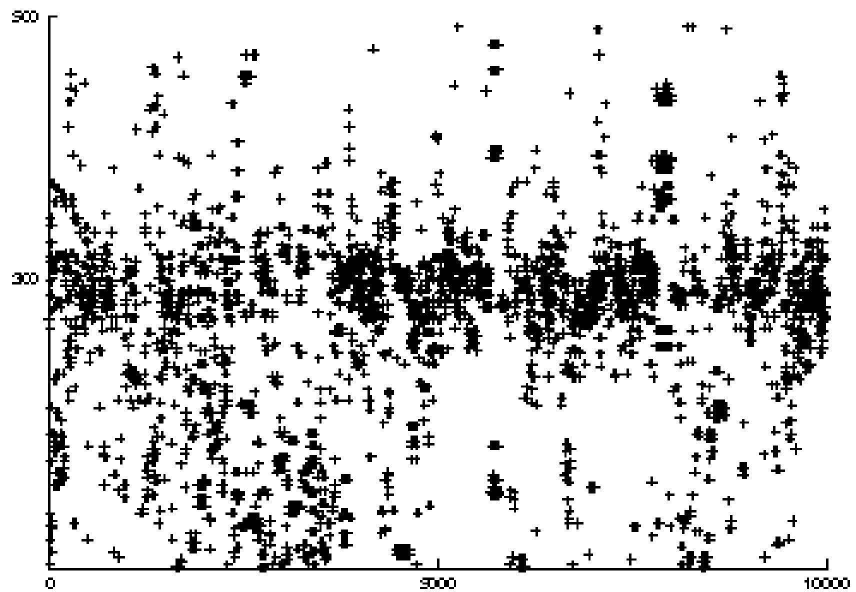
How can we determine the model parameters?

Raw Sensor Data

Measured distances for expected distance of 300 cm.

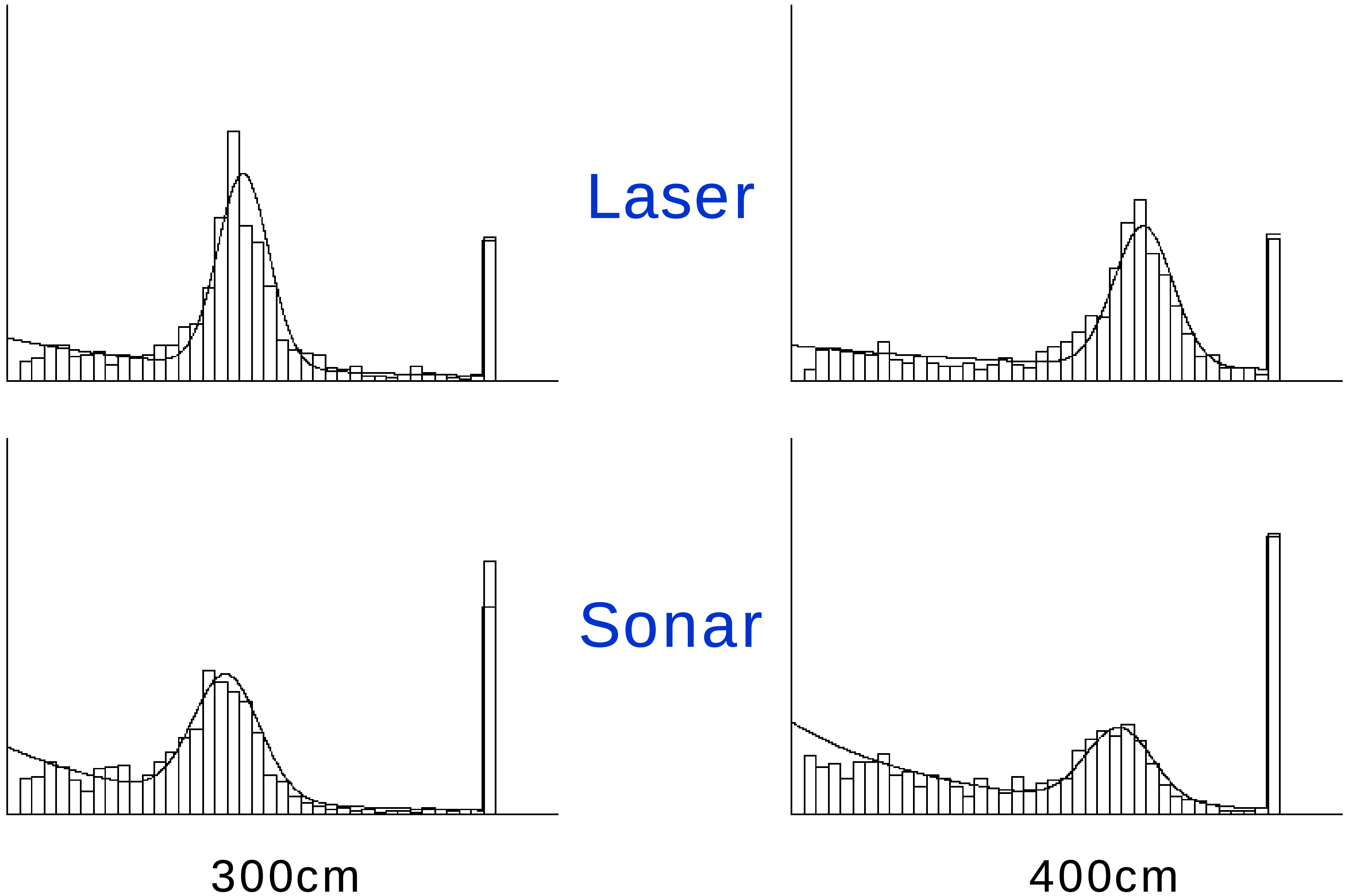


Sonar



Laser

Approximation Results



Representations for Bayesian Robot Localization

Discrete approaches ('95)
- Topological representation ('95)
  - uncertainty handling (POMDPs)
  - occas. global localization, recovery
- Grid-based, metric representation ('96)
  - global localization, recovery

Particle filters ('99)
- sample-based representation
- global localization, recovery

AI

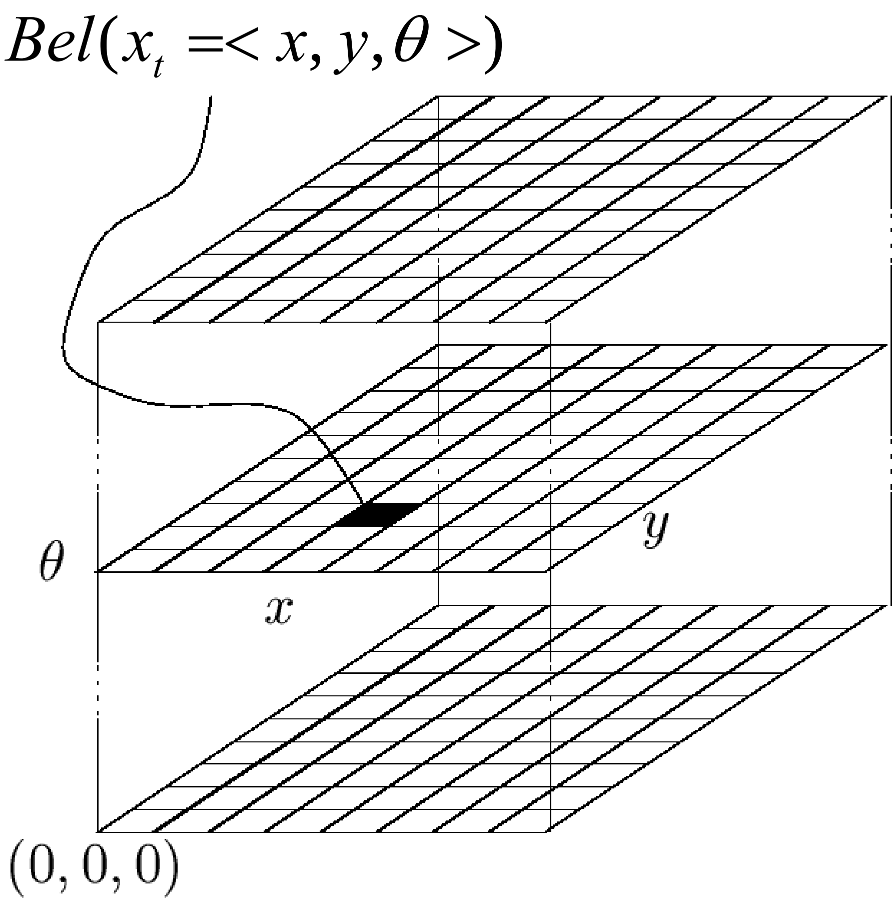
Kalman filters (late-80s?)
- Gaussians
- approximately linear models
- position tracking

Robotics

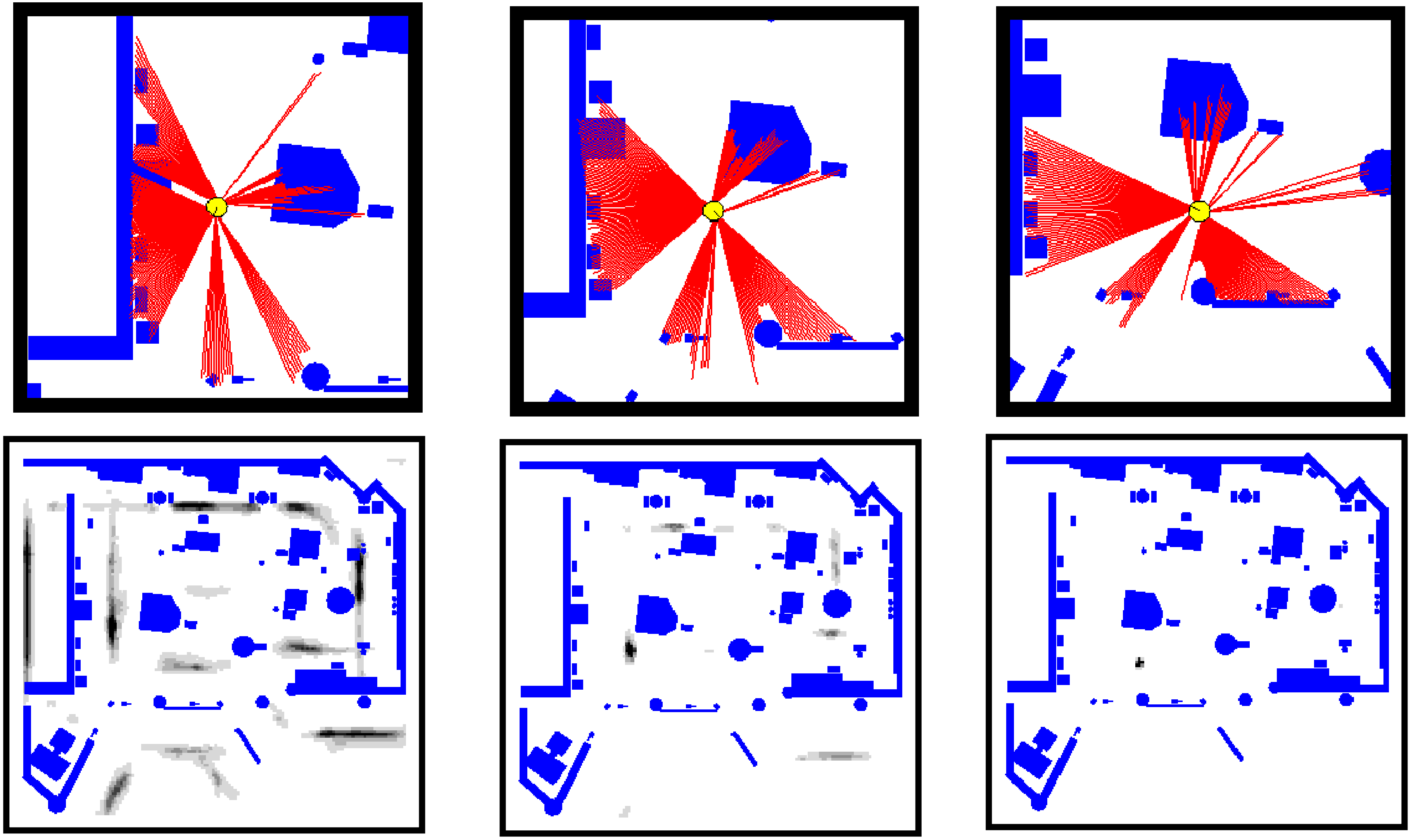
Multi-hypothesis ('00)
- multiple Kalman filters
- global localization, recovery

Discrete Grid Filters

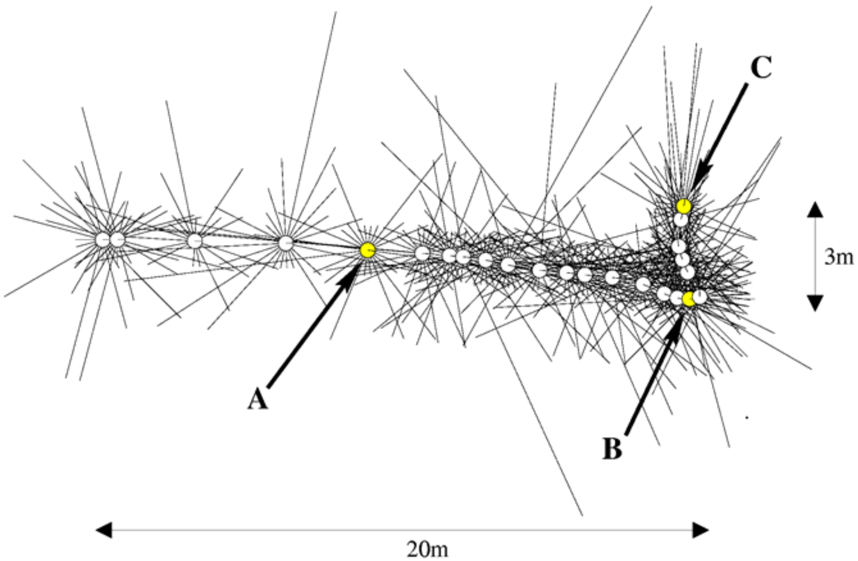
Piecewise Constant Representation

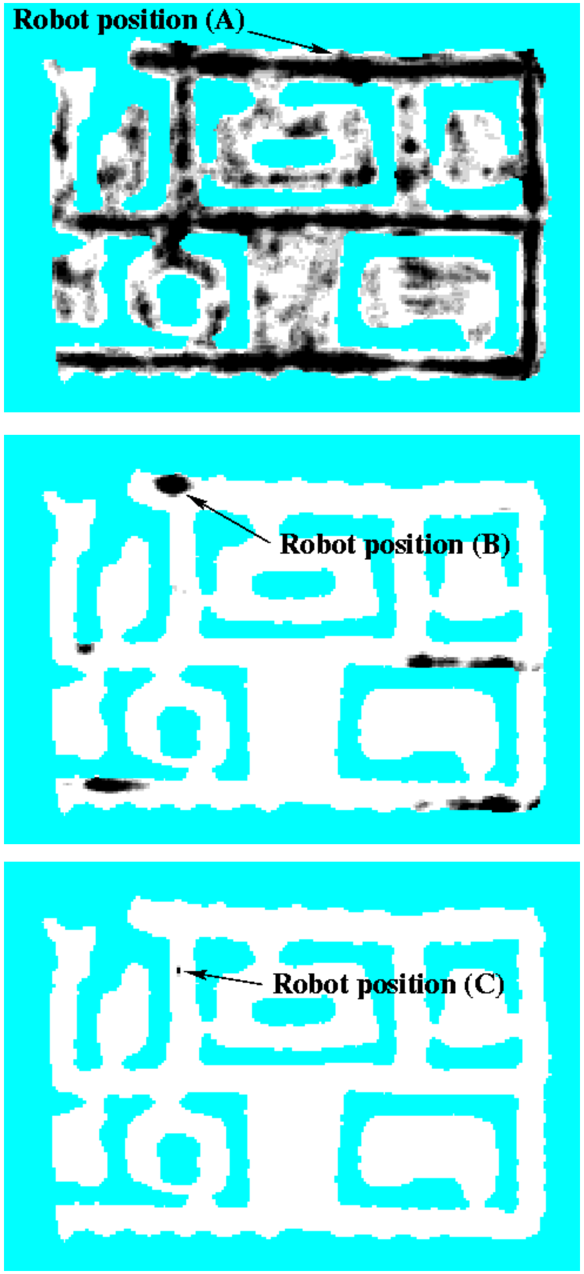


Grid-based Localization



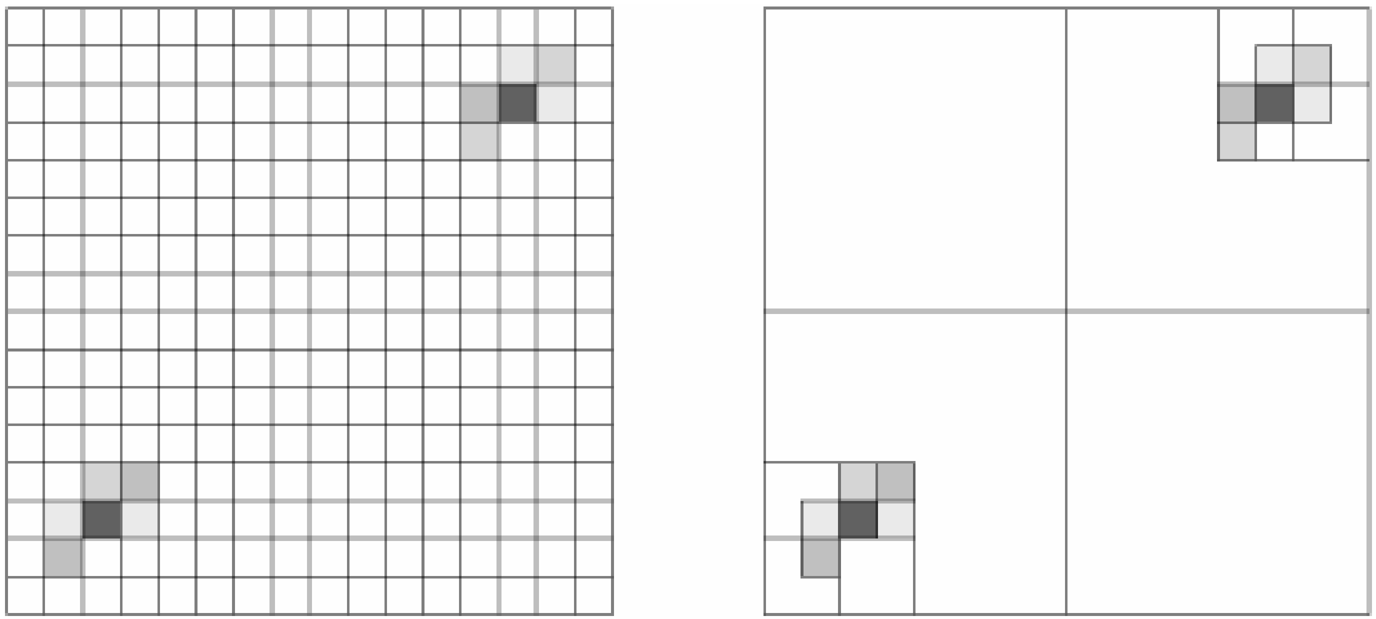
Sonars and Occupancy Grid Map





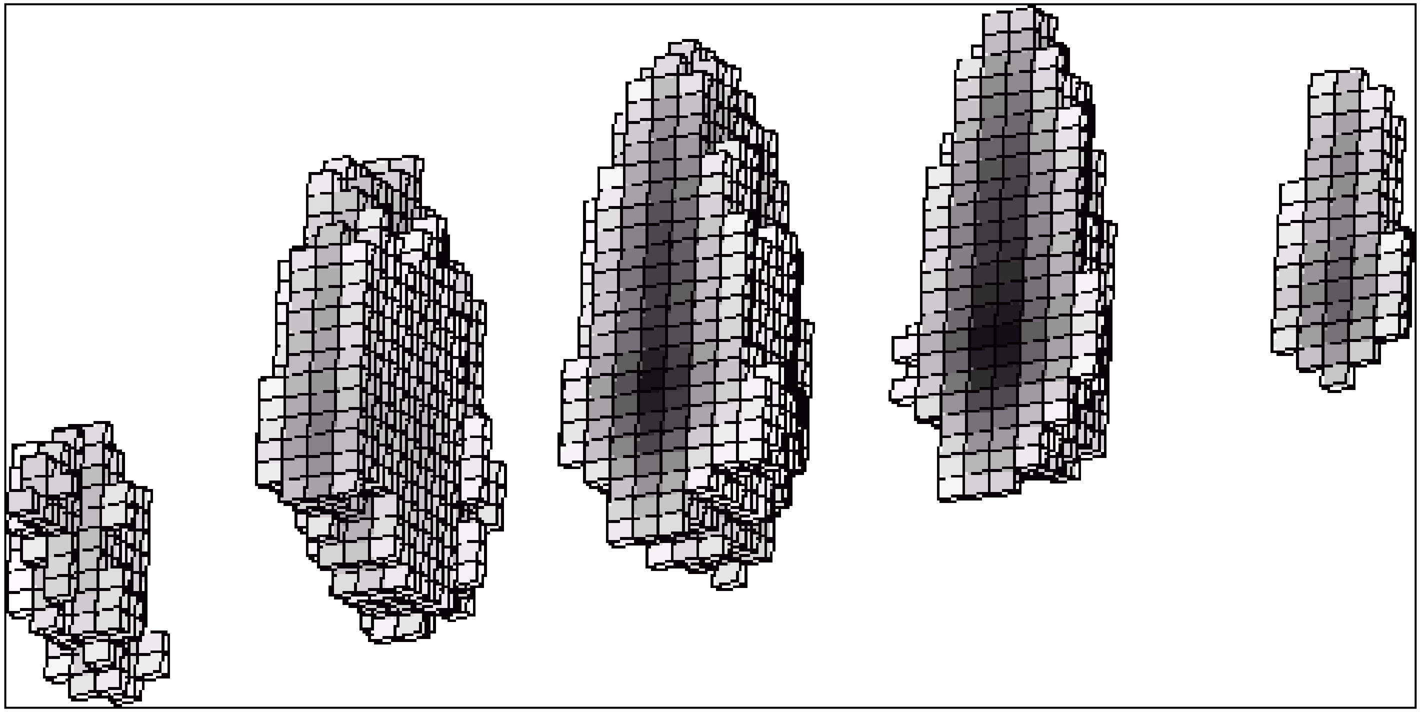
Tree-based Representation

Idea: Represent density using a variant of Octrees



Tree-based Representations

- Efficient in space and time
- Multi-resolution



Particle Filters

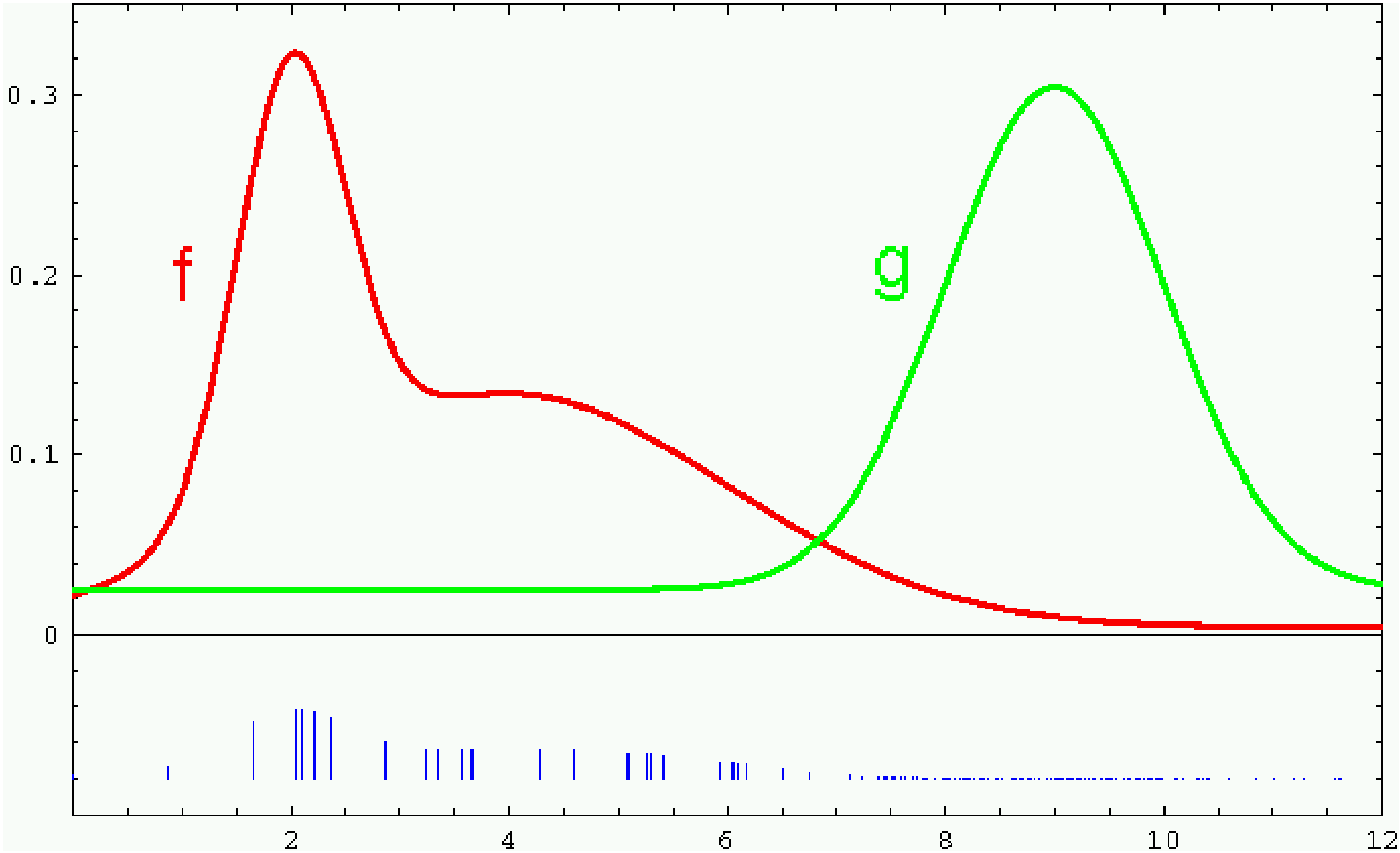
Particle Filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]d

Importance Sampling



Weight samples: $w = f / g$

Particle Filter Algorithm

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

draw x^i_{t-1} from $Bel(x_{t-1})$

draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

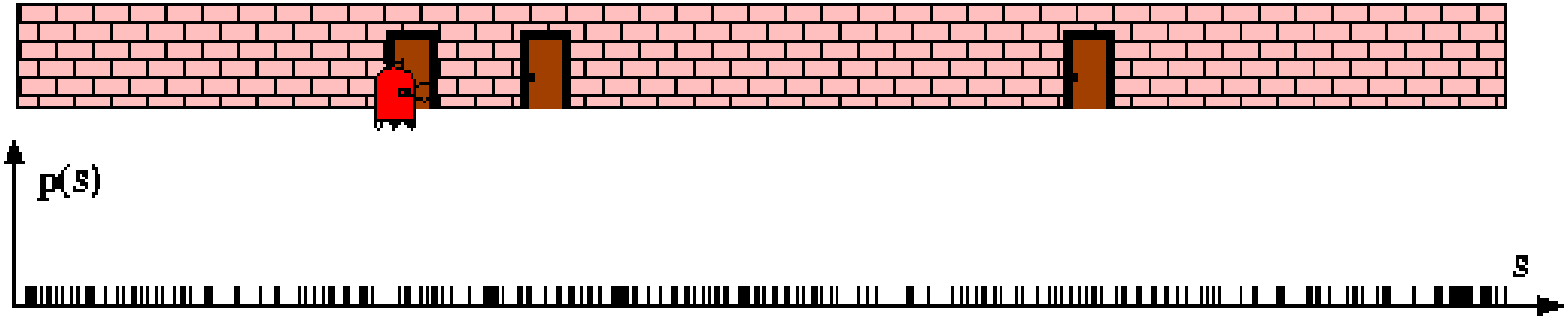
Importance factor for x^i_t:

$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}}$$

$$= \frac{\eta\ p(z_t \mid x_t)\ p(x_t \mid x_{t-1}, u_{t-1})\ Bel\ (x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\ Bel\ (x_{t-1})}$$

$$\propto p(z_t \mid x_t)$$

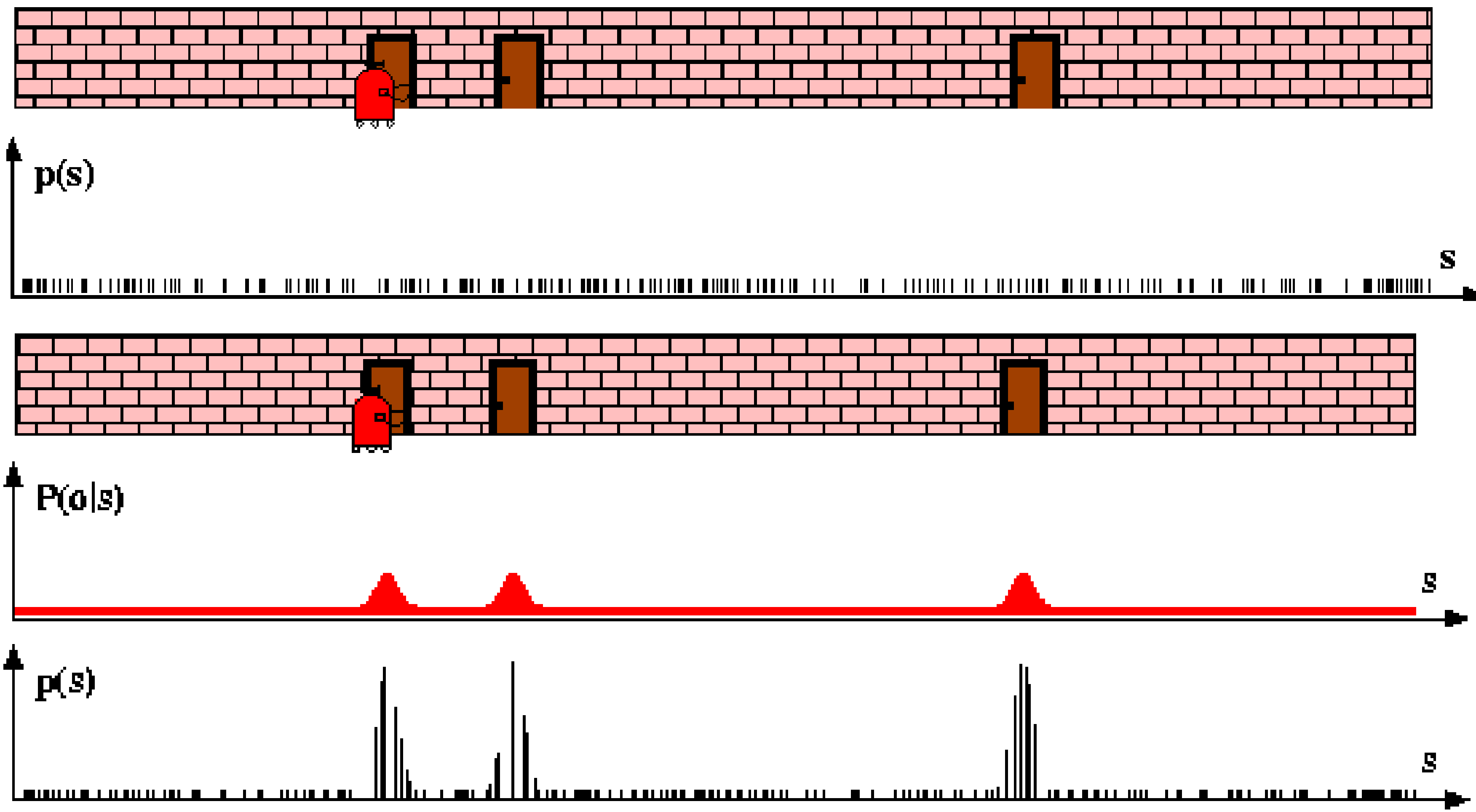
Particle Filters



Sensor Information: Importance Sampling

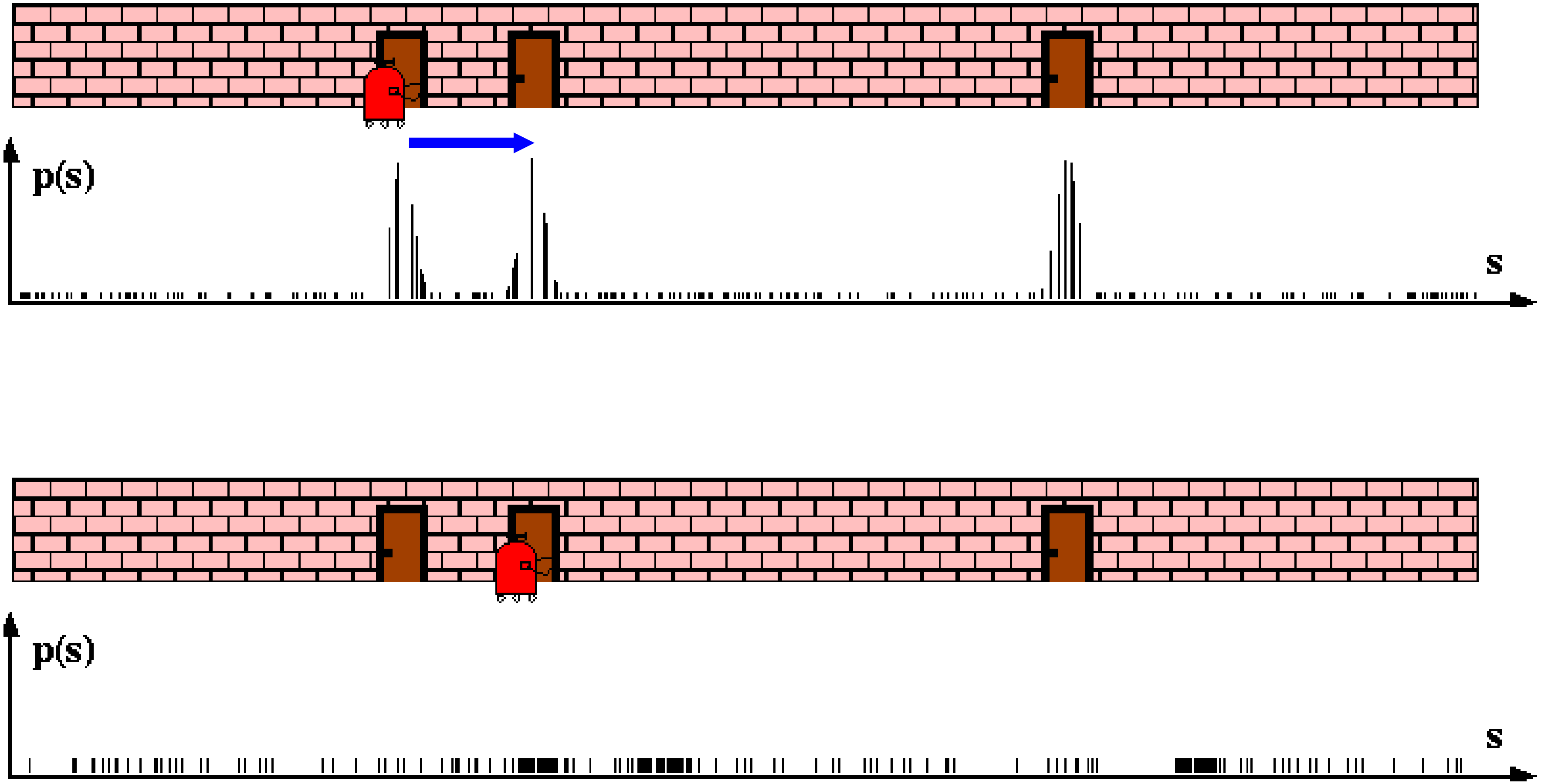
$$Bel(x) \quad \leftarrow \quad \alpha \ p(z\,|\,x)\,Bel^{-}(x)$$

$$w \quad \leftarrow \quad \frac{\alpha \ p(z\,|\,x)\,Bel^{-}(x)}{Bel^{-}(x)} \quad = \quad \alpha \ p(z\,|\,x)$$



Robot Motion

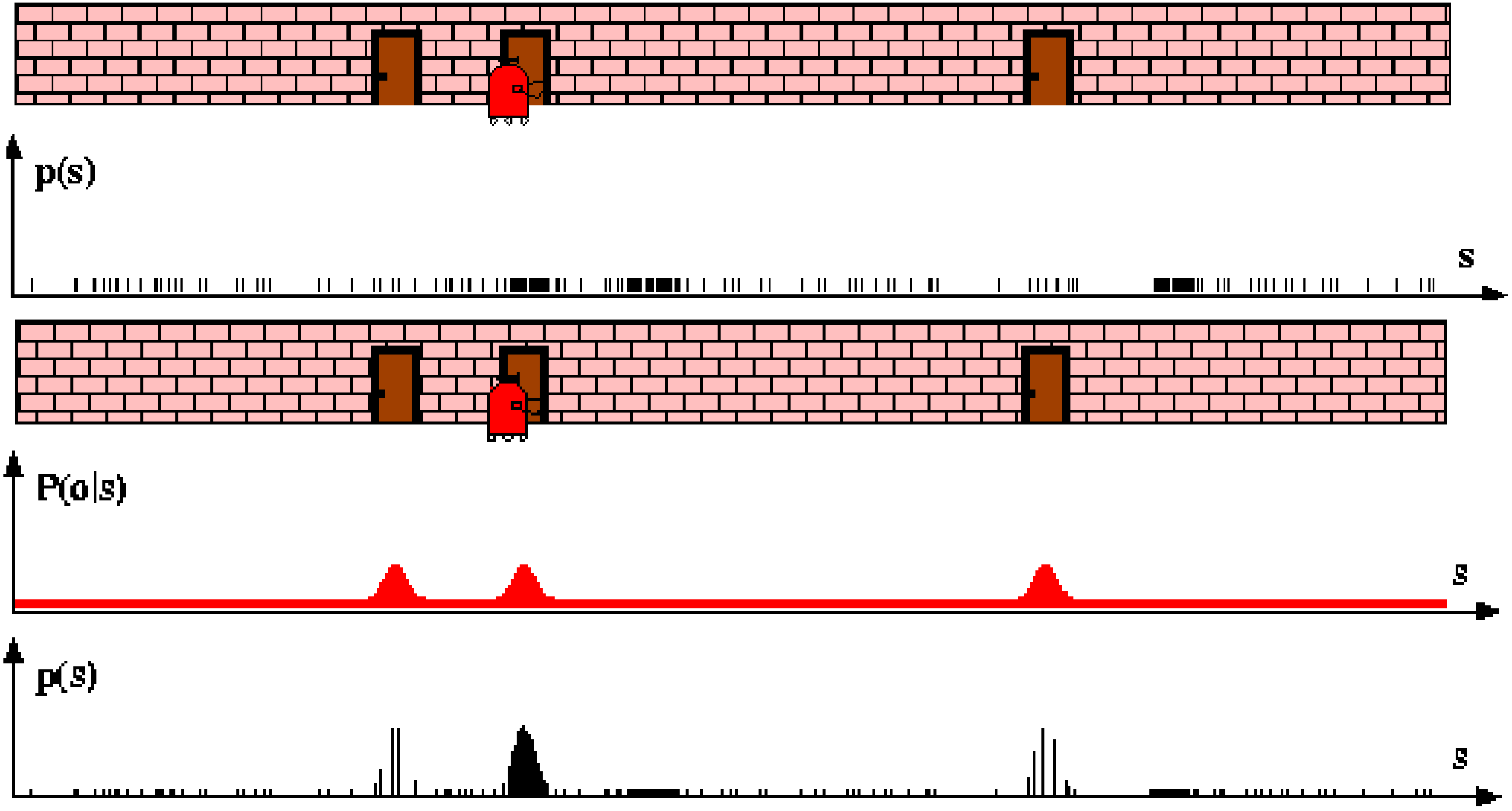
$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



Sensor Information: Importance Sampling

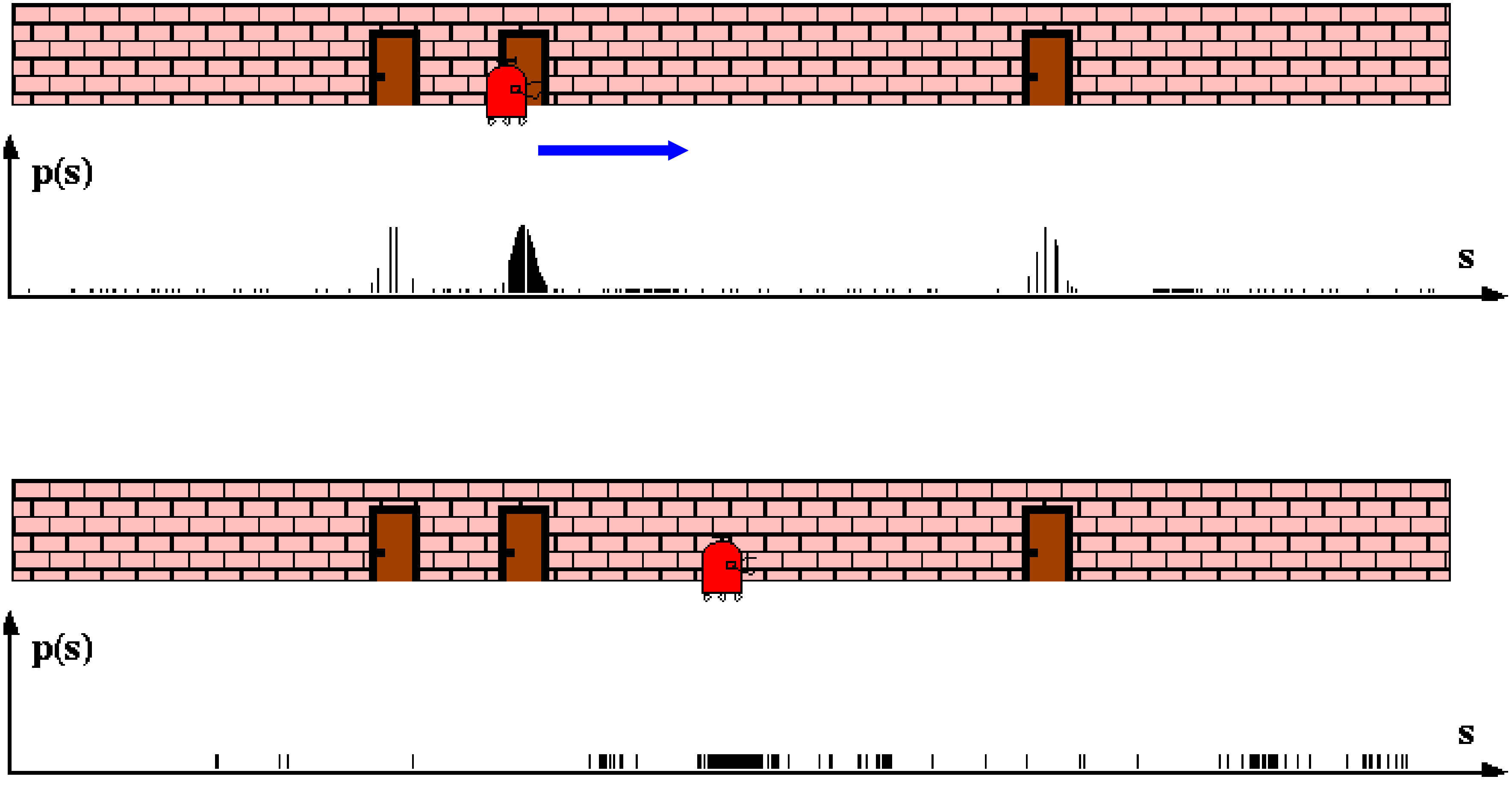
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^-(x)$$

$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^-(x)}{Bel^-(x)} = \alpha\, p(z \mid x)$$

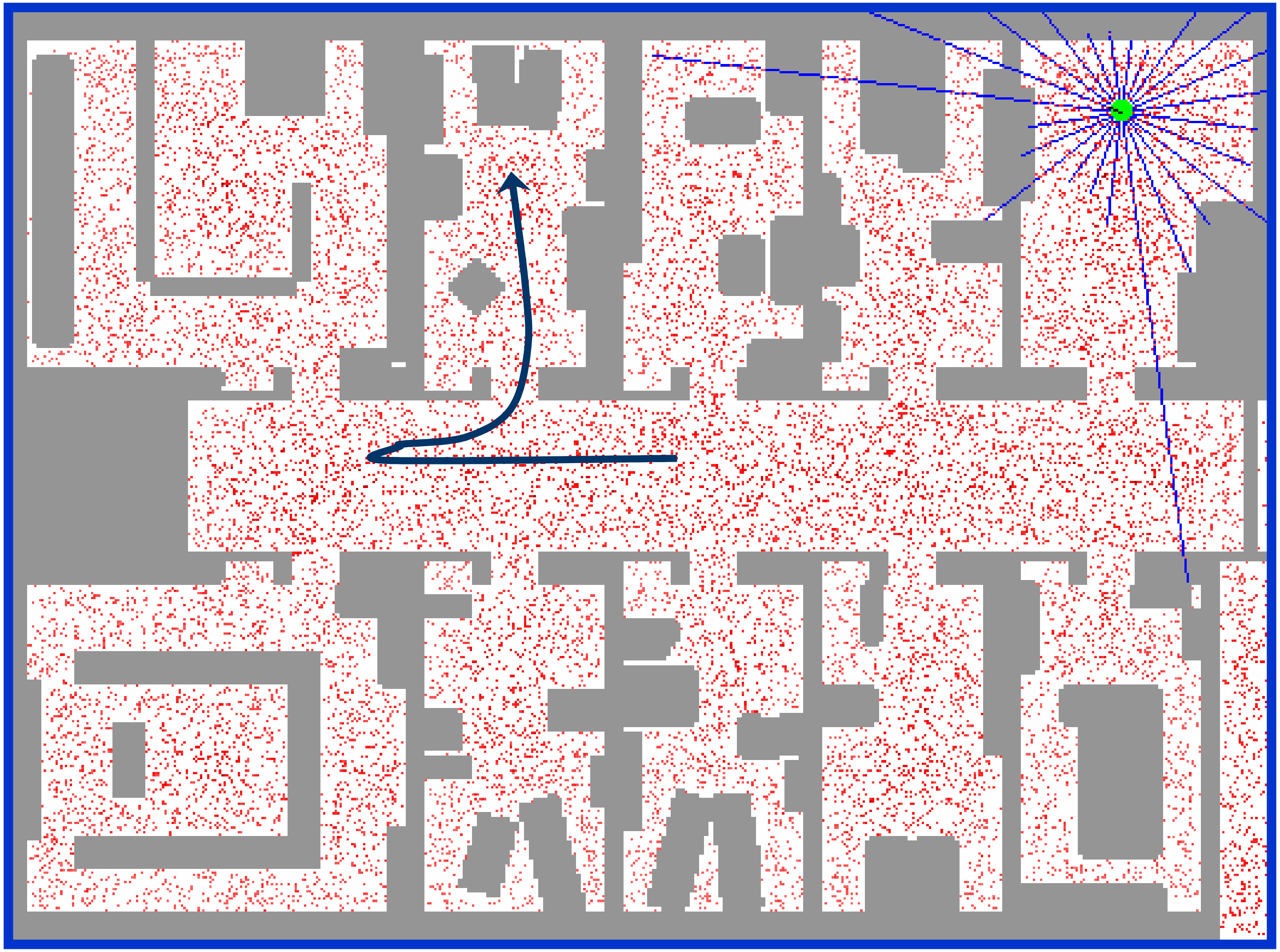


Robot Motion

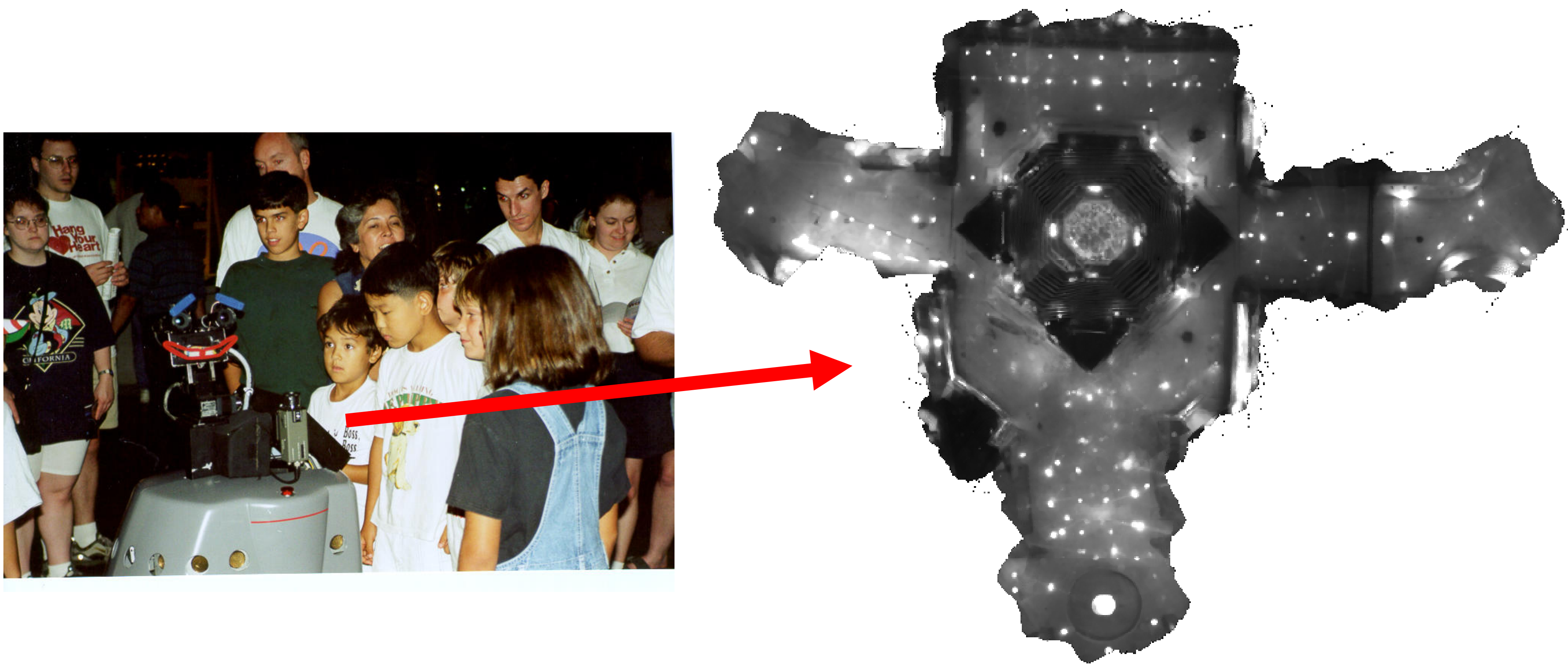
$$Bel^{-}(x) \quad \leftarrow \quad \int p(x \mid u, x')\, Bel(x')\ \mathrm{d}x'$$



Sample-based Localization (sonar)

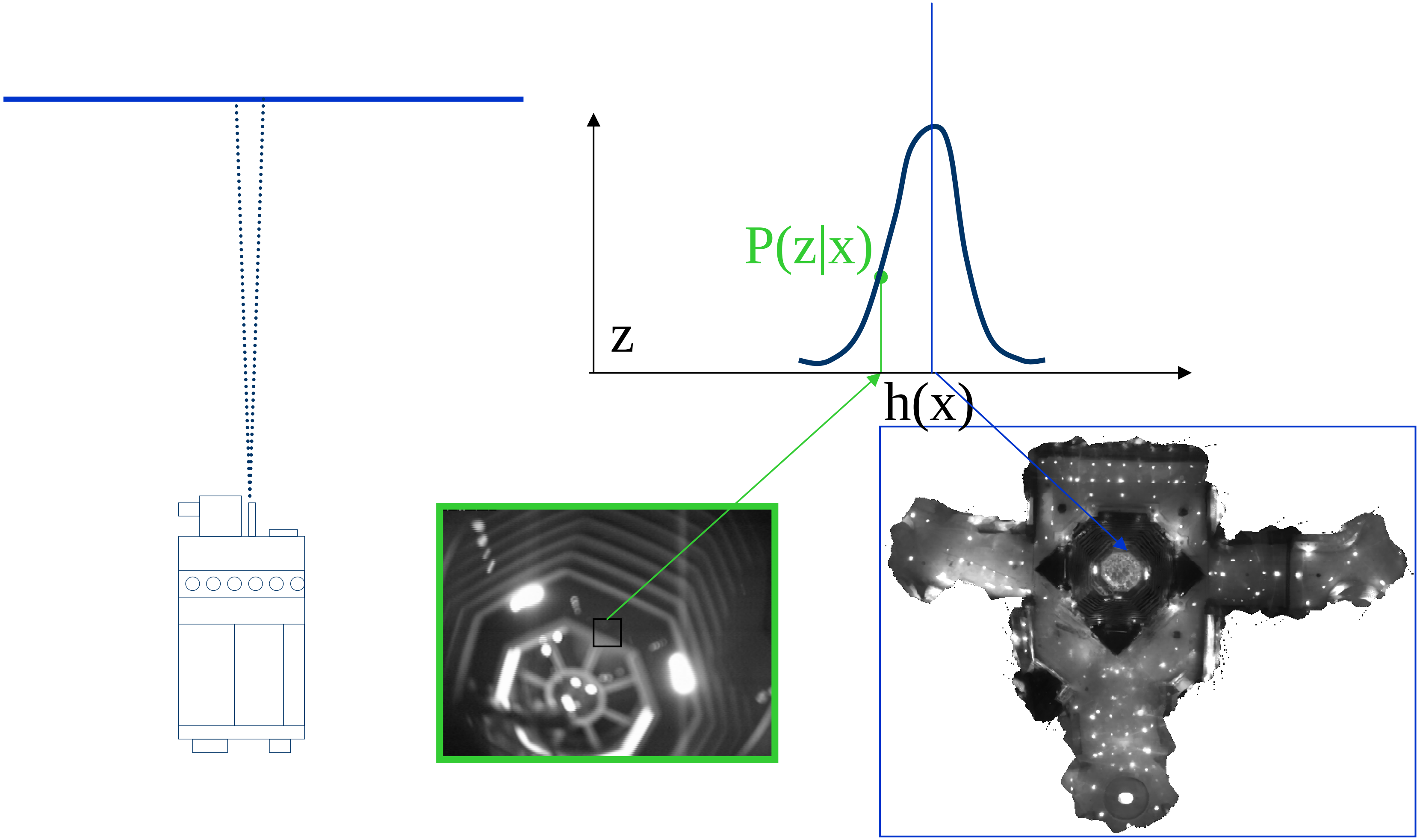


Using Ceiling Maps for Localization



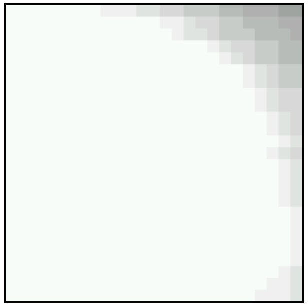
[Dellaert et al. 99]

Vision-based Localization

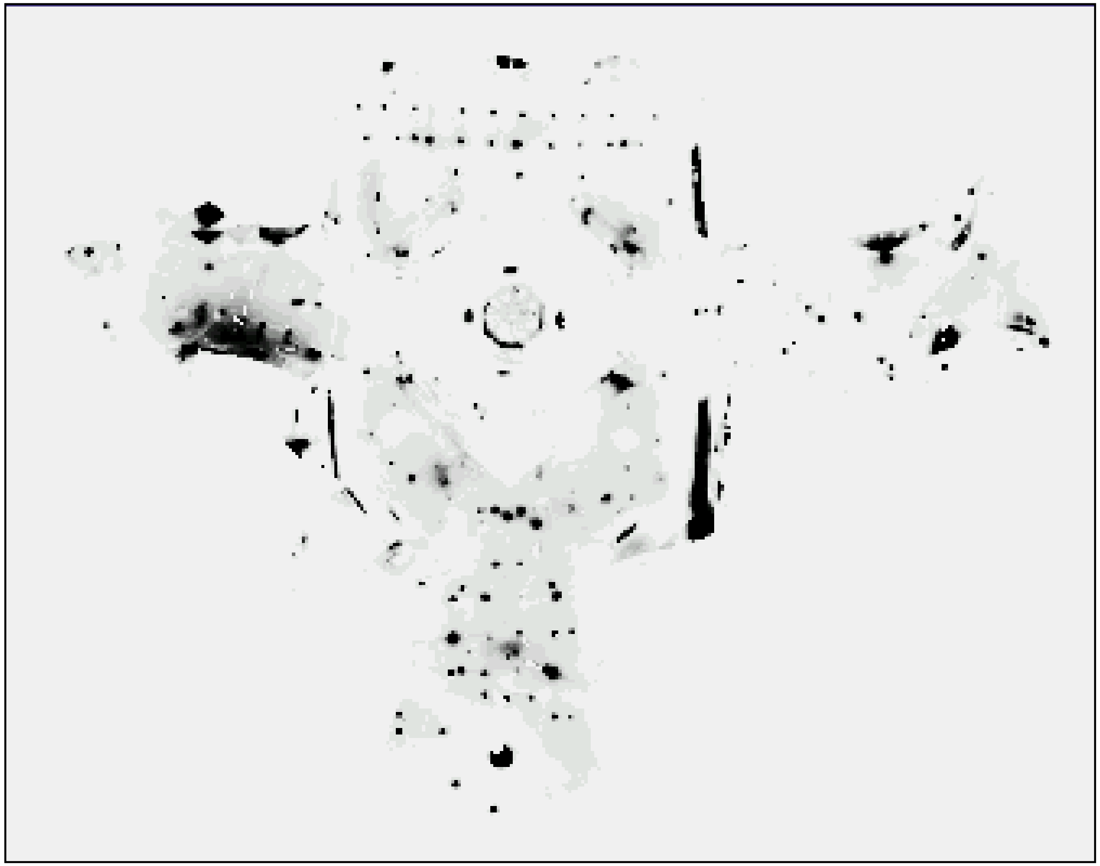


Under a Light

Measurement z:

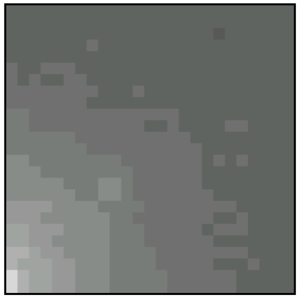


***P(z|x)*:**

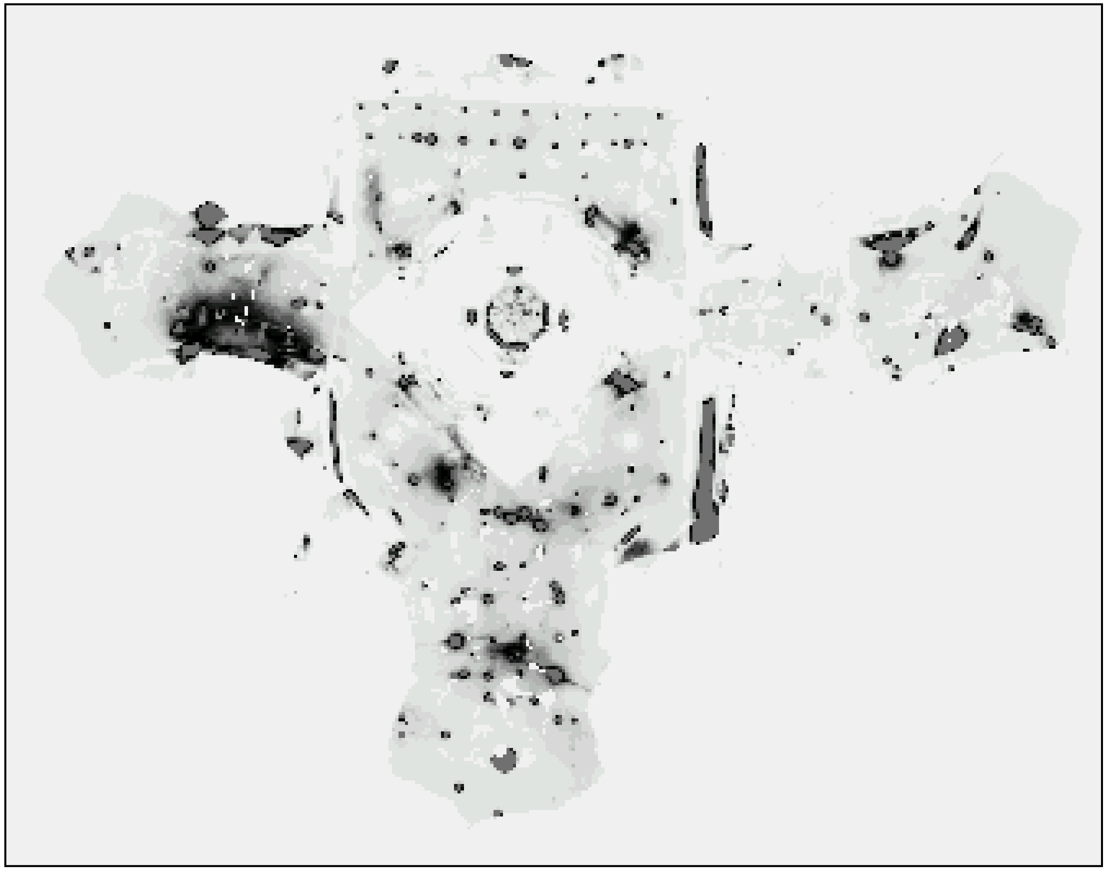


Next to a Light

Measurement z:



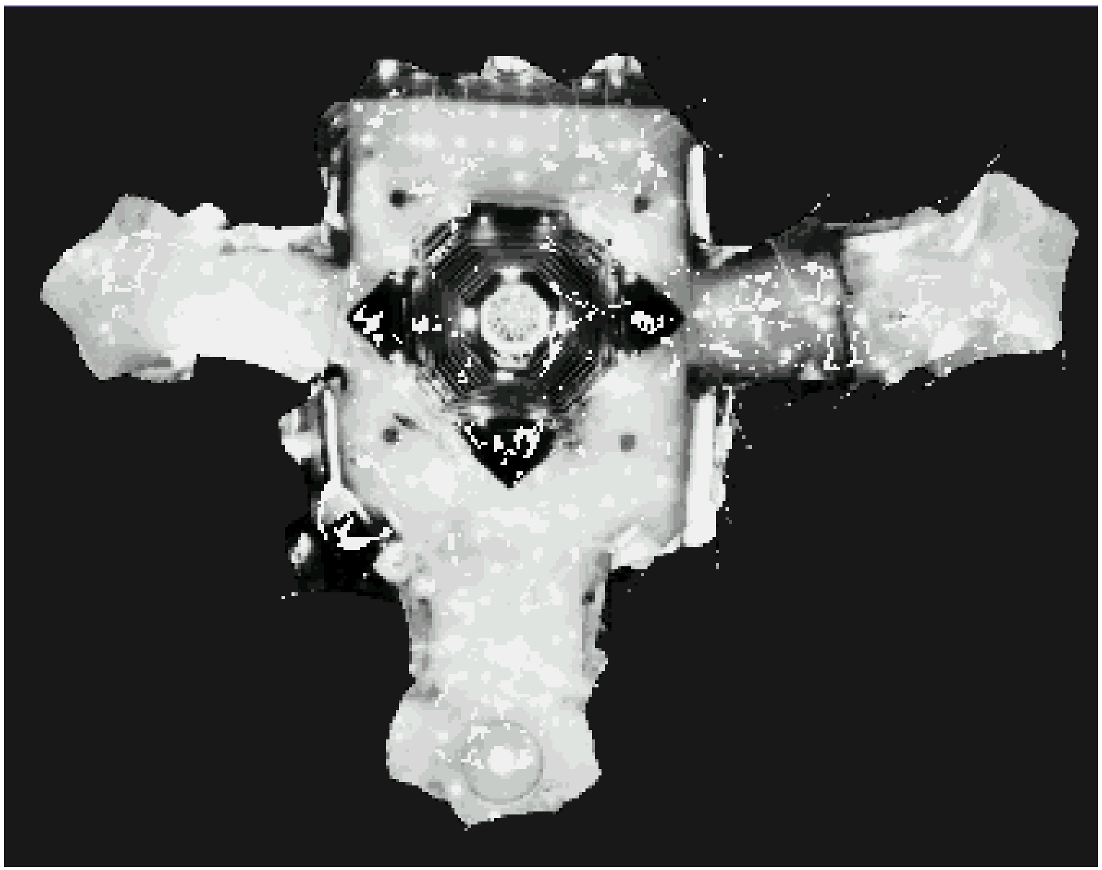
$P(z|x)$:



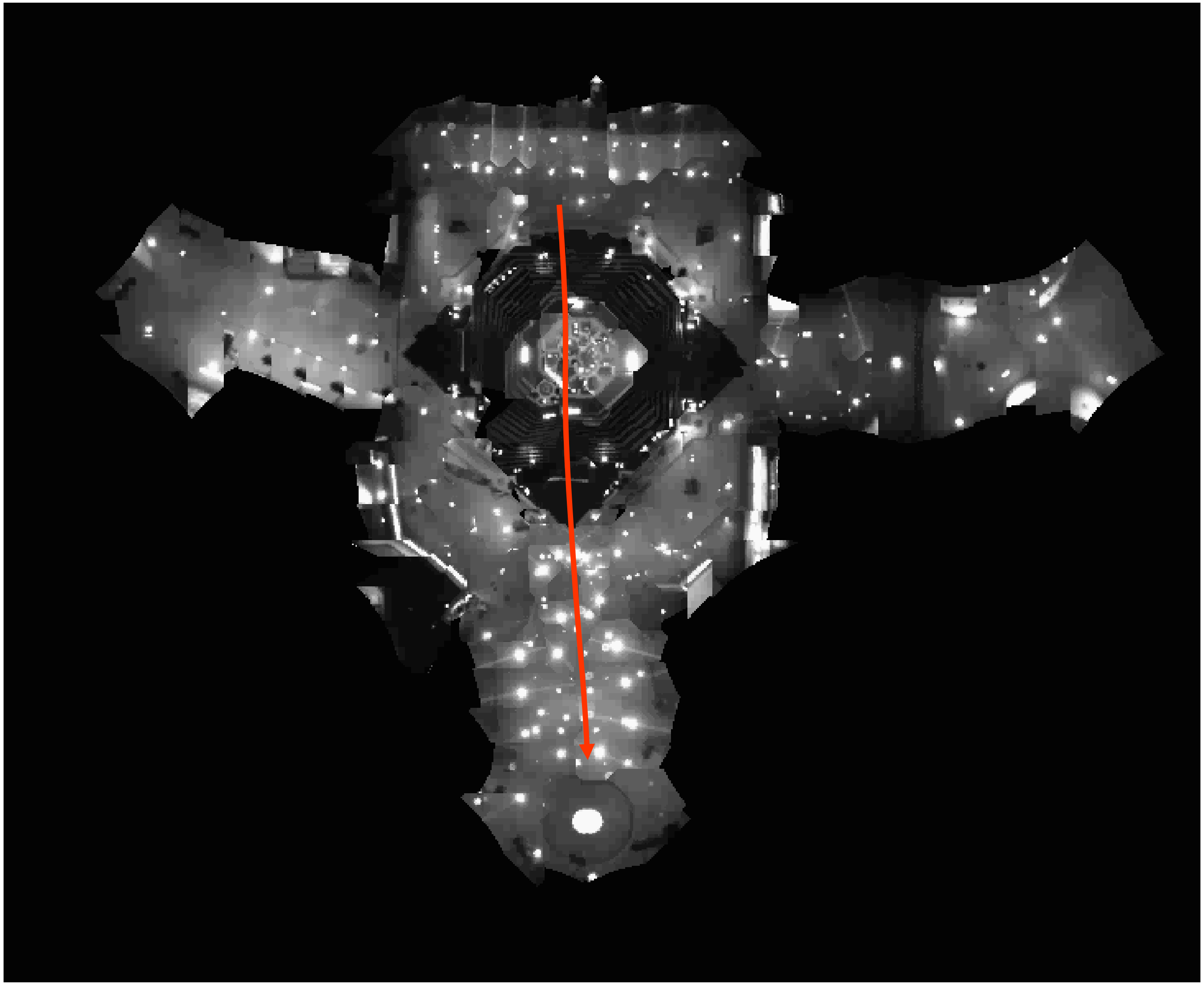
Elsewhere

Measurement z:

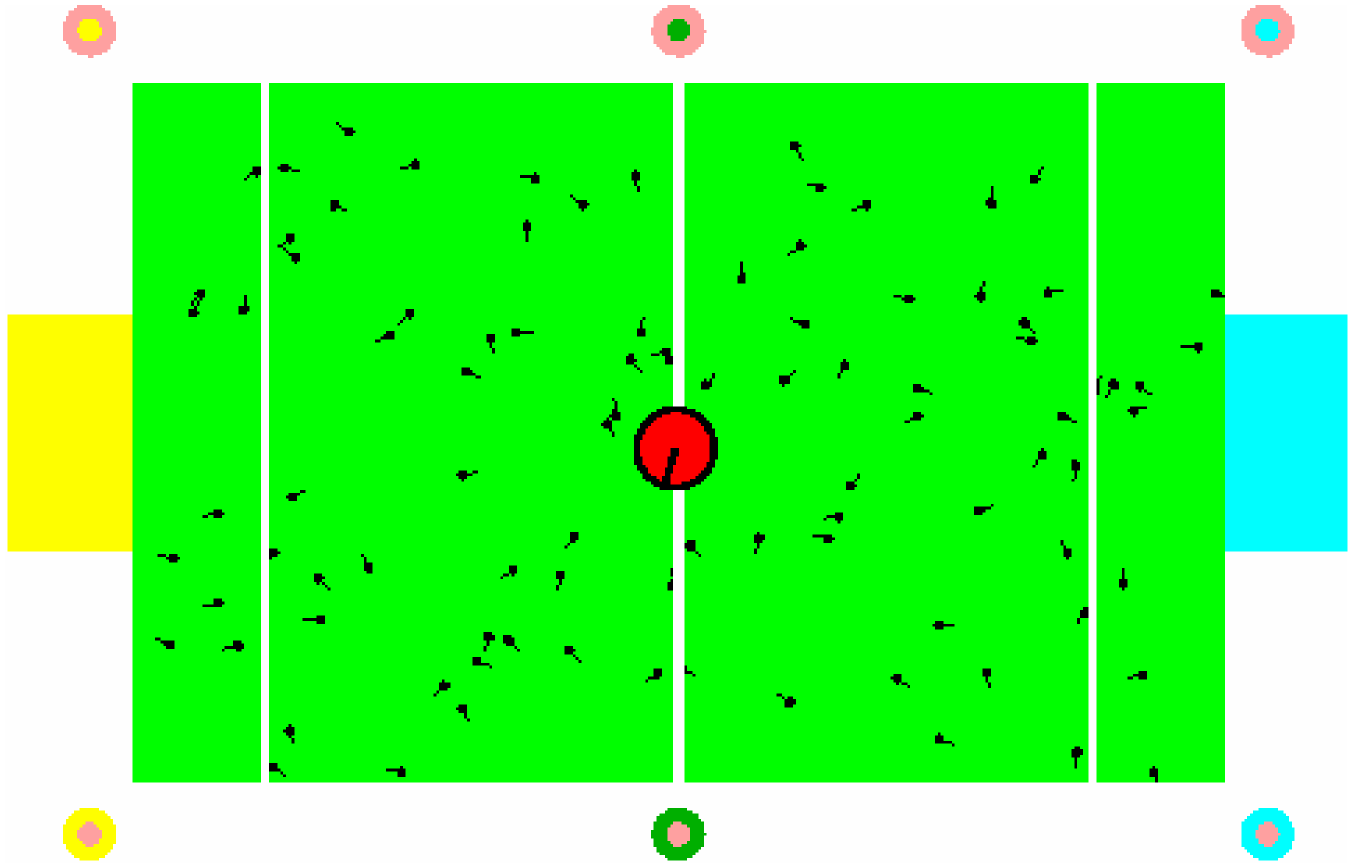
***P(z|x)*:**



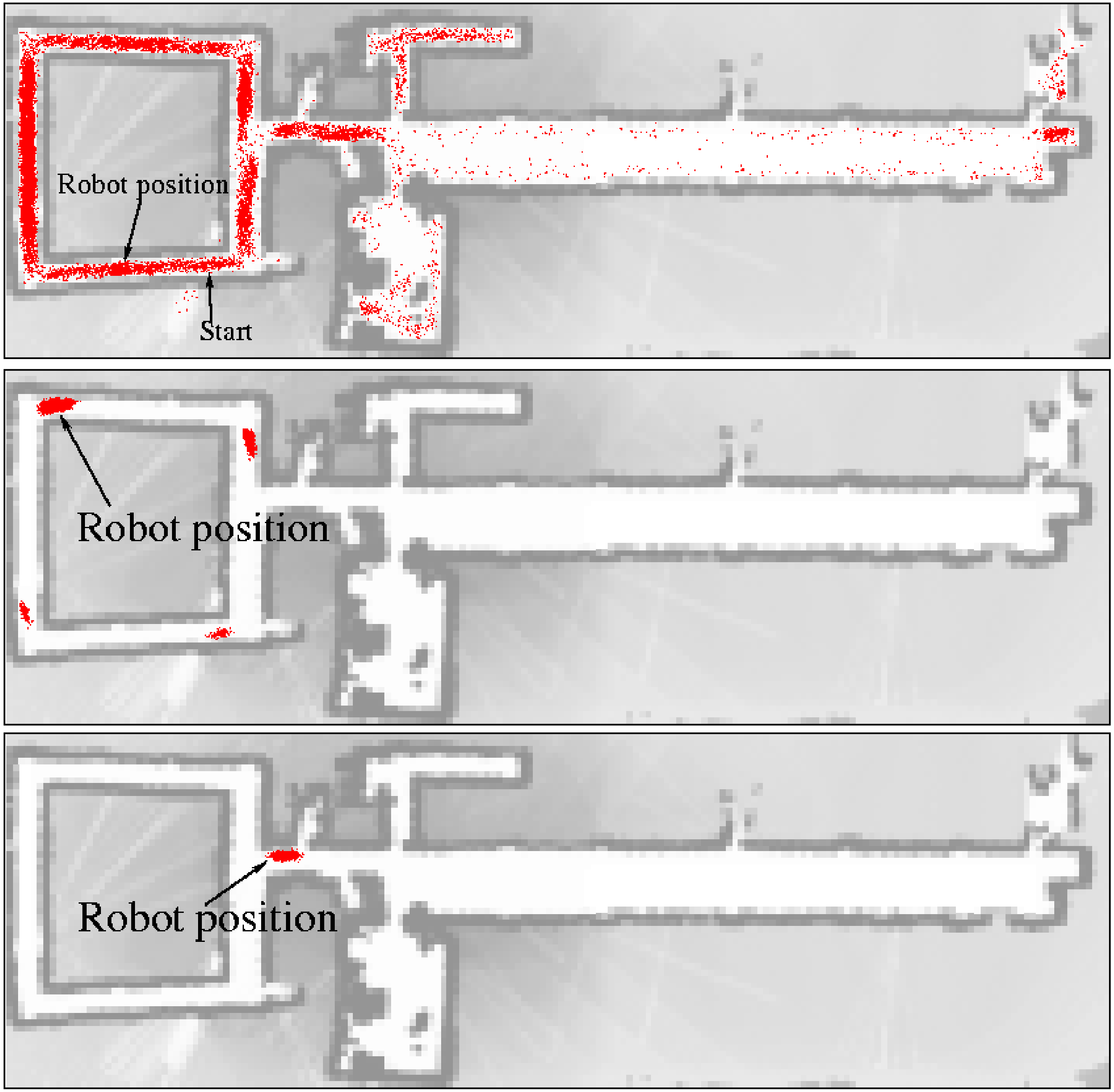
Global Localization Using Vision



Localization for AIBO robots



Adaptive Sampling



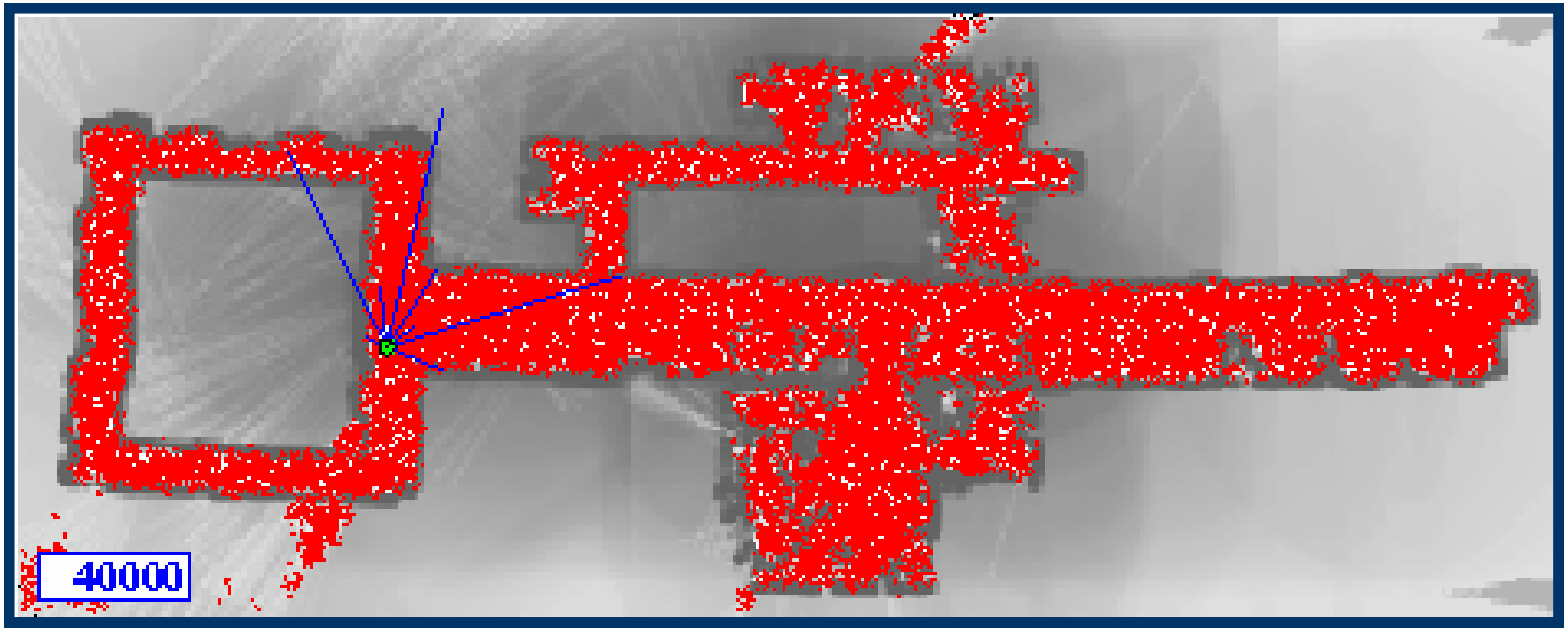
KLD-sampling

- **Idea**:
  - Assume we know the true belief.
  - Represent this belief as a multinomial distribution.
  - Determine number of samples such that we can guarantee that, with probability (1- δ*)*, the KL-distance between the true posterior and the sample-based approximation is less than ε.

- **Observation**:
  - For fixed δ and ε, number of samples only depends on number k of bins with support:

$$n = \frac{1}{2\varepsilon}\mathrm{X}^2(k-1,1-\delta) \cong \frac{k-1}{2\varepsilon}\left\{1-\frac{2}{9(k-1)}+\sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3$$

Example Run Sonar



Example Run Laser

Kalman Filters

Bayes Filter Reminder

- Prediction

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

- Correction

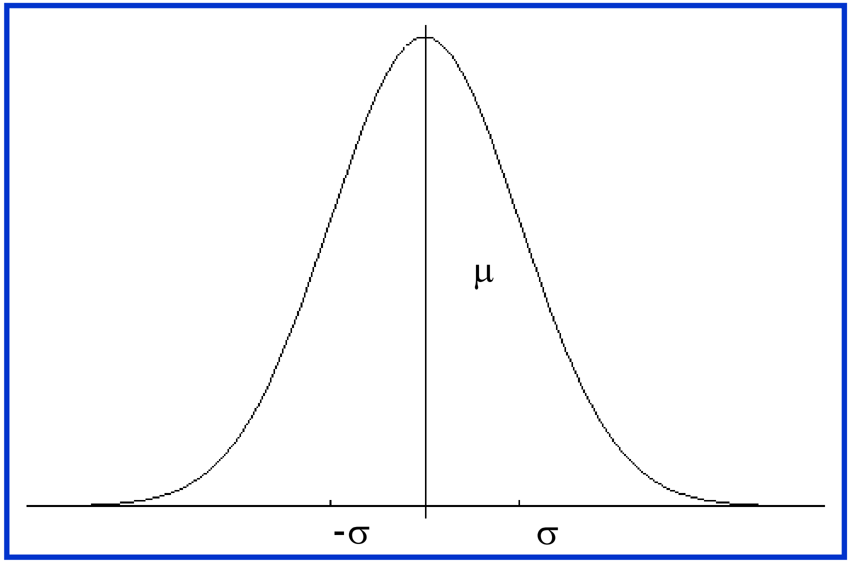
$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

Gaussians

$$p(x) \sim N(\mu, \sigma^2):$$

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

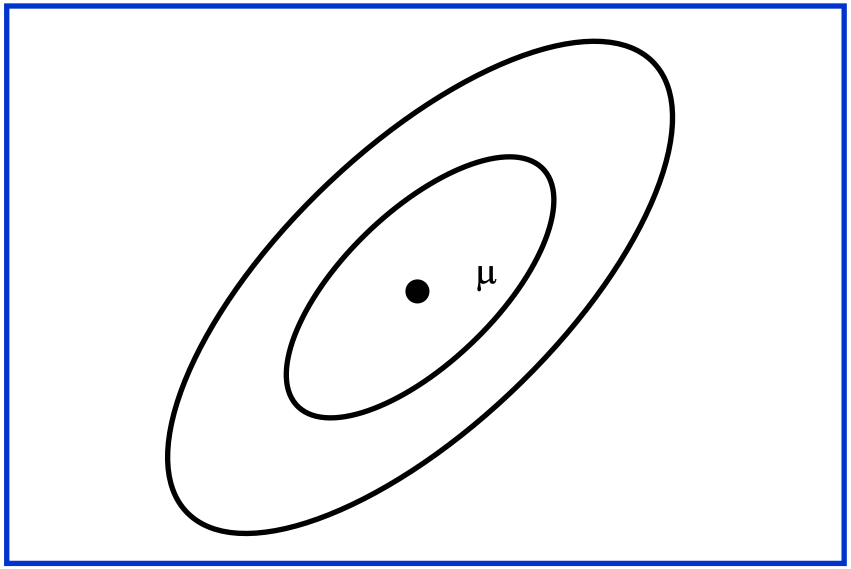
Univariate



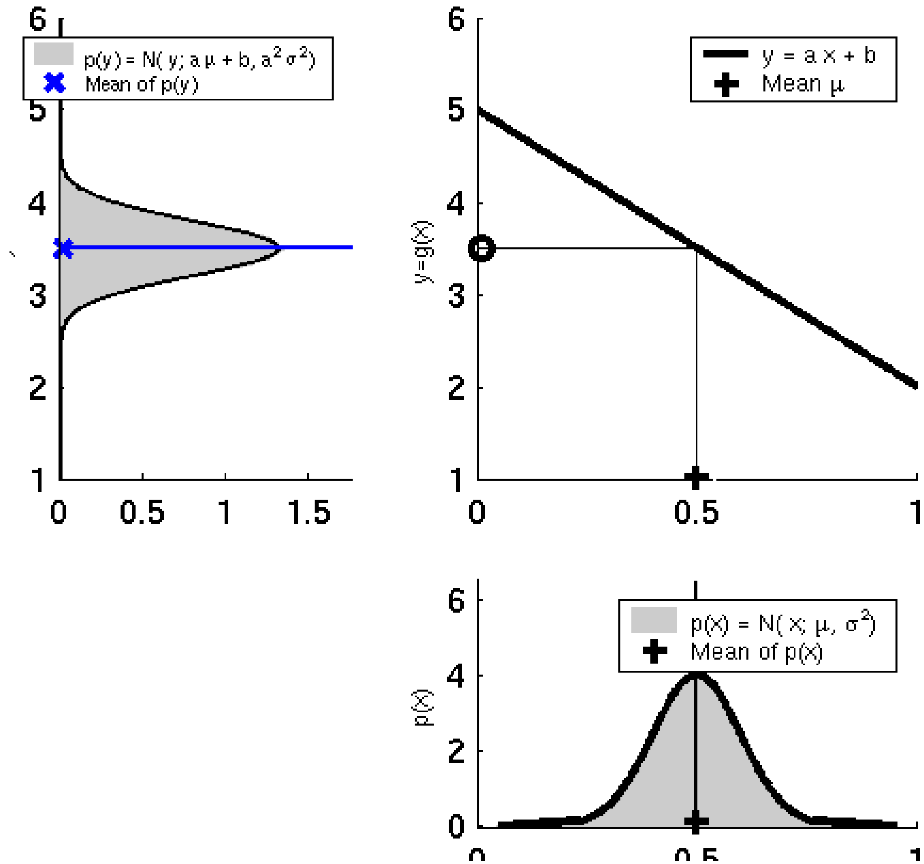
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate



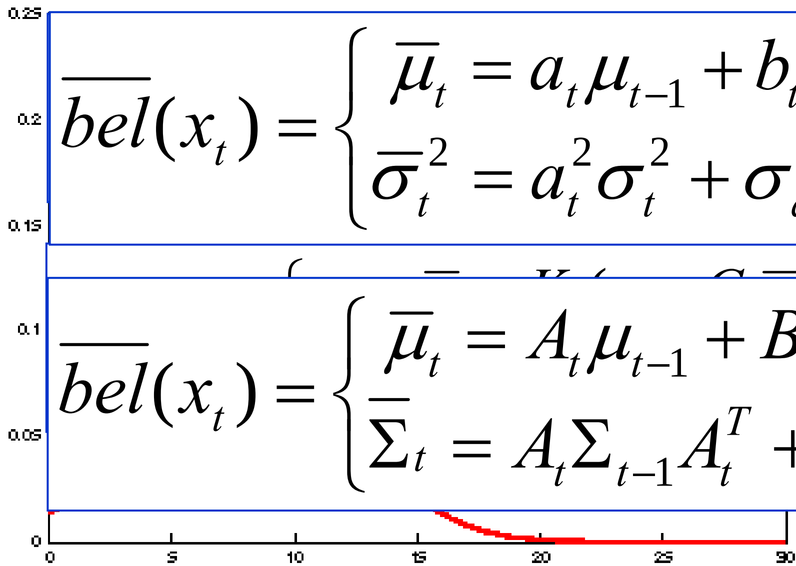
Gaussians and Linear Functions

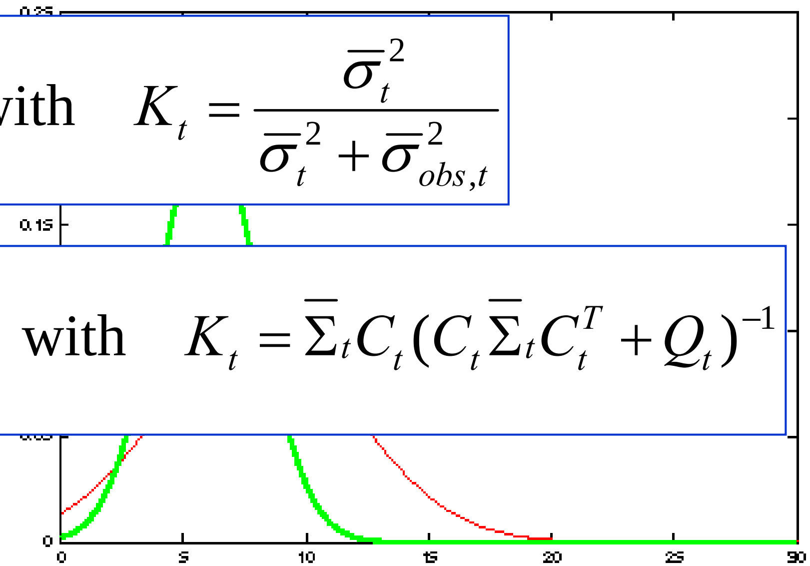


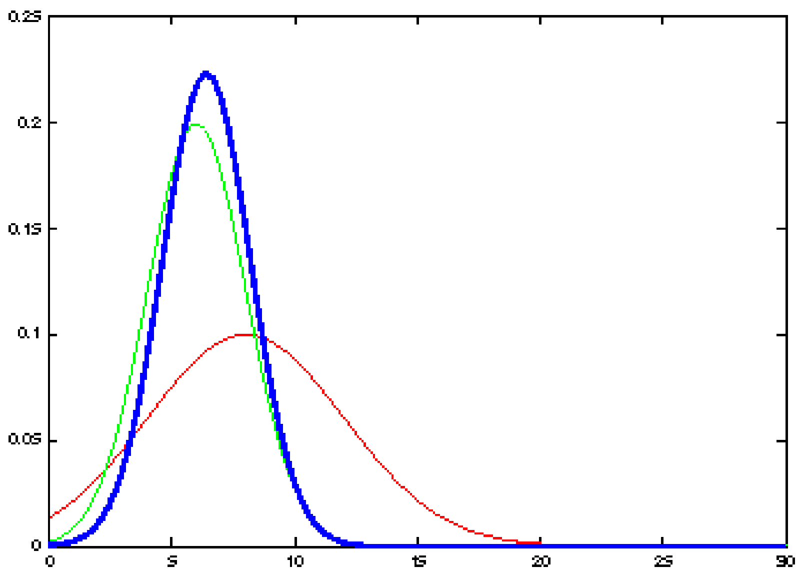
Kalman Filter Updates in 1D

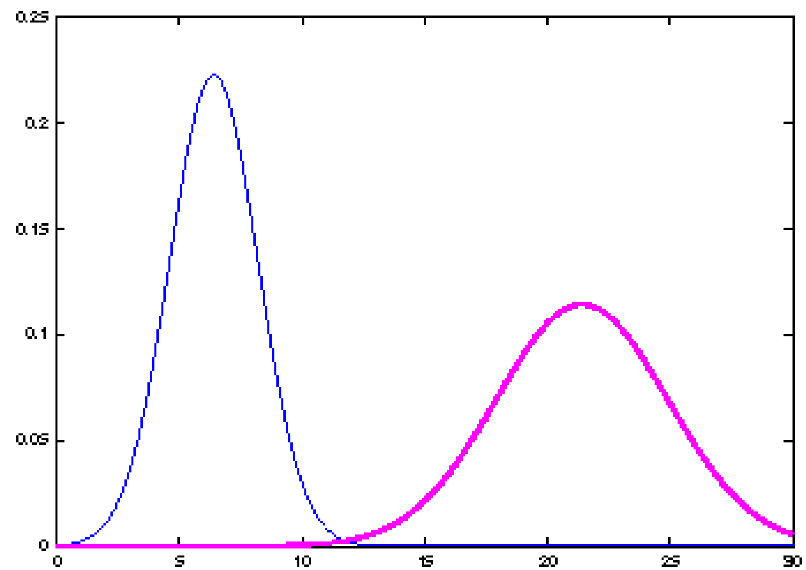
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases} \quad \text{with} \quad K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$









Kalman Filter Algorithm

1. Algorithm **Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t C_t (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9. Return μ_t, Σ_t

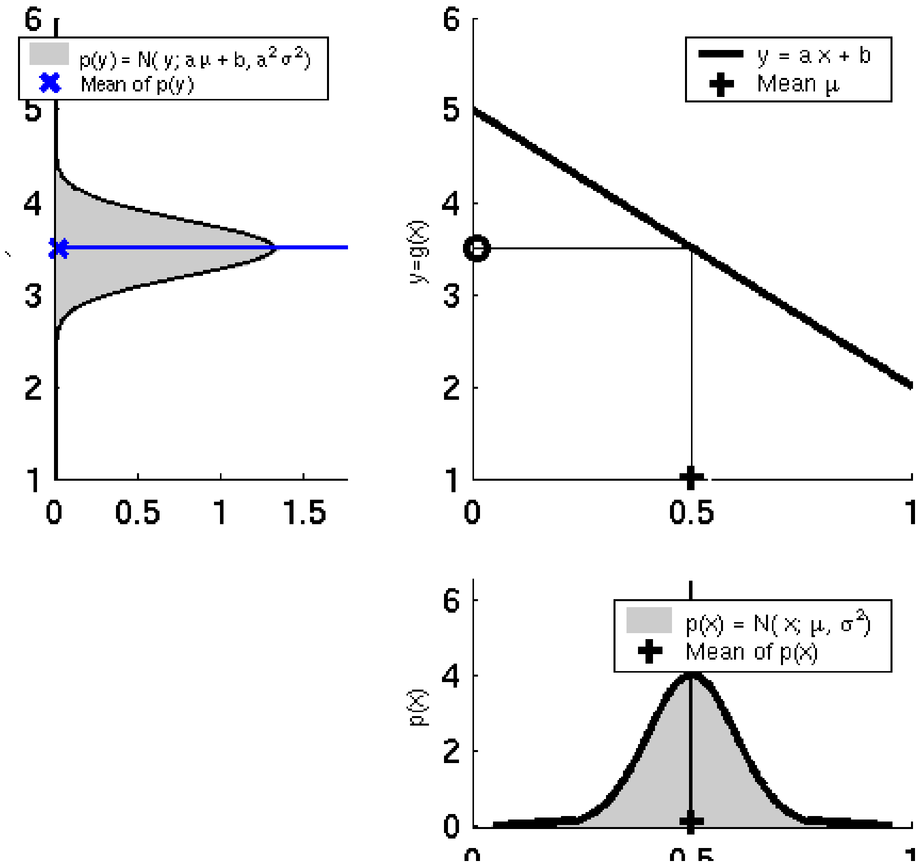
Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

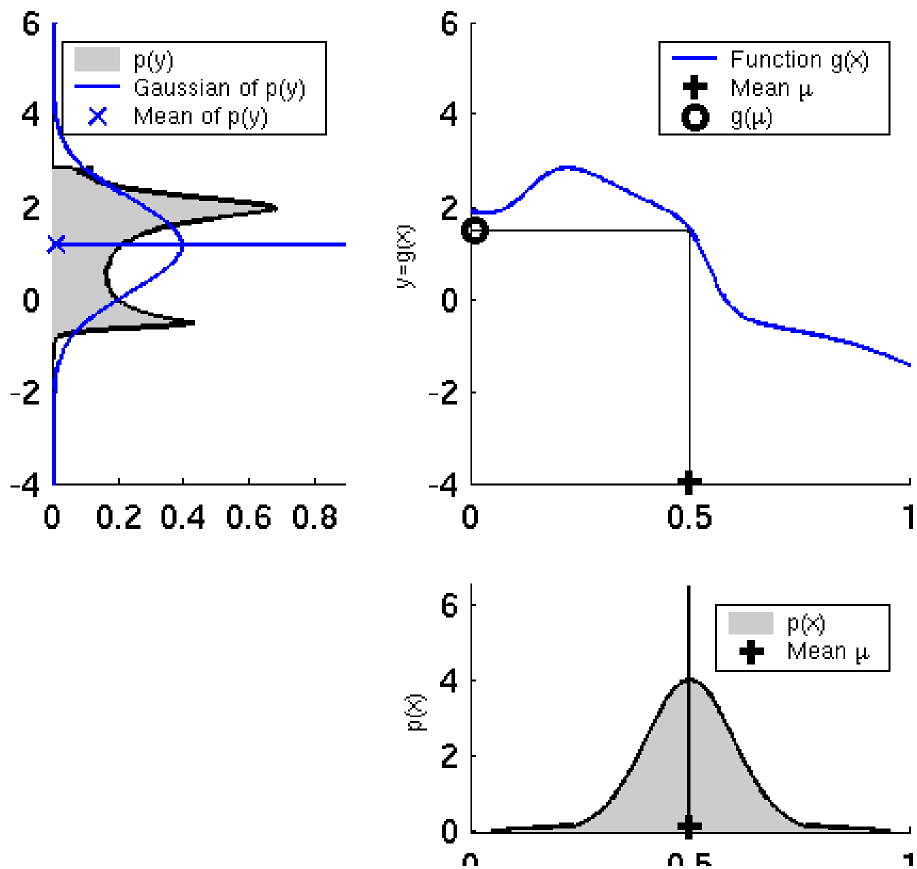
$$\boxed{x_t = g(u_t, x_{t-1})}$$

$$\boxed{z_t = h(x_t)}$$

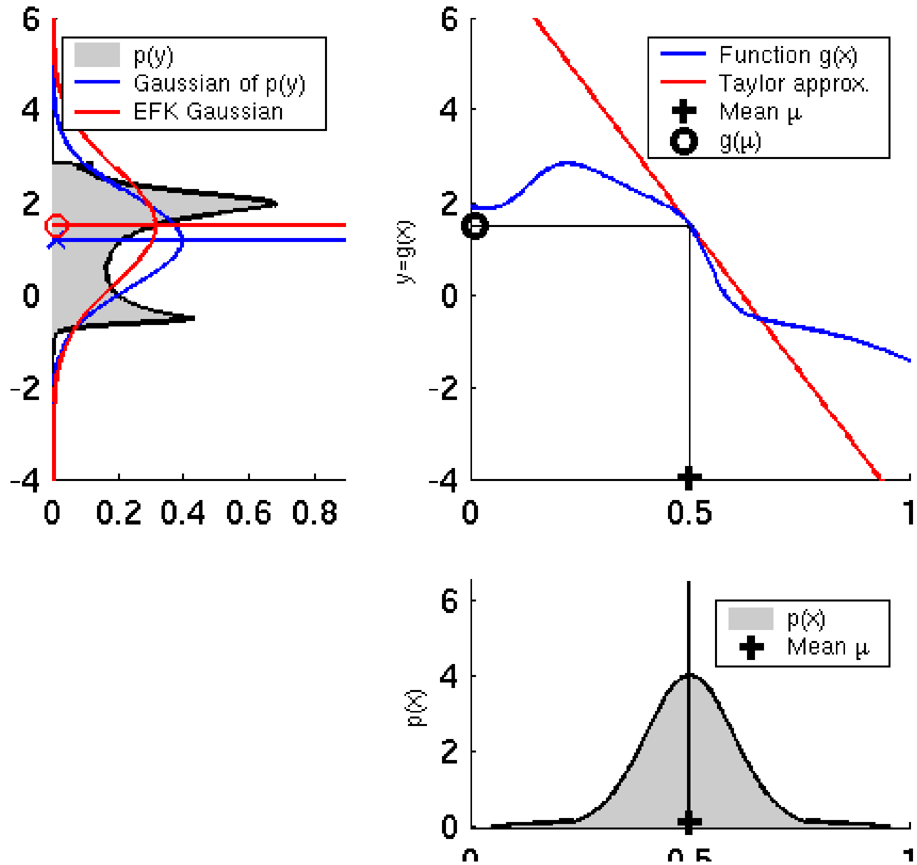
Linearity Assumption Revisited



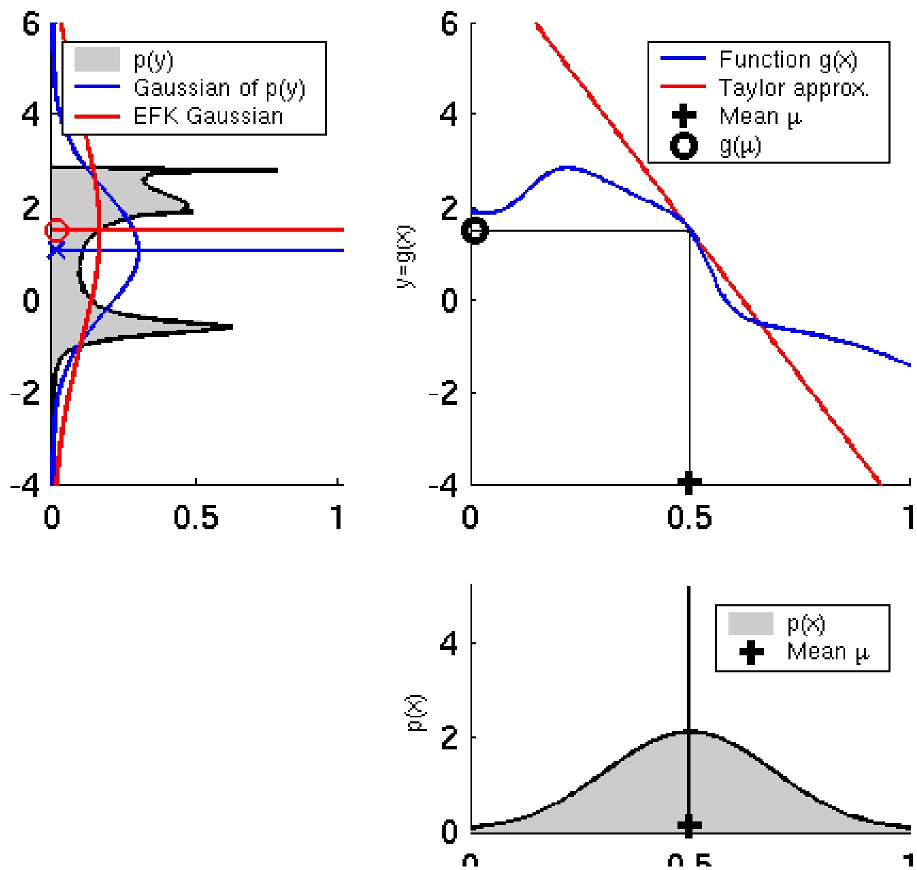
Non-linear Function



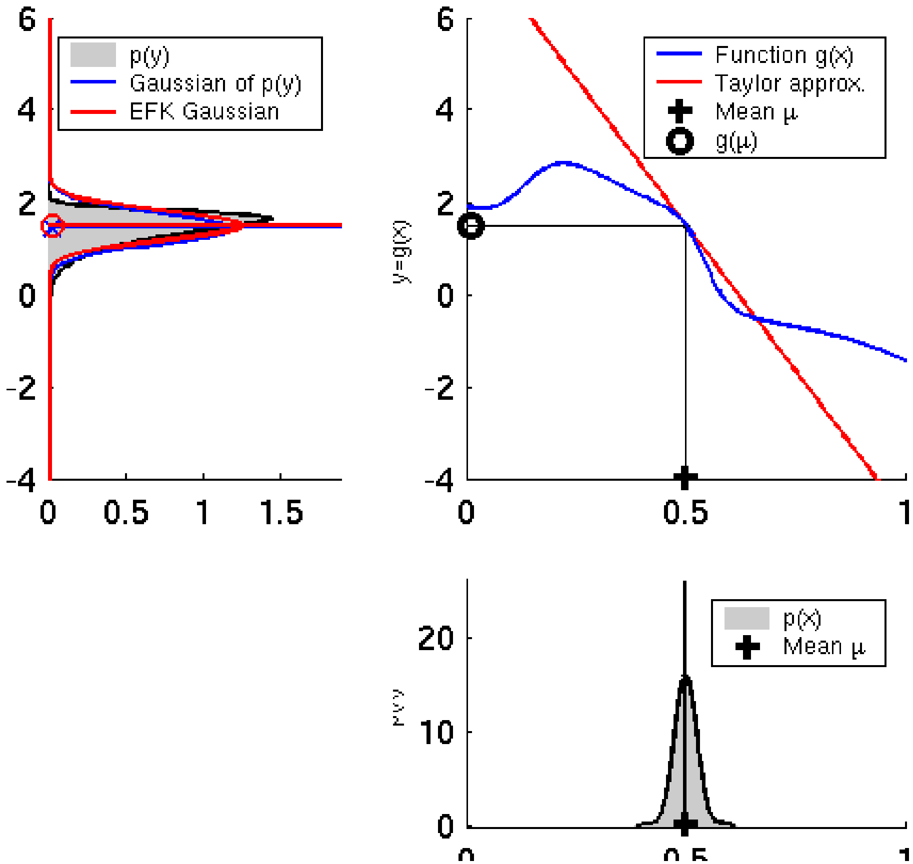
EKF Linearization (1)



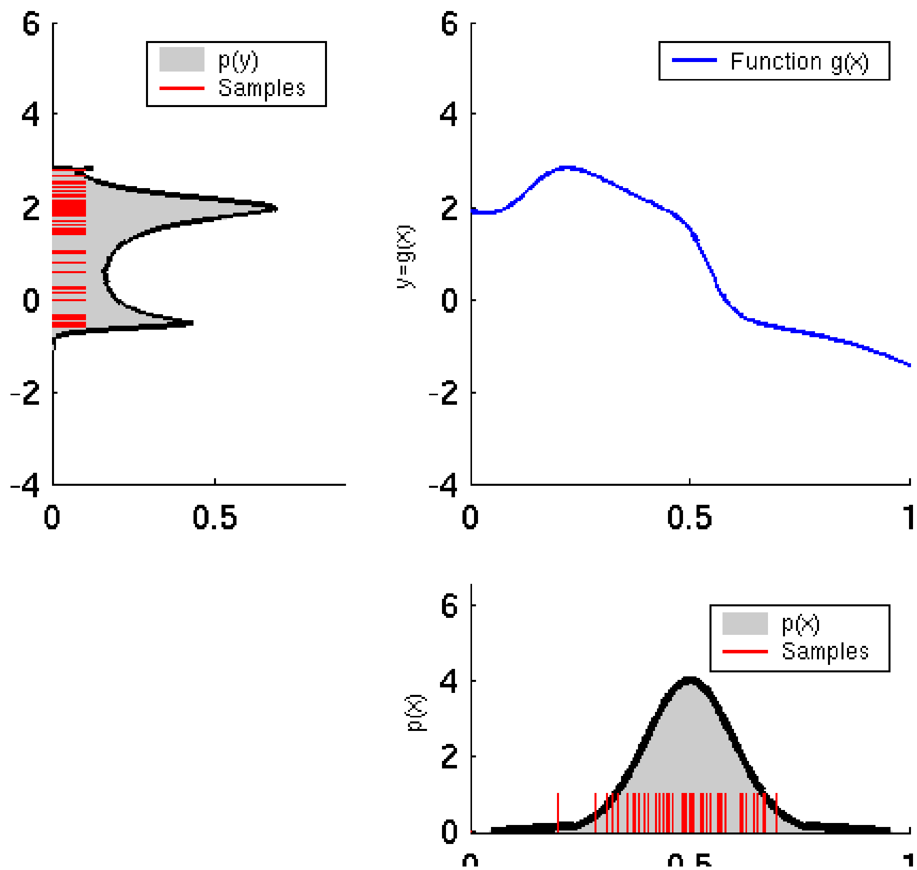
EKF Linearization (2)



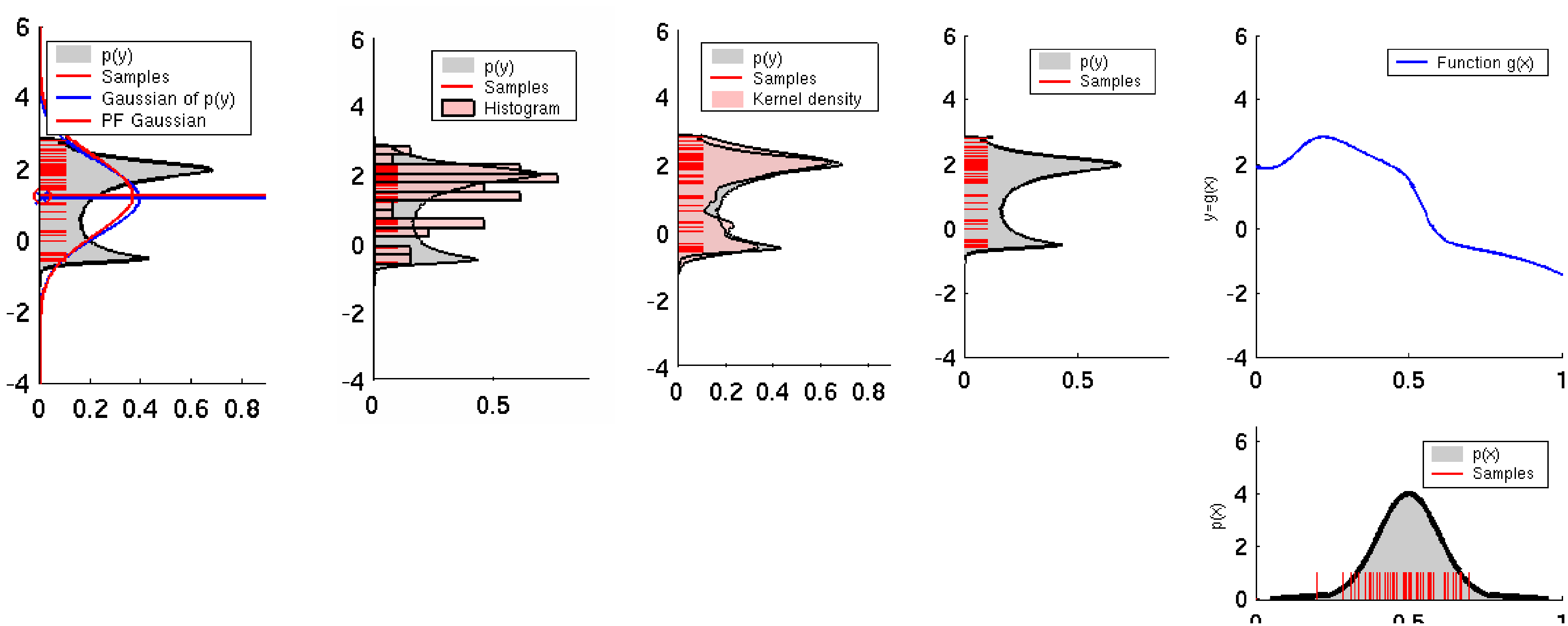
EKF Linearization (3)



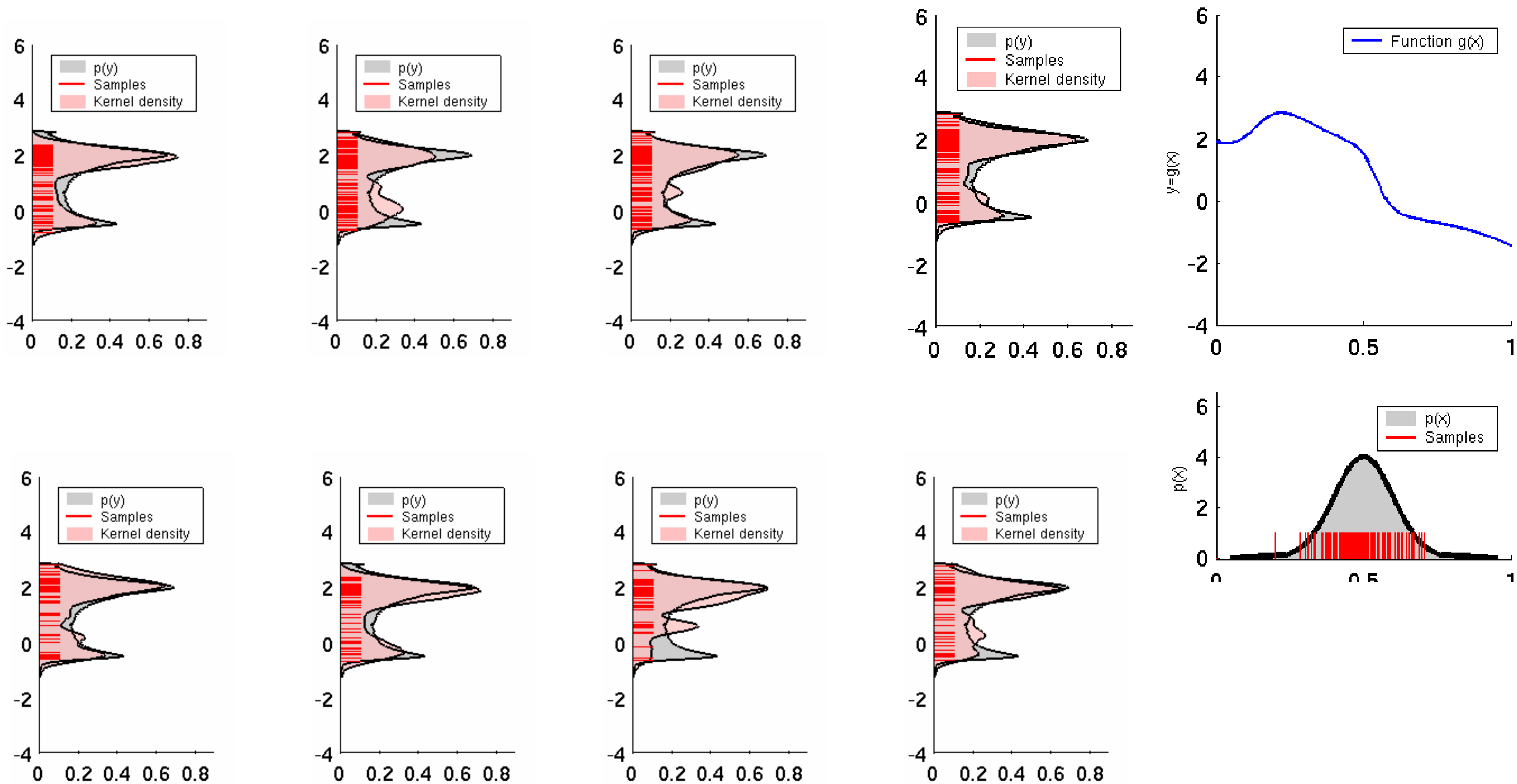
Particle Filter Projection



Density Extraction



Sampling Variance



EKF Algorithm

1. **Extended_Kalman_filter**(μ_{t-1}, Σ_{t-1}, u_t, z_t):

2. Prediction:

3. $\bar{\mu}_t = g(u_t, \mu_{t-1})$ $\longleftarrow$ $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$

4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$ $\longleftarrow$ $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction:

6. $K_t = \bar{\Sigma}_t H_t (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$ $\longleftarrow$ $K_t = \bar{\Sigma}_t C_t (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$

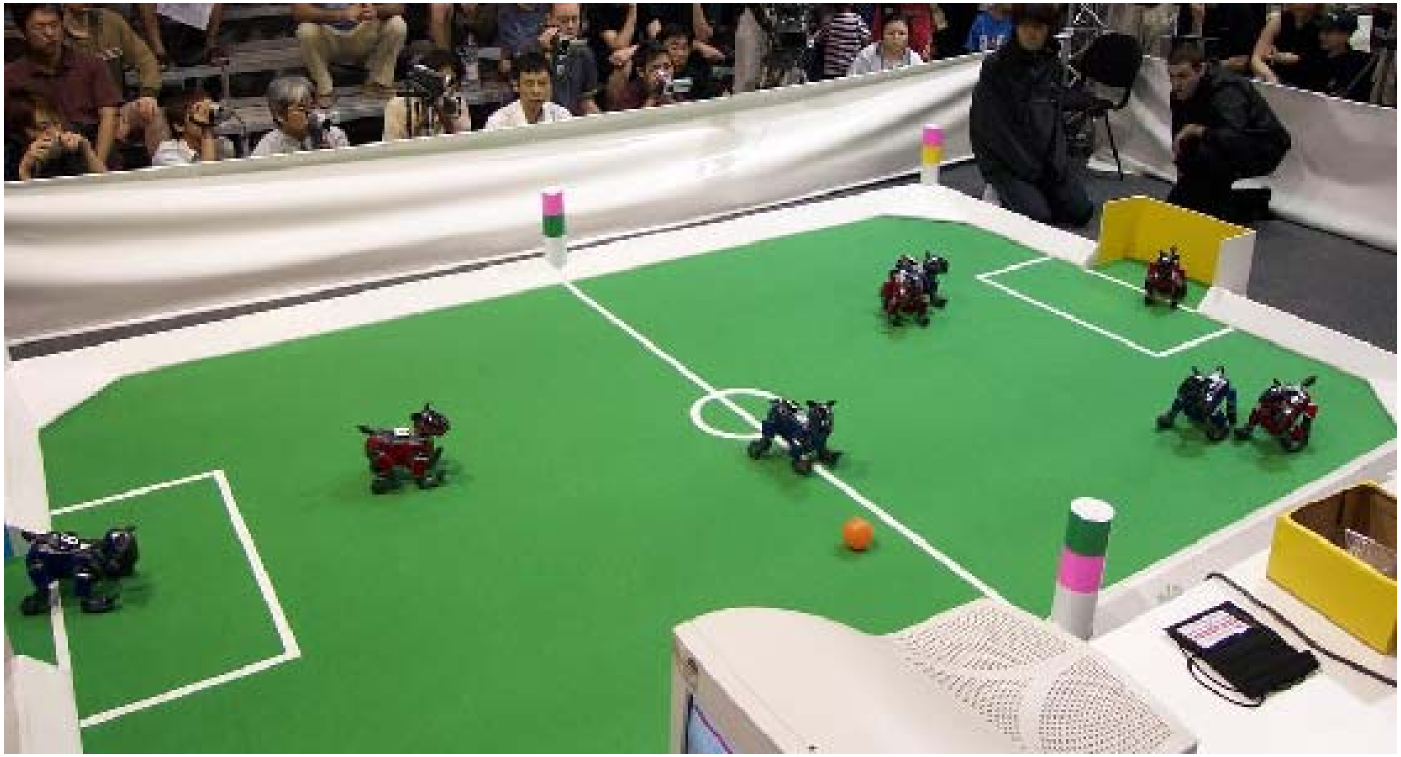
7. $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$ $\longleftarrow$ $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$

8. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$ $\longleftarrow$ $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

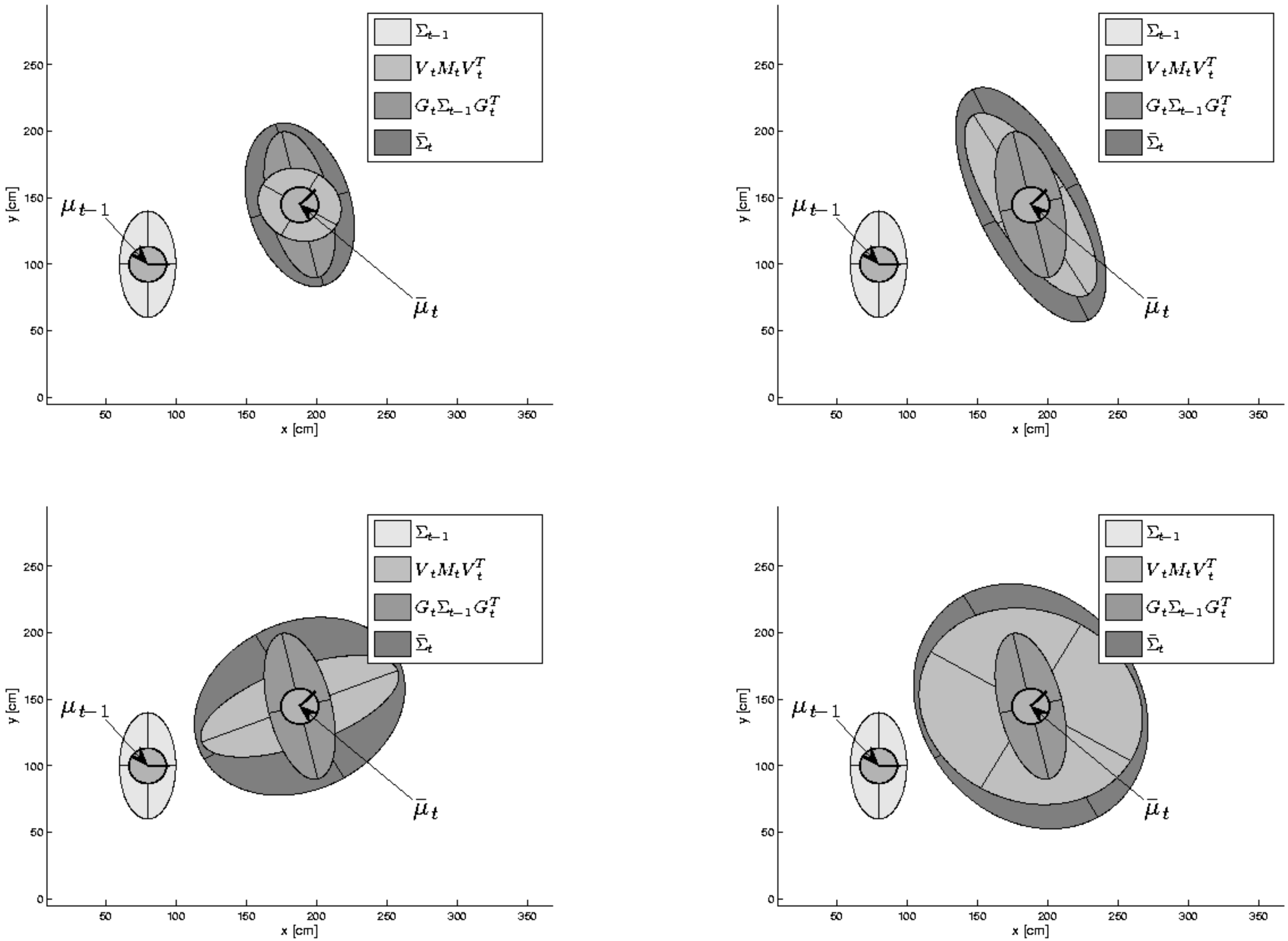
9. Return μ_t, Σ_t

$$H_t = \frac{\partial h(\bar{\mu}_t)}{x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{x_{t-1}}$$

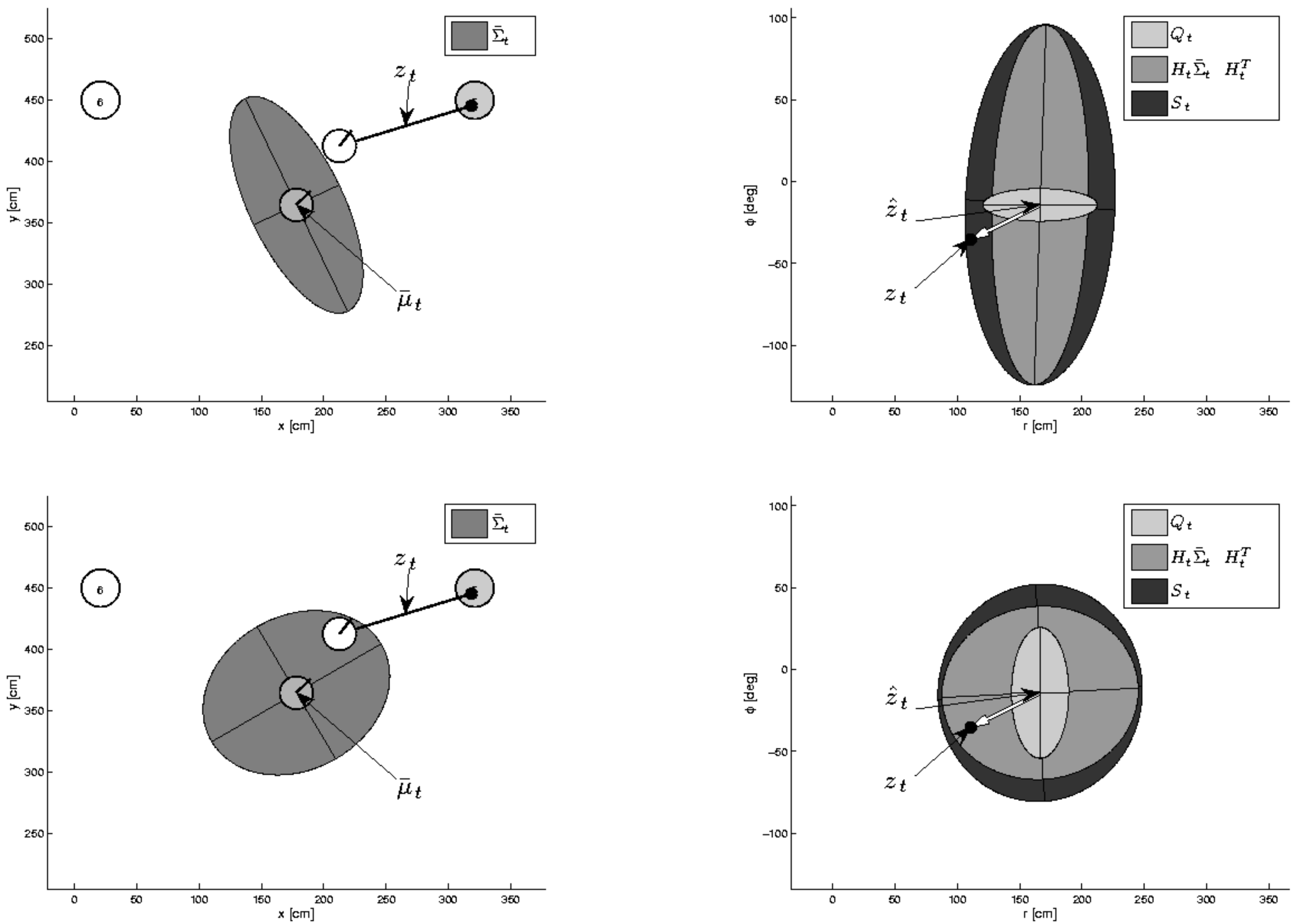
Landmark-based Localization



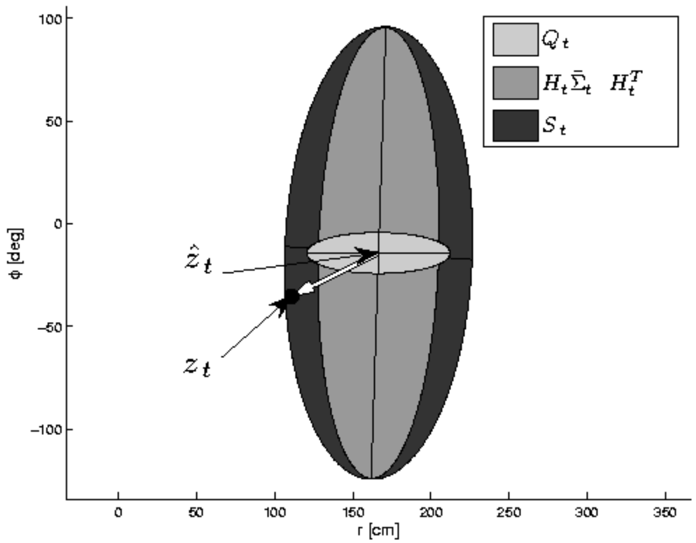
EKF Prediction Step

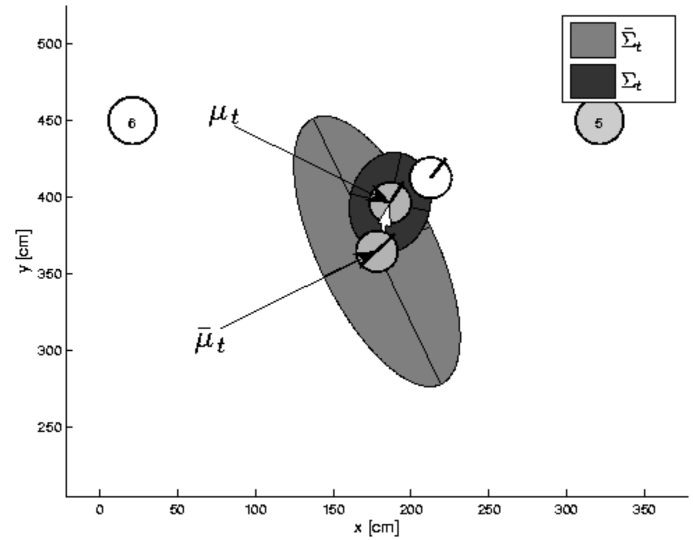


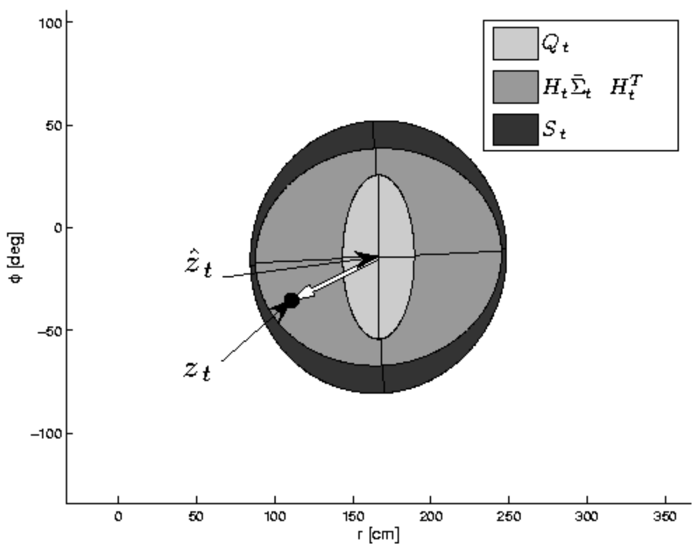
EKF Observation Prediction Step

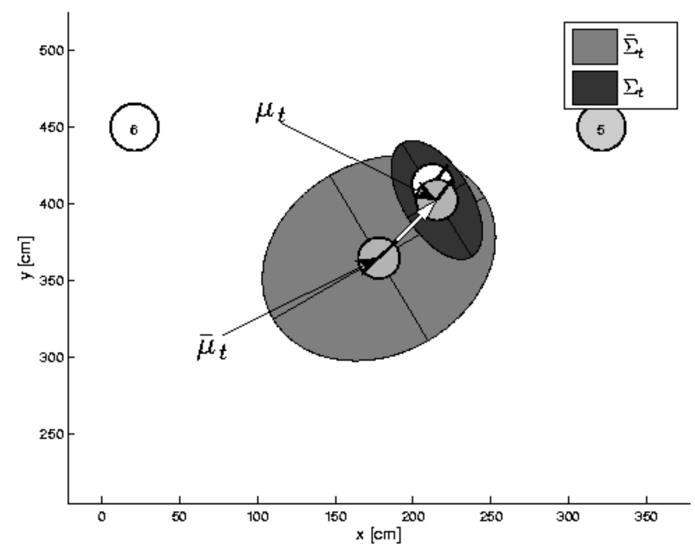


EKF Correction Step

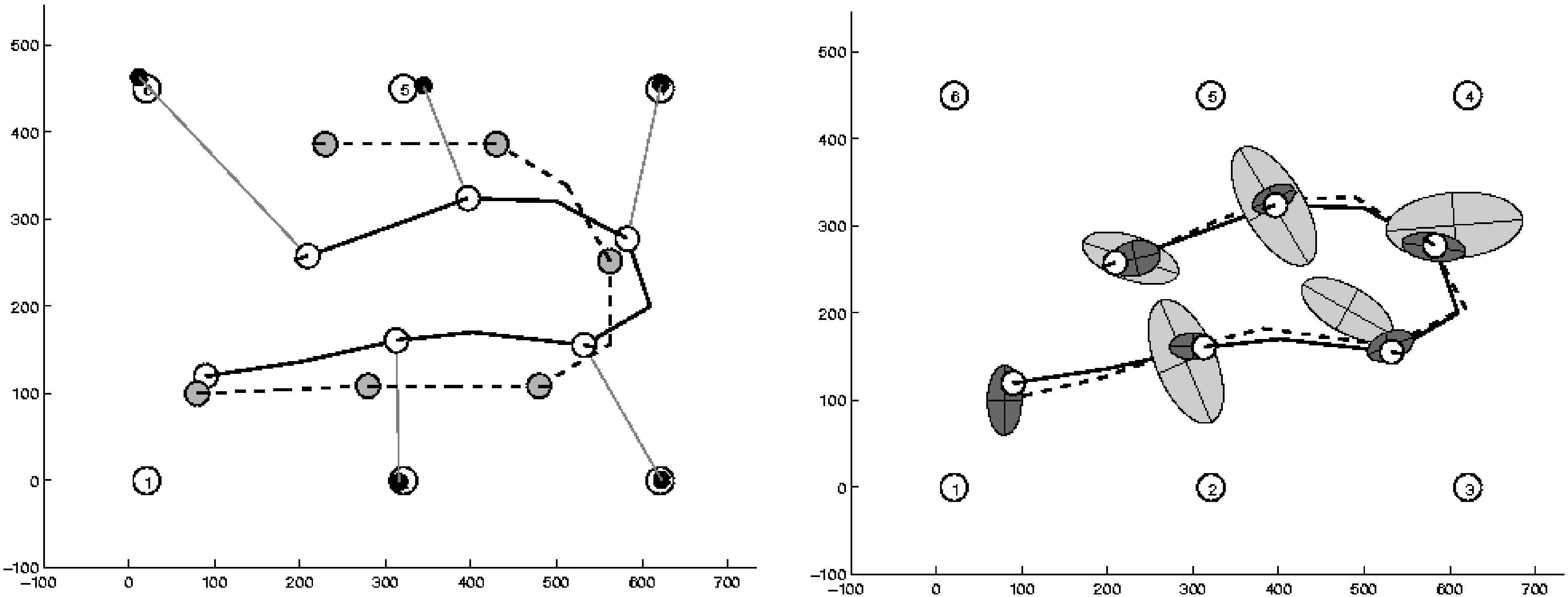




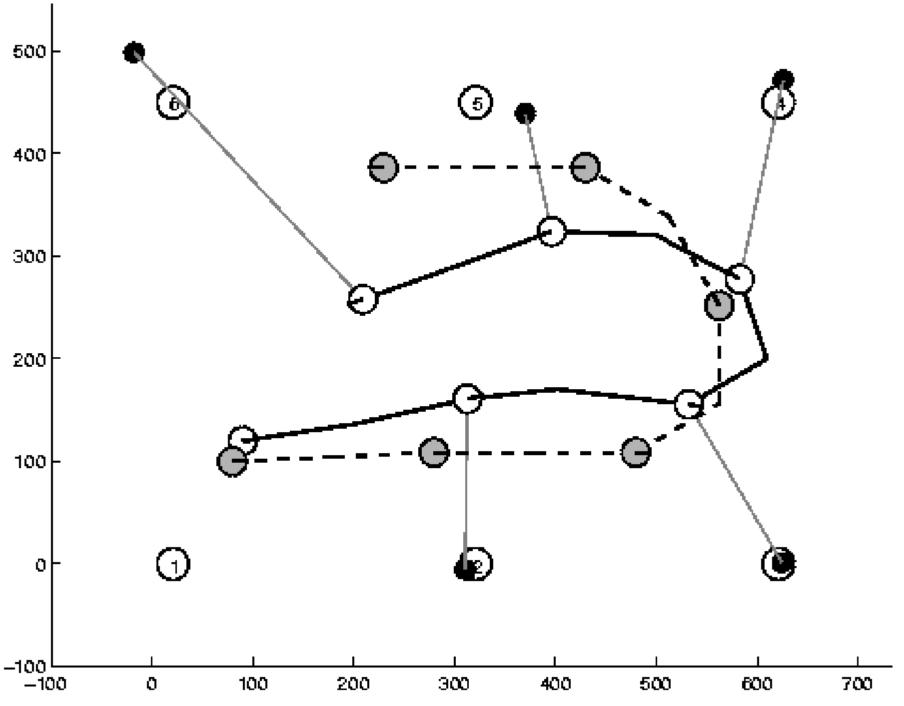


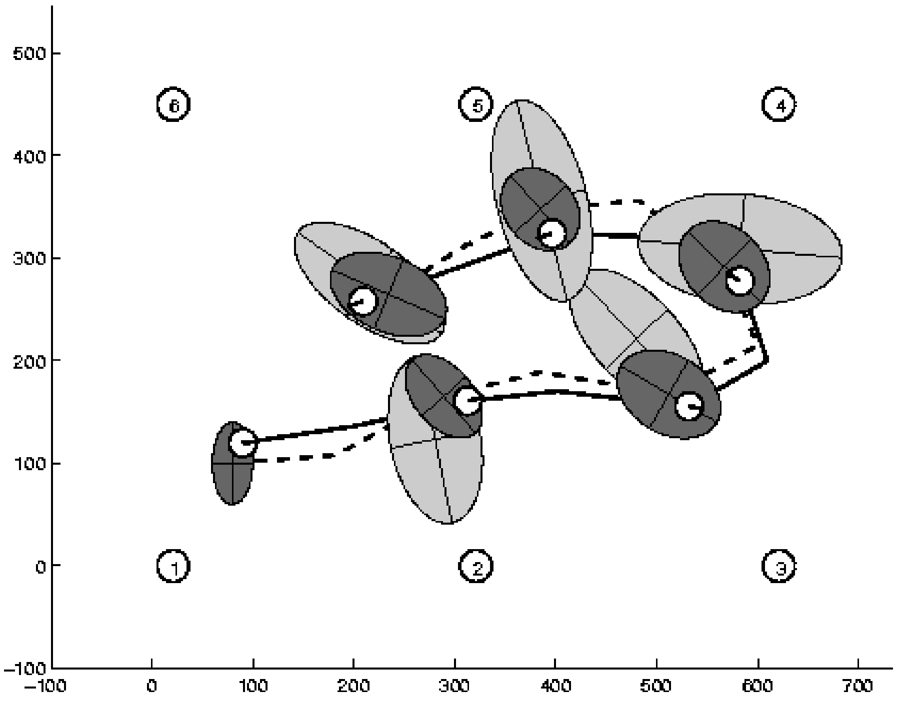


Estimation Sequence (1)

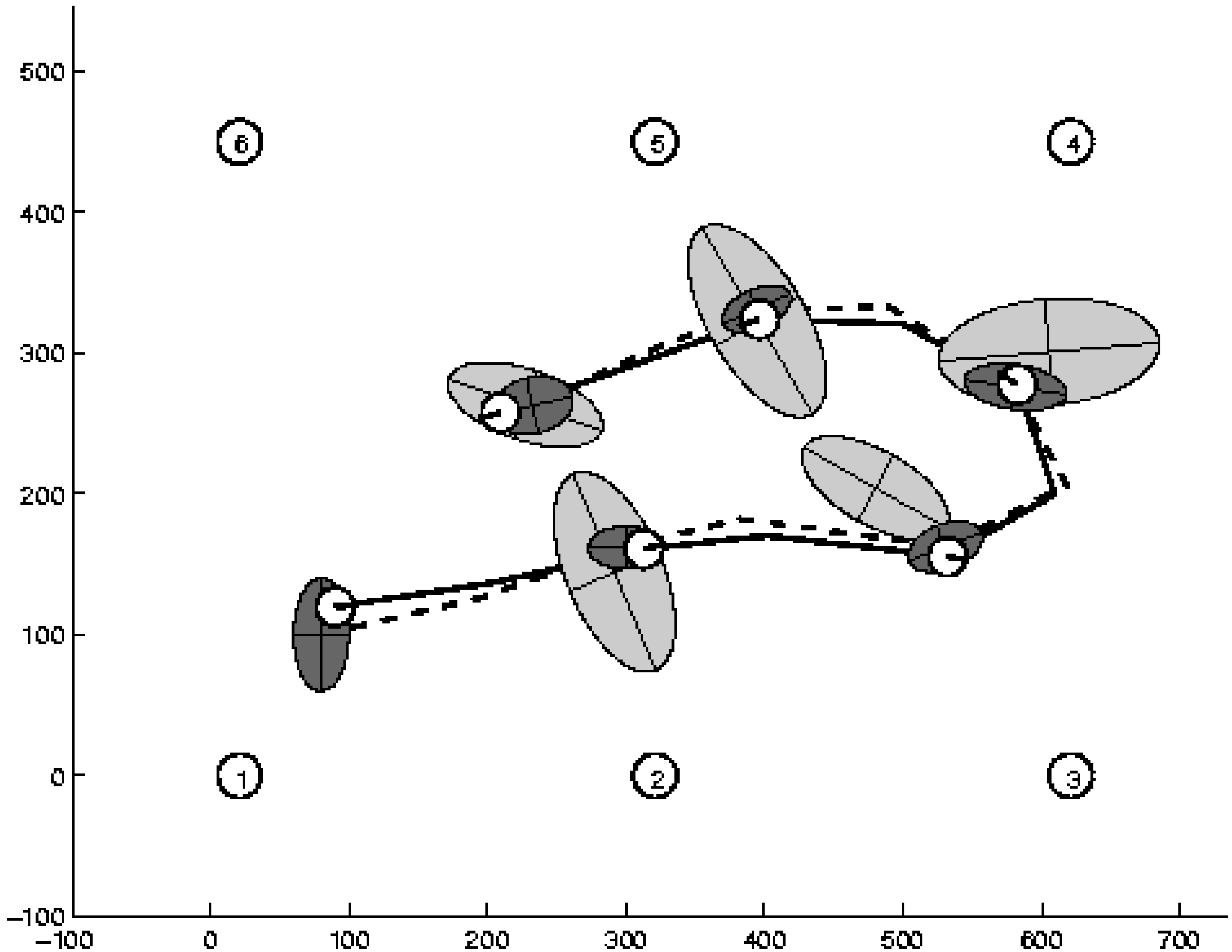


Estimation Sequence (2)





Comparison to GroundTruth

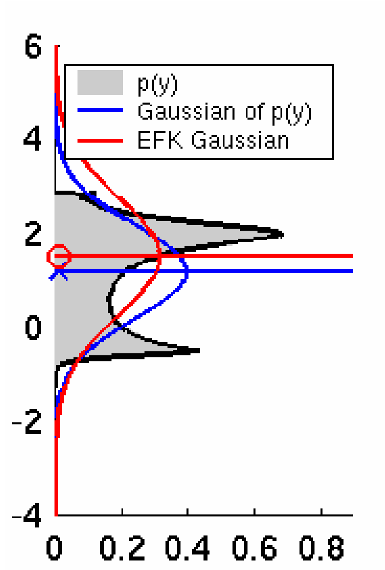


EKF Summary

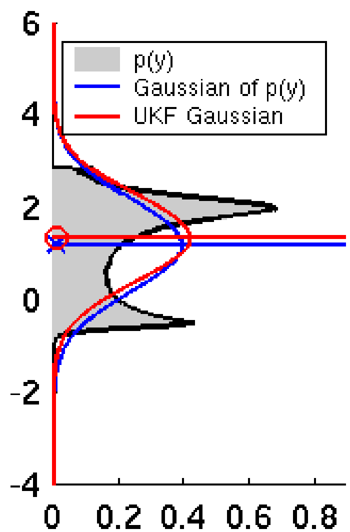
- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n:
$$O(k^{2.376} + n^2)$$

- Not optimal!
- Can diverge if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!

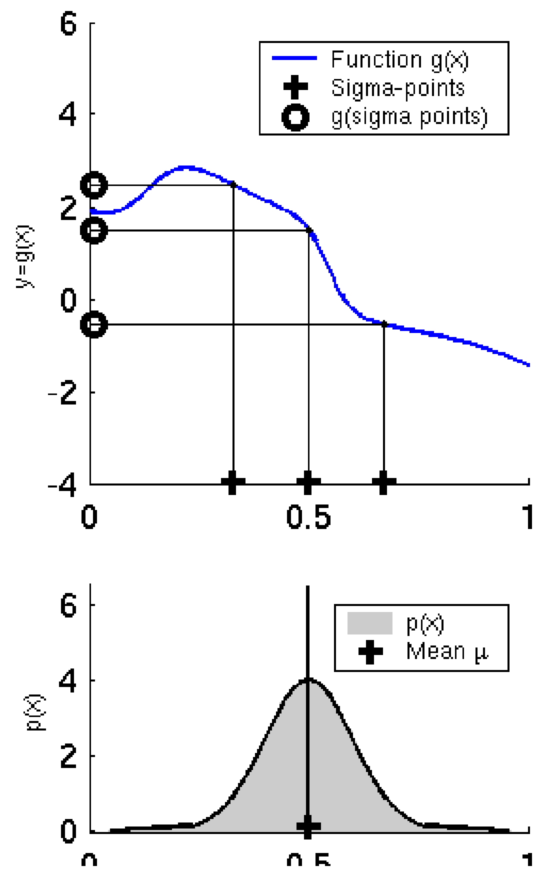
Linearization via Unscented Transform



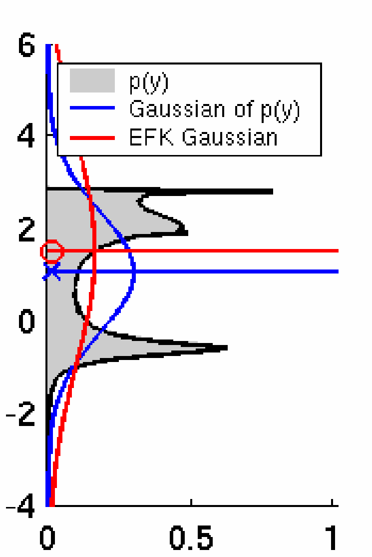
EKF



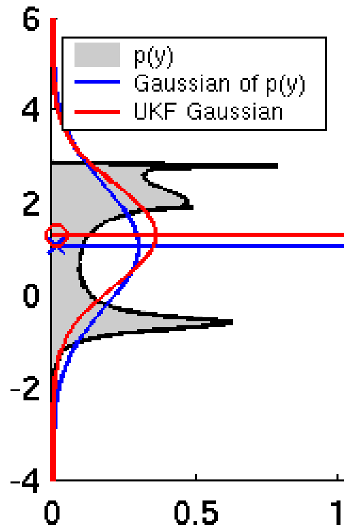
UKF



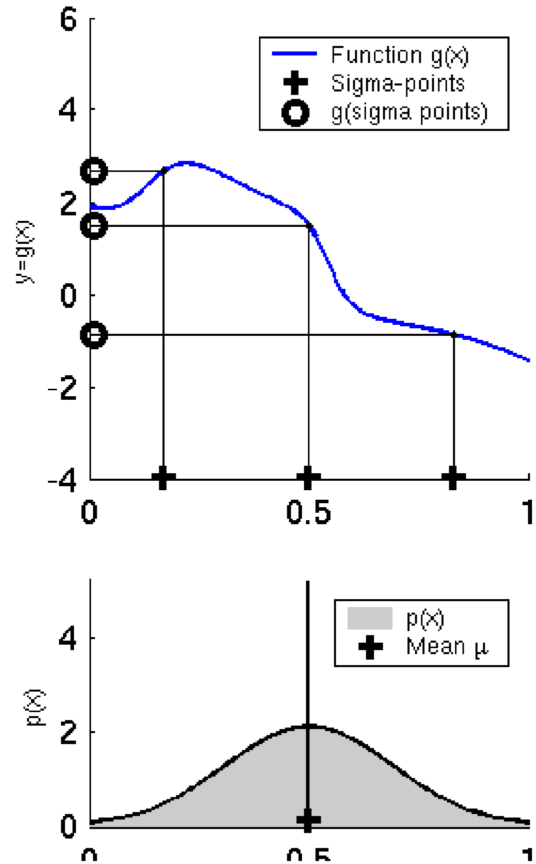
UKF Sigma-Point Estimate (2)



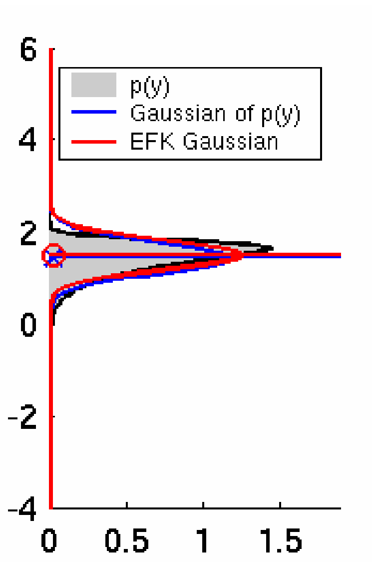
EKF

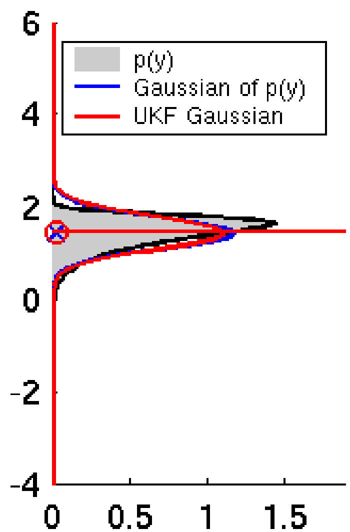


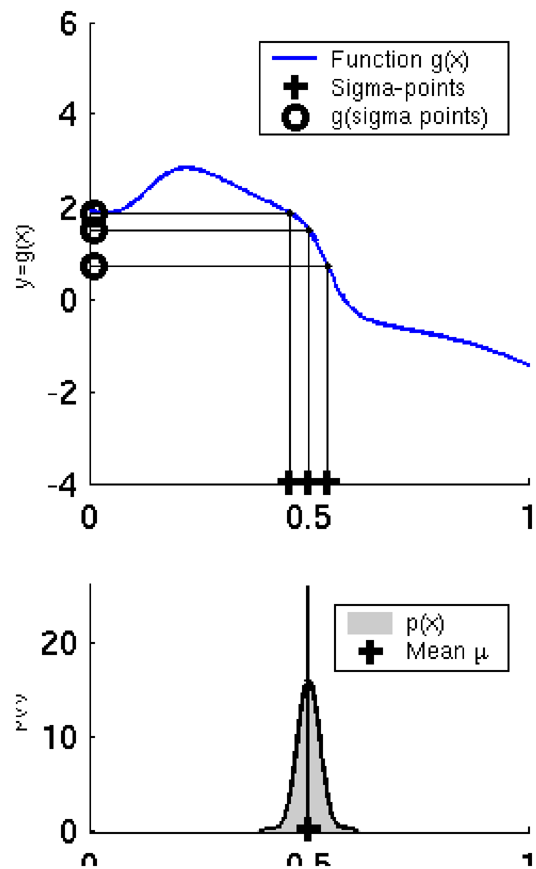
UKF



UKF Sigma-Point Estimate (3)







EKF

UKF

Unscented Transform

Sigma points

$$\chi^0 = \mu$$

$$\chi^i = \mu \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i$$

Weights

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta)$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\ldots,2n$$

Pass sigma points through nonlinear function

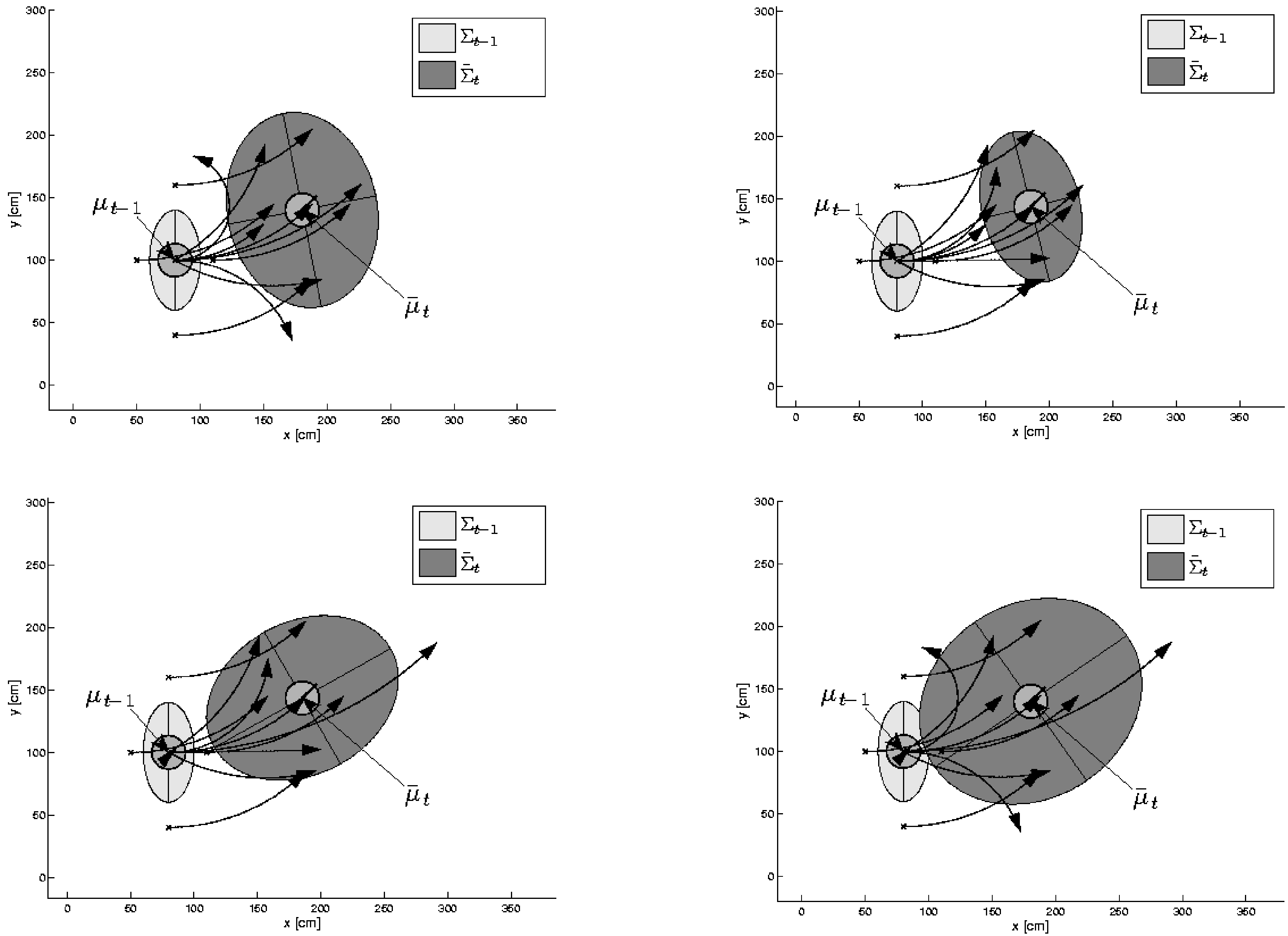
$$\psi^i = g(\chi^i)$$

Recover mean and covariance

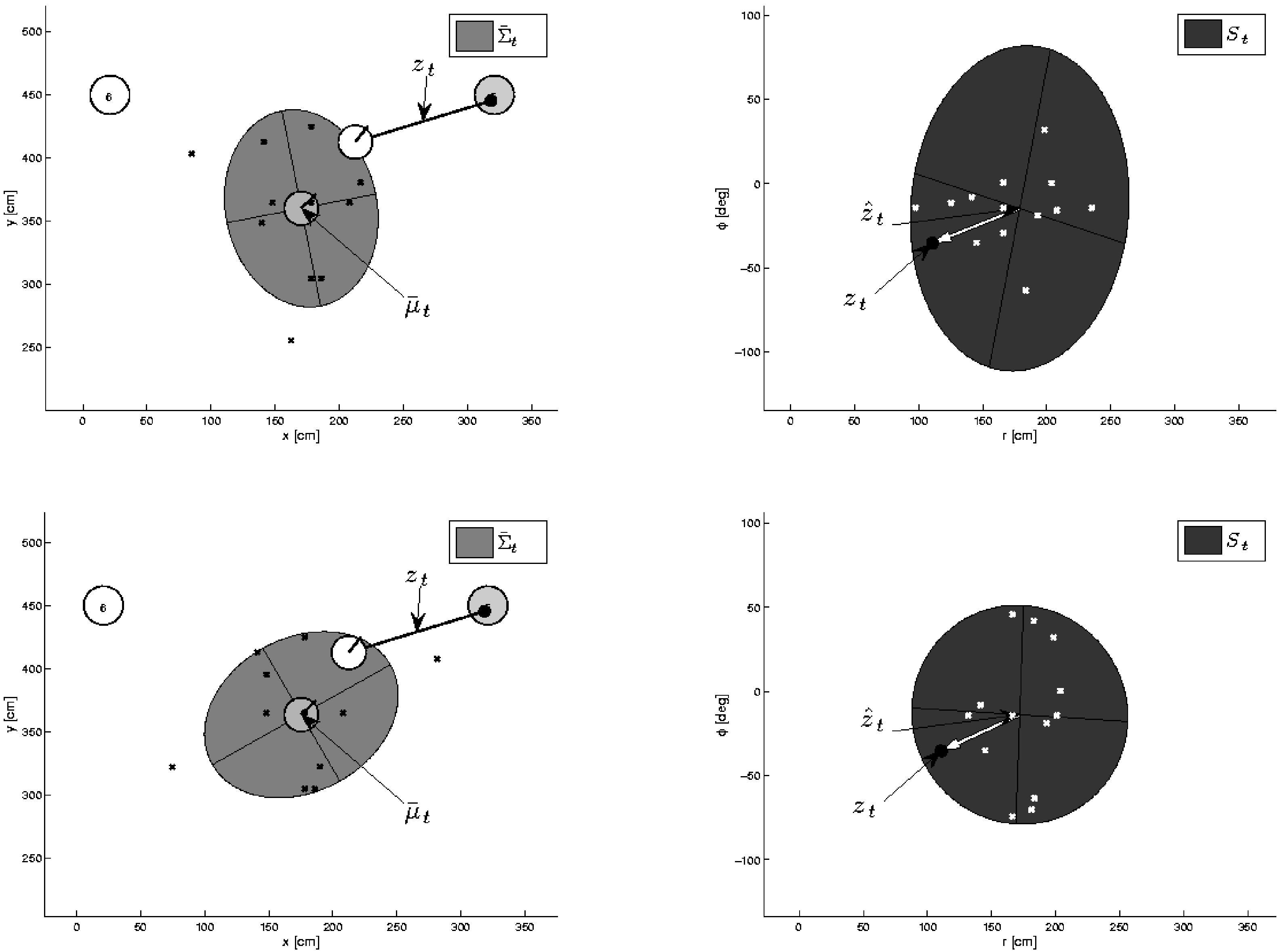
$$\mu' = \sum_{i=0}^{2n} w_m^i \psi^i$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^i (\psi^i - \mu)(\psi^i - \mu)^T$$

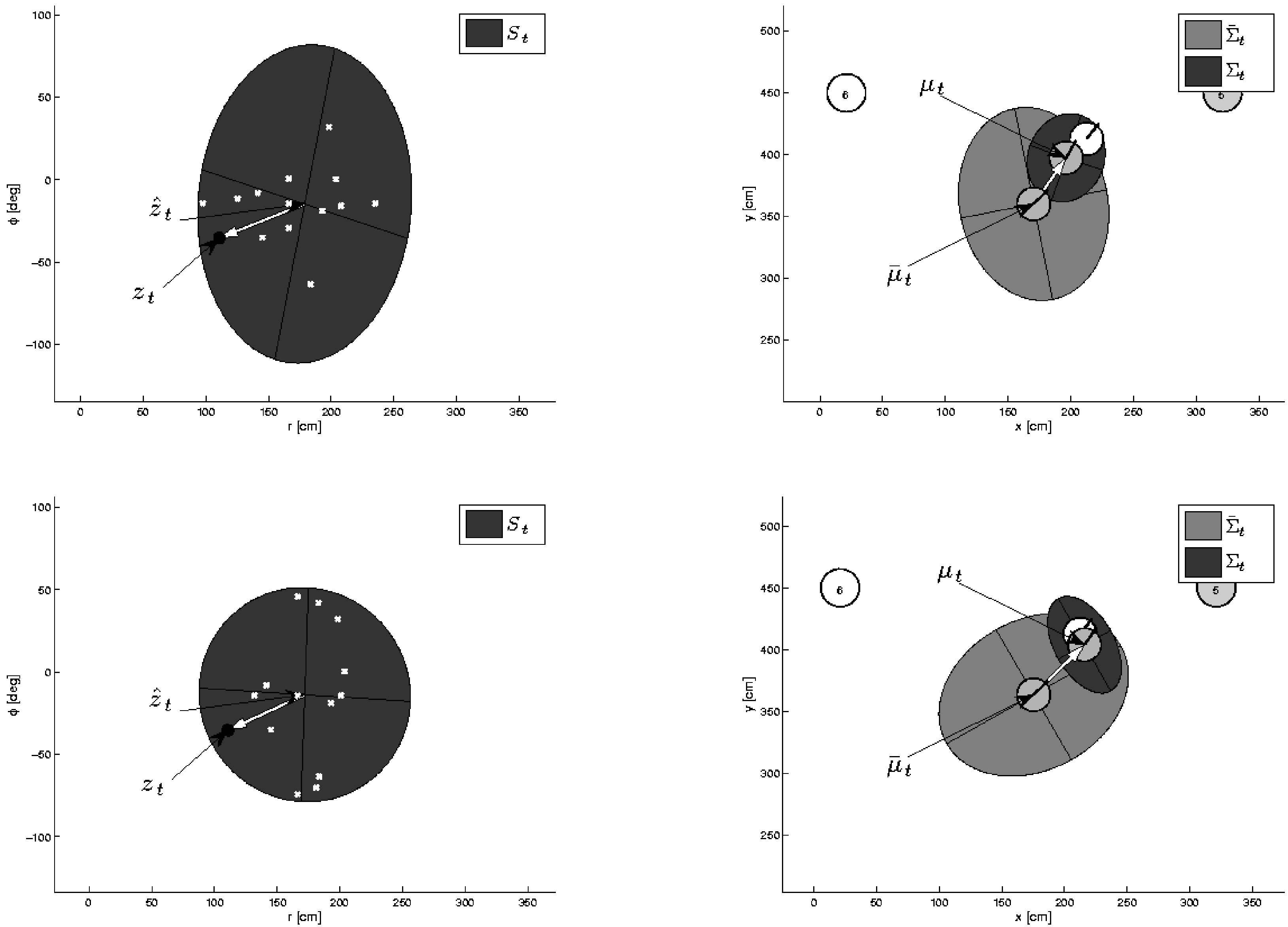
UKF Prediction Step



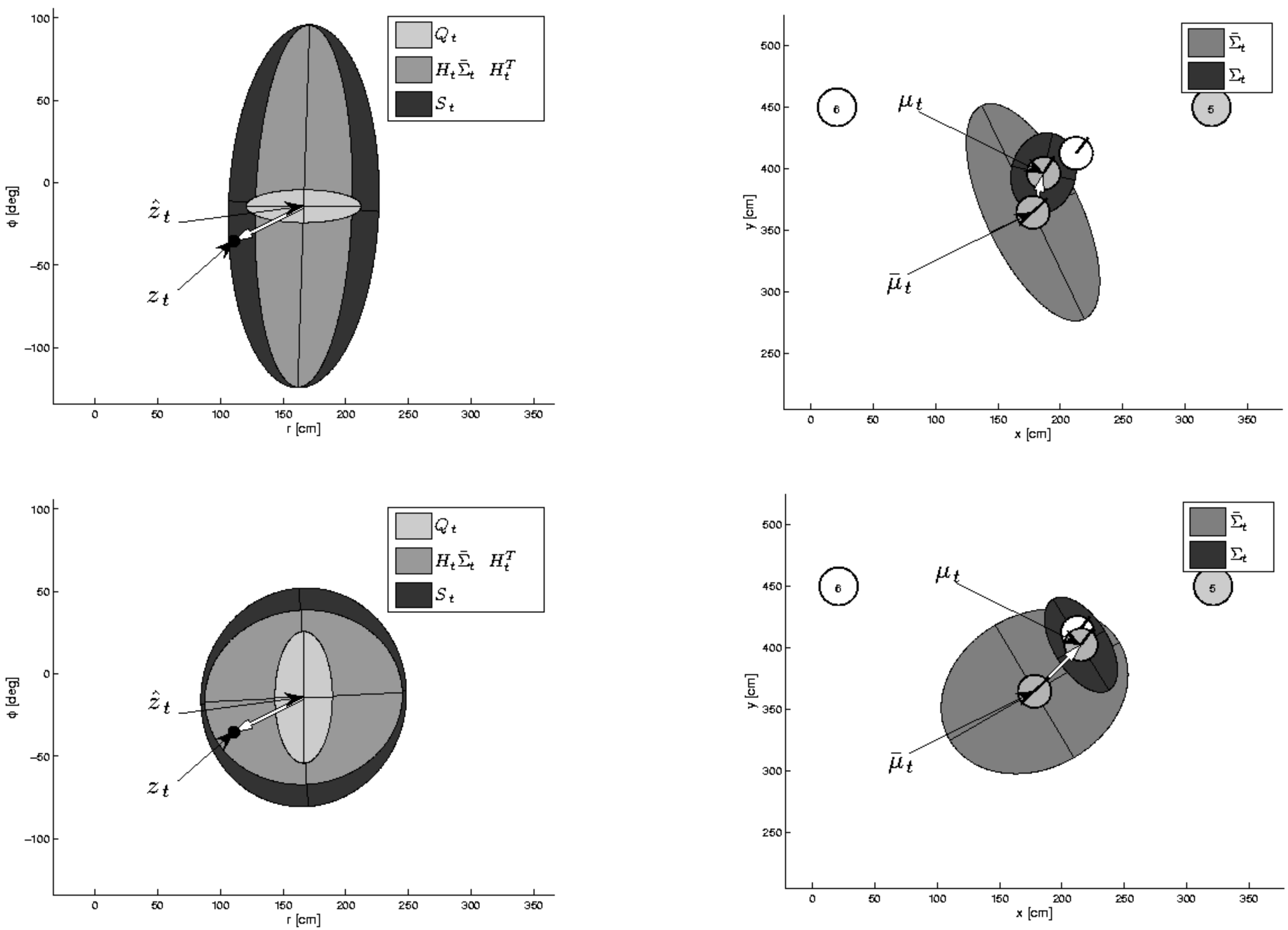
UKF Observation Prediction Step



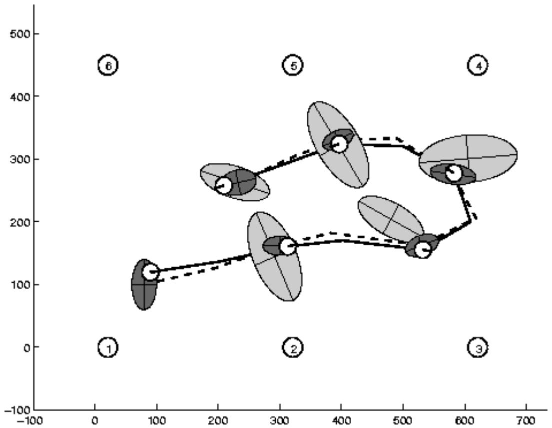
UKF Correction Step



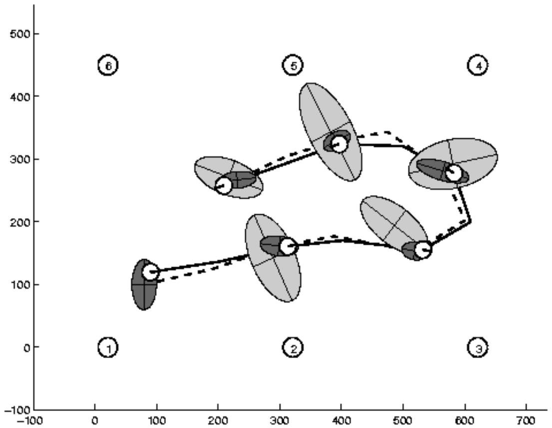
EKF Correction Step



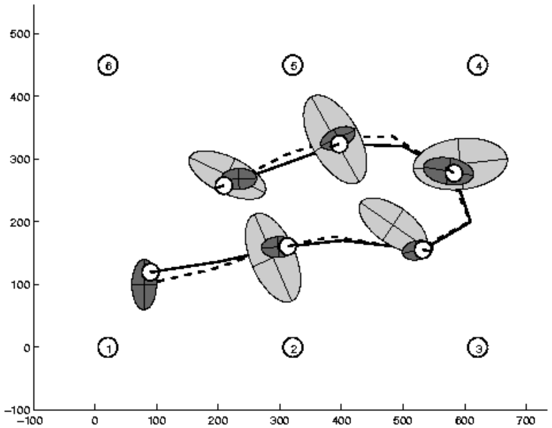
Estimation Sequence



EKF

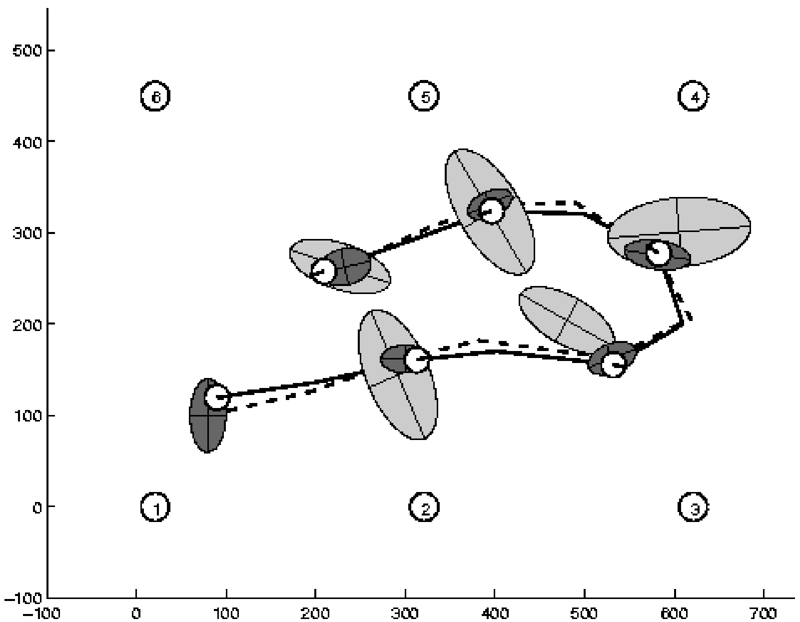


PF

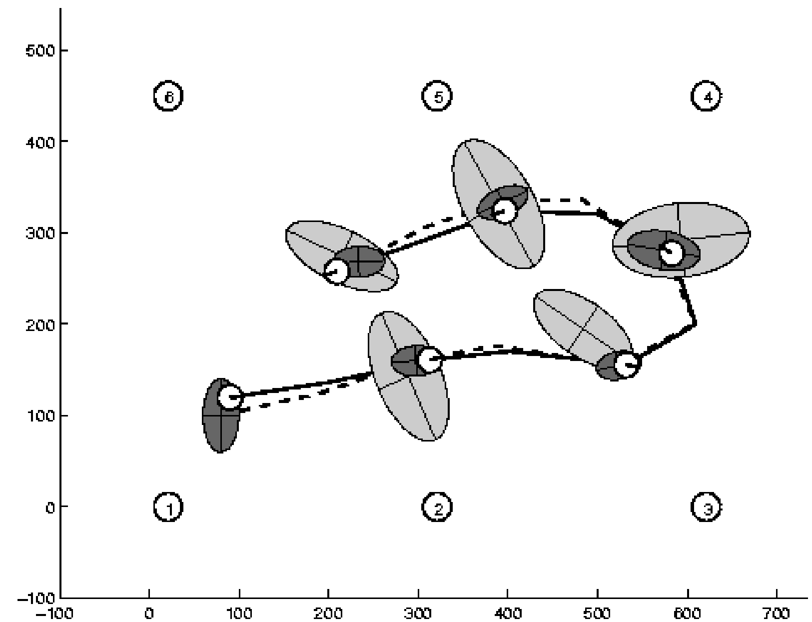


UKF

Estimation Sequence

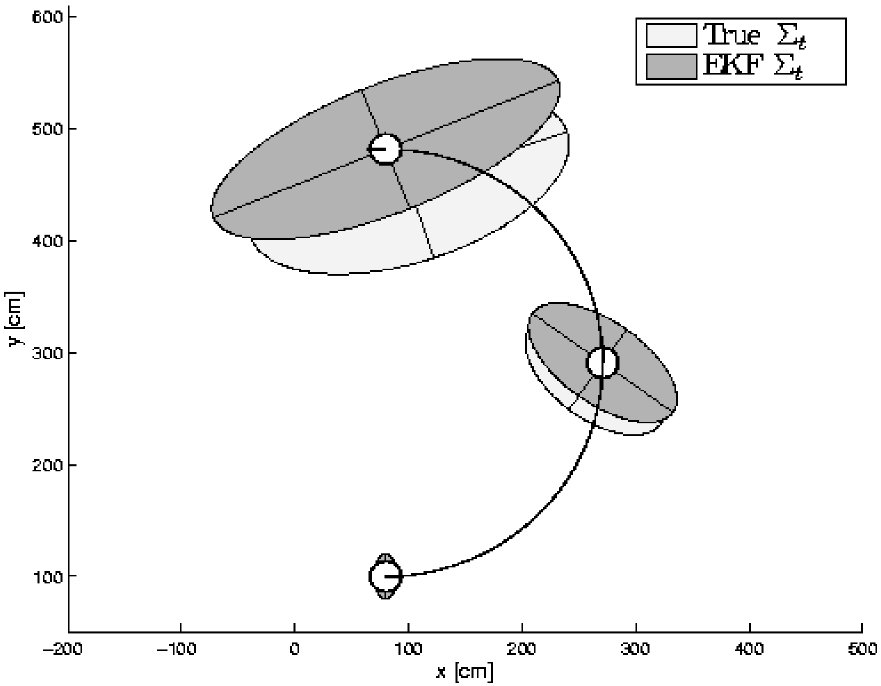


EKF

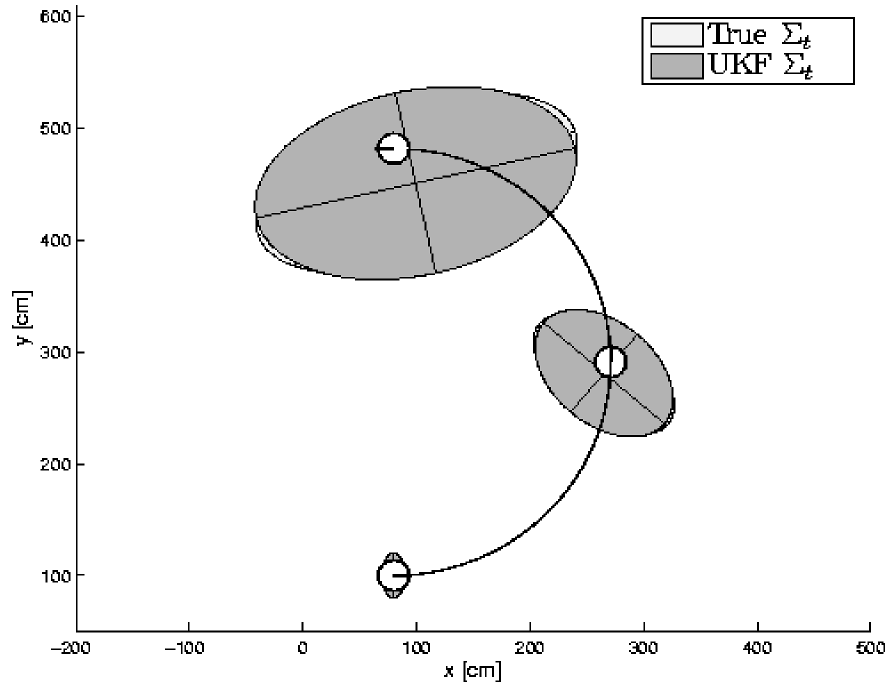


UKF

Prediction Quality



EKF



UKF

UKF Summary

- Highly efficient: Same complexity as EKF, with a constant factor slower in typical practical applications
- Better linearization than EKF: Accurate in first two terms of Taylor expansion (EKF only first term)
- Derivative-free: No Jacobians needed
- Still not optimal!

SLAM: Simultaneous Localization and Mapping

Mapping with Raw Odometry

SLAM:

Simultaneous Localization and Mapping

- Full SLAM:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

- Online SLAM:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int\int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \, dx_1, dx_2, \ldots, dx_{t-1}$$

Integrations typically done one at a time

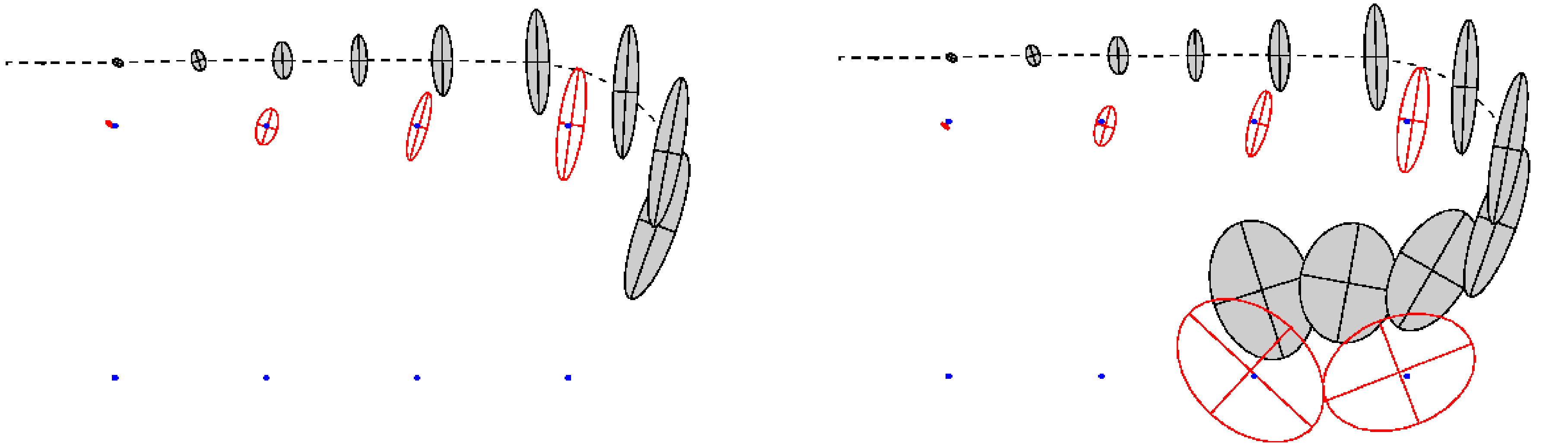
SLAM: Mapping with Kalman Filters

- Map with N landmarks:(2N+3)-dimensional Gaussian

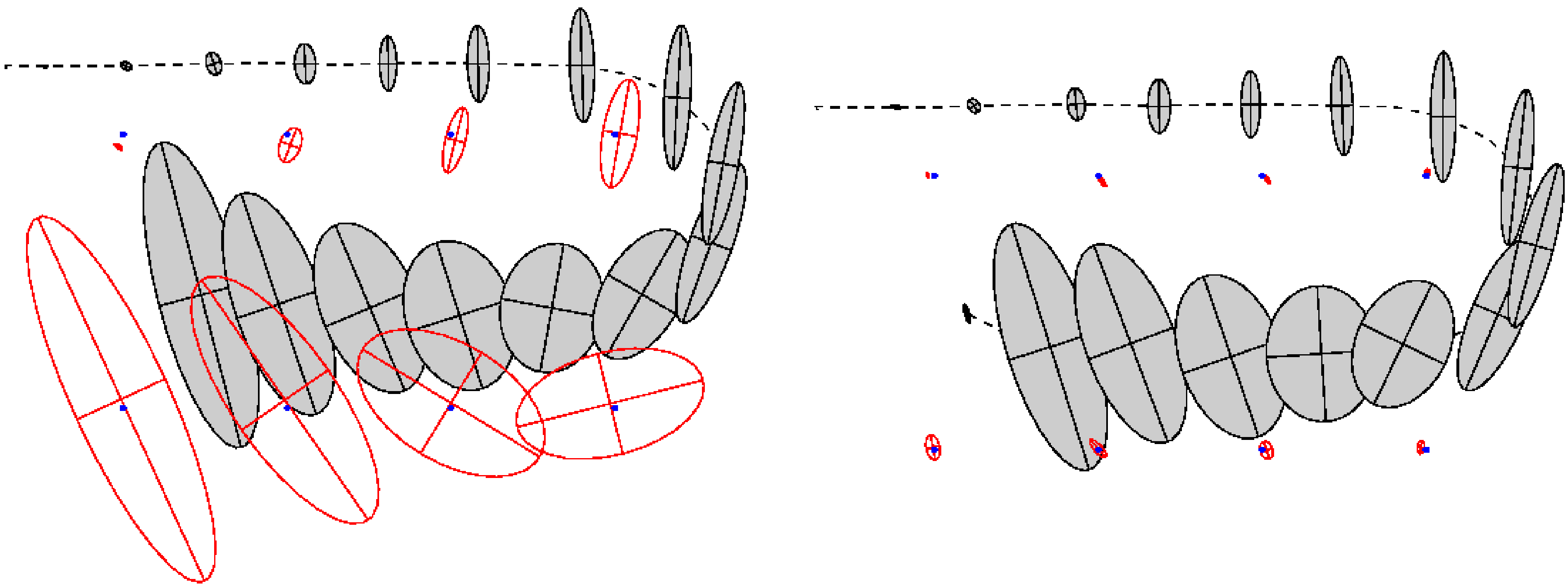
$$Bel(x_t, m_t) = \left\langle \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_N \\ x \\ y \\ \theta \end{pmatrix}, \begin{pmatrix} \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} & \sigma_{l_1 x} & \sigma_{l_1 y} & \sigma_{l_1 \theta} \\ \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} & \sigma_{l_2 x} & \sigma_{l_2 y} & \sigma_{l_2 \theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 & \sigma_{l_N x} & \sigma_{l_N y} & \sigma_{l_N \theta} \\ \sigma_{l_1 x} & \sigma_{l_2 x} & \cdots & \sigma_{l_N x} & \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{l_1 y} & \sigma_{l_2 y} & \cdots & \sigma_{l_N y} & \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{l_1 \theta} & \sigma_{l_2 \theta} & \cdots & \sigma_{l_N \theta} & \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} \right\rangle$$

- Can handle hundreds of dimensions

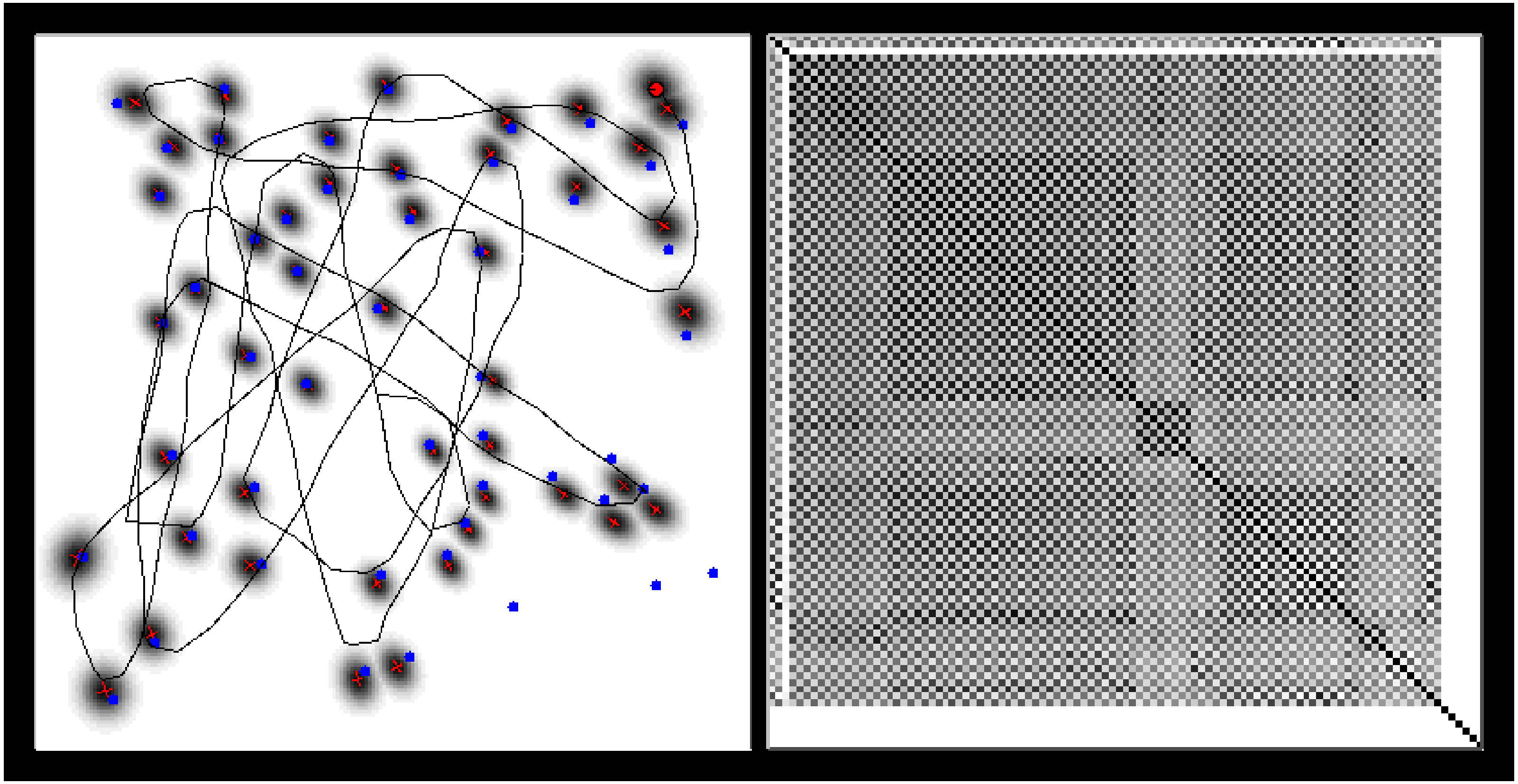
SLAM: Mapping with Kalman Filters



SLAM: Mapping with Kalman Filters



SLAM: Mapping with Kalman Filters

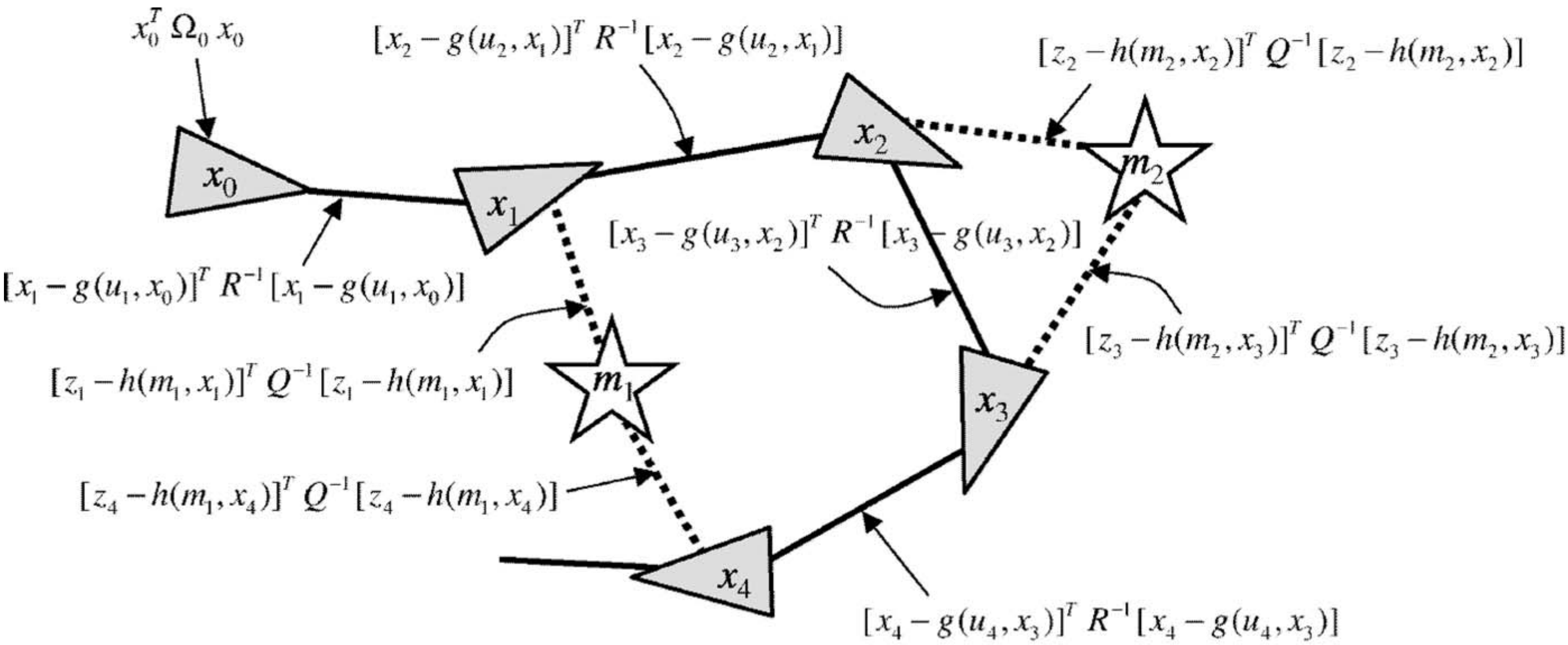


Map Correlation matrix

Graph-SLAM

- Full SLAM technique
- Generates probabilistic links
- Computes map only occasionally
- Based on Information Filter form

Graph-SLAM Idea



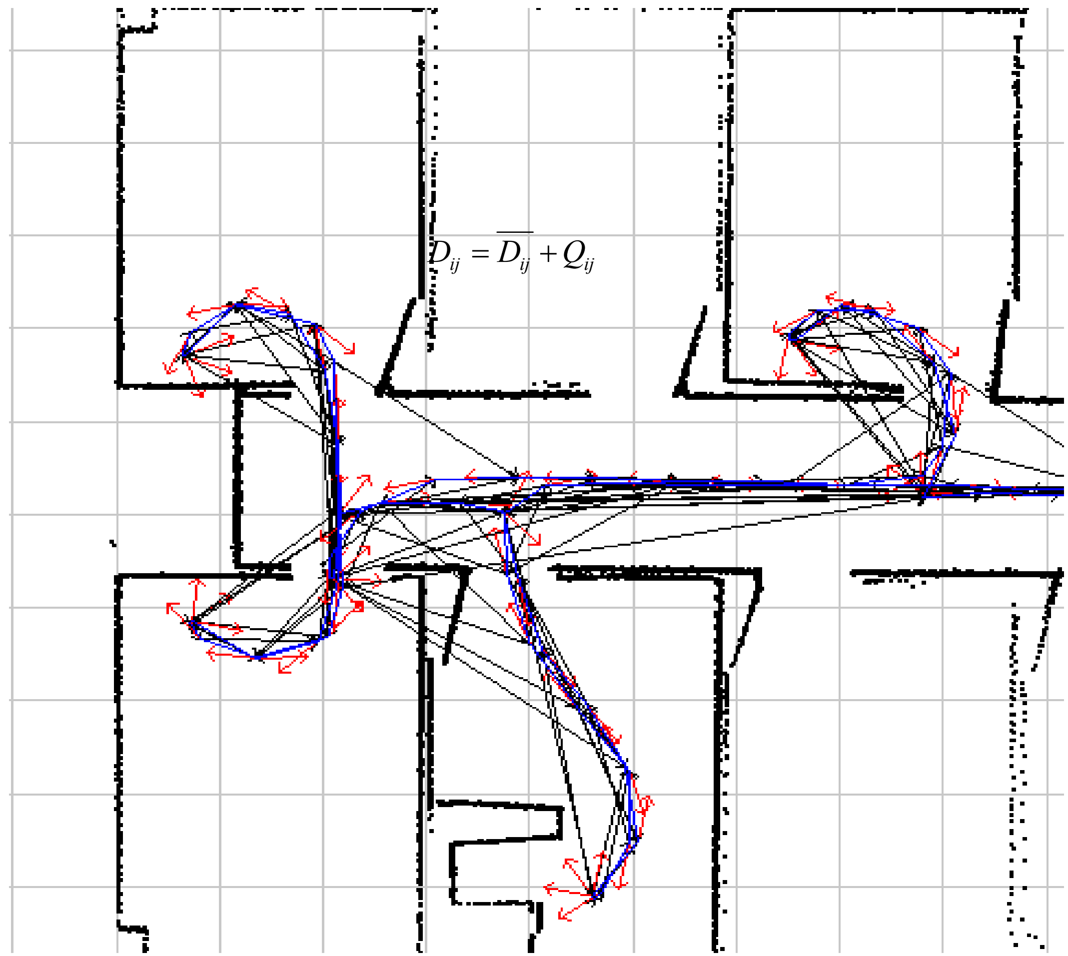
<u>Sum of all constraints:</u>

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

Robot Poses and Scans [Lu and Milios 1997]

- Successive robot poses connected by odometry
- Sensor readings yield constraints between poses
- Constraints represented by Gaussians

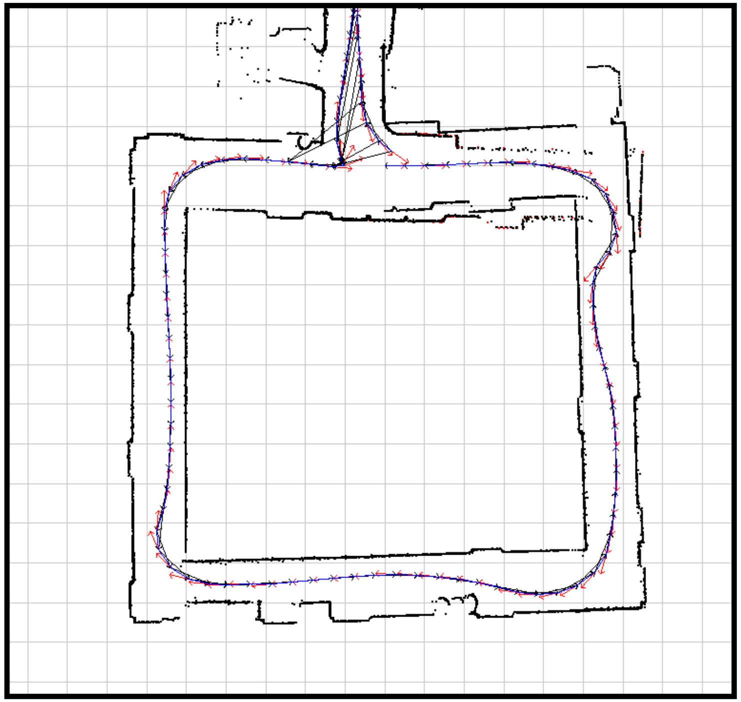
$$D_{ij} = \overline{D_{ij}} + Q_{ij}$$



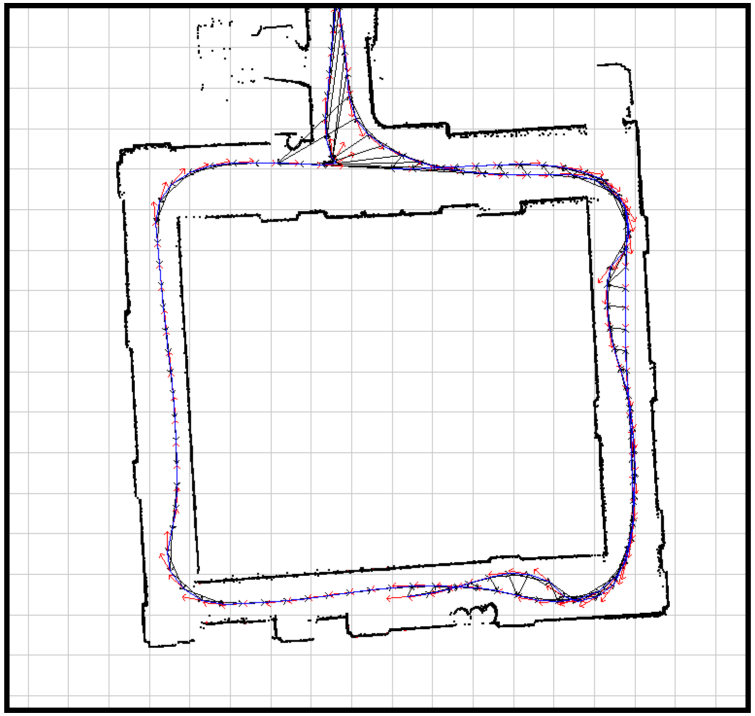
- Globally optimal estimate $\arg\max_{X_i} \left[P(D_{ij} \mid \overline{D_{ij}}) \right]$

Loop Closure

- Use scan patches to detect loop closure
- Add new position constraints
- Deform the network based on covariances of matches



Before loop closure



After loop closure

Efficient Map Recovery

- Minimize constraint function $J_{GraphSLAM}$ using standard optimization techniques (gradient descent, Levenberg Marquardt, conjugate gradient)

Mapping the Allen Center

Rao-Blackwellised Particle Filters

Rao-Blackwellized Mapping

Compute a posterior over the map and possible trajectories of the robot :

map and trajectory

measurements

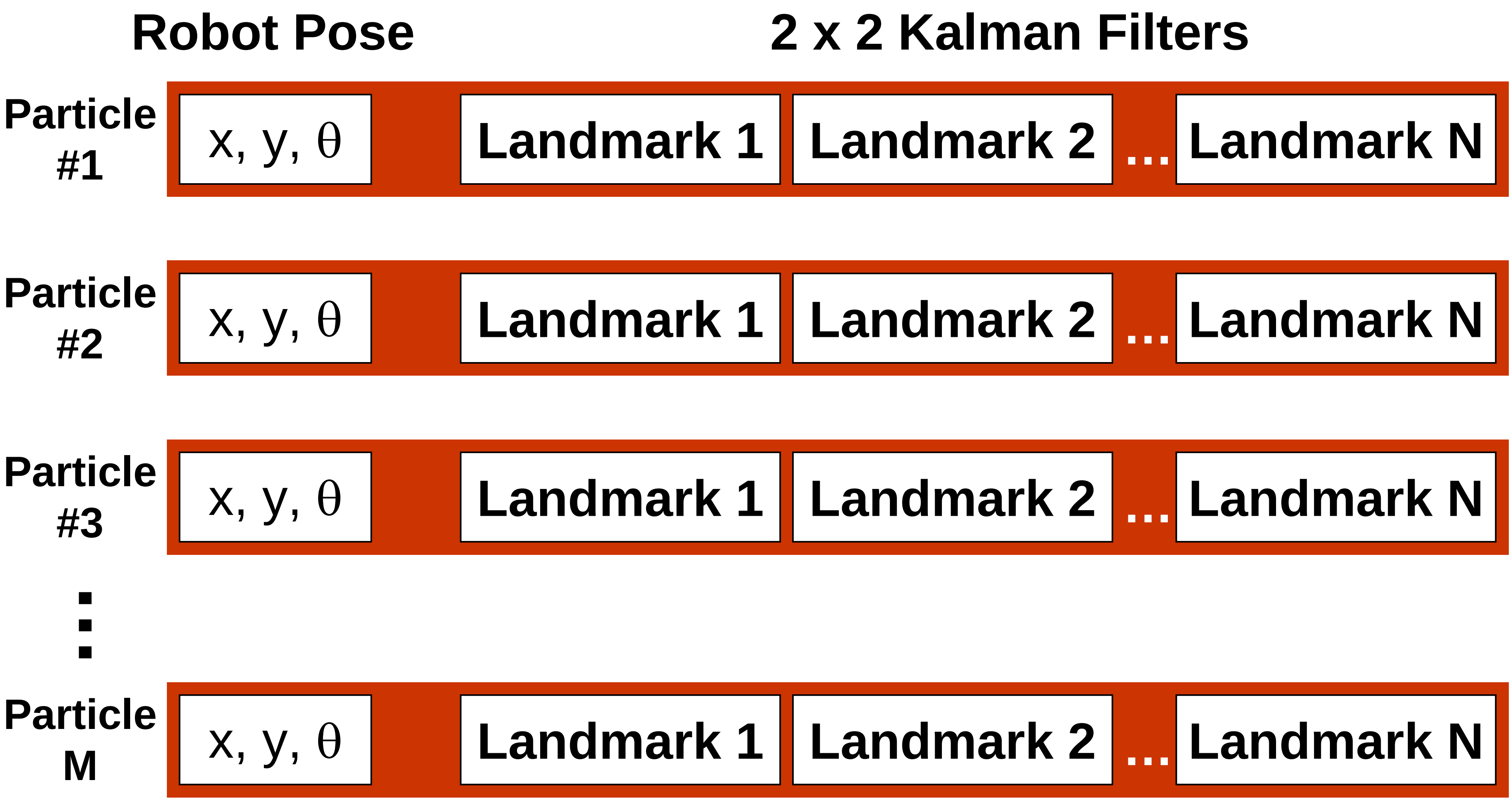
$$p(x_{1:t}, m \mid z_{1:t}, u_{0:t-1})$$
$$= p(m \mid x_{1:t}, z_{1:t}, u_{0:t-1})\, p(x_{1:t} \mid z_{1:t}, u_{0:t-1})$$

map

robot motion

trajectory

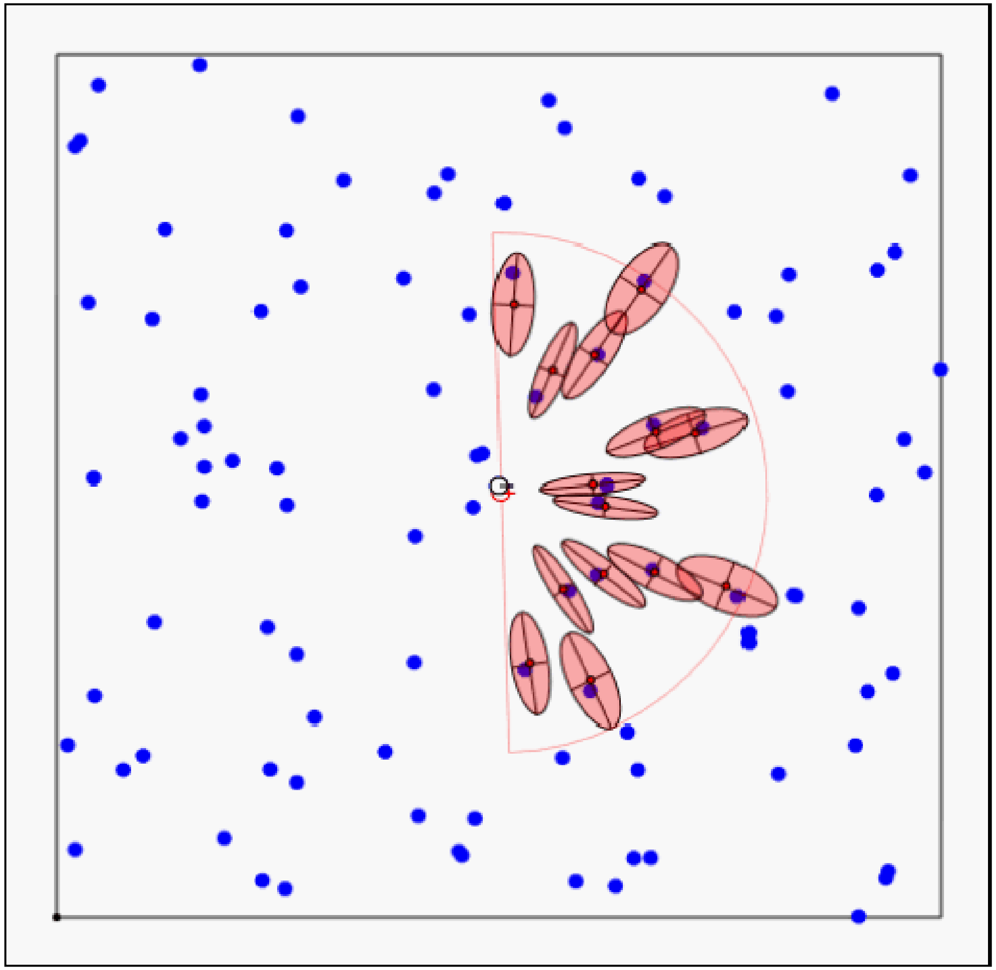
FastSLAM



[Begin courtesy of Mike Montemerlo]

FastSLAM – Simulation

- Up to 100,000 landmarks
- 100 particles
- 10^3 times fewer parameters than EKF SLAM



Blue line = true robot path
Red line = estimated robot path
Black dashed line = odometry

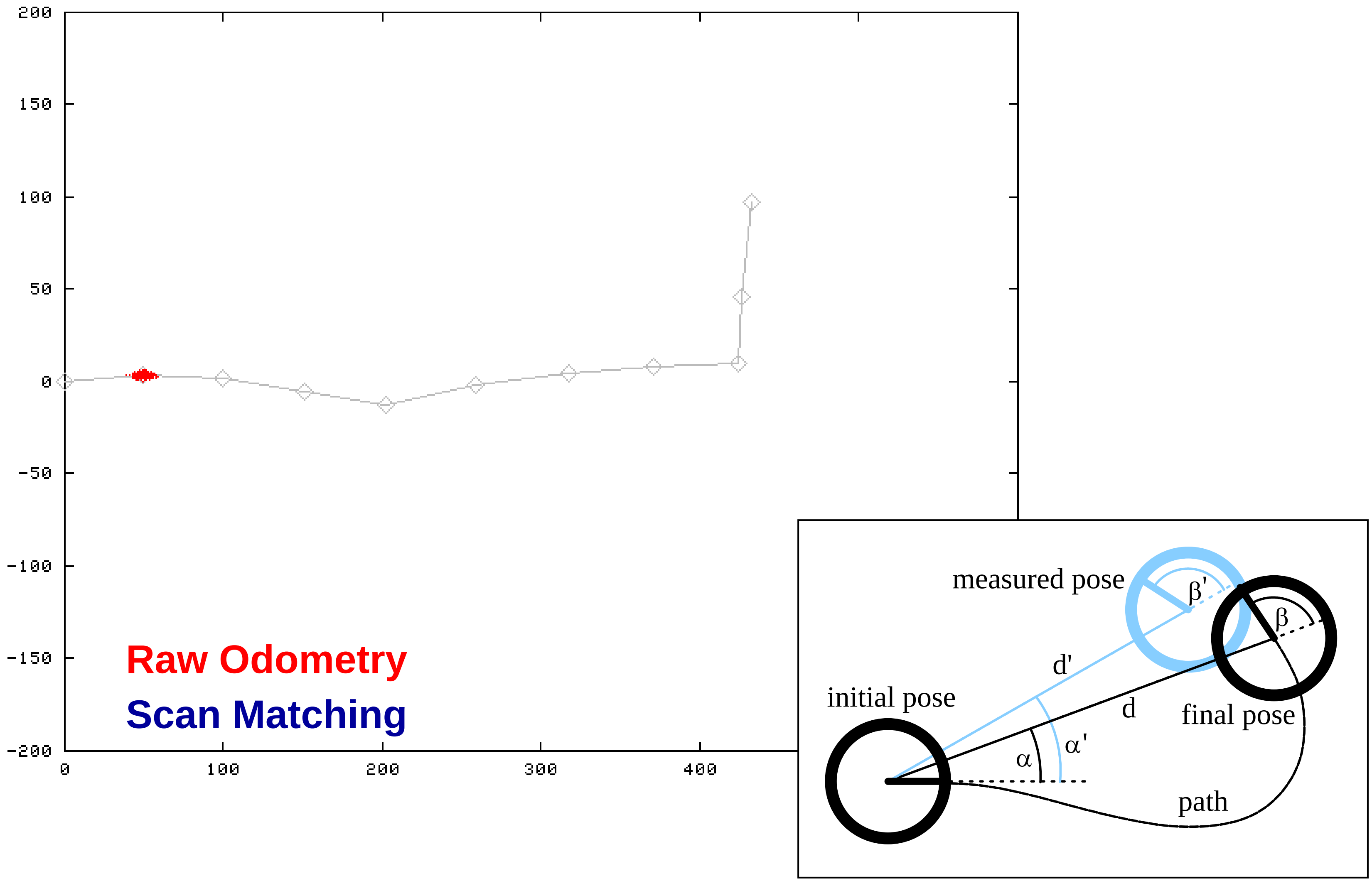
Victoria Park Results

- 4 km traverse
- 100 particles
- Uses negative evidence to remove spurious landmarks

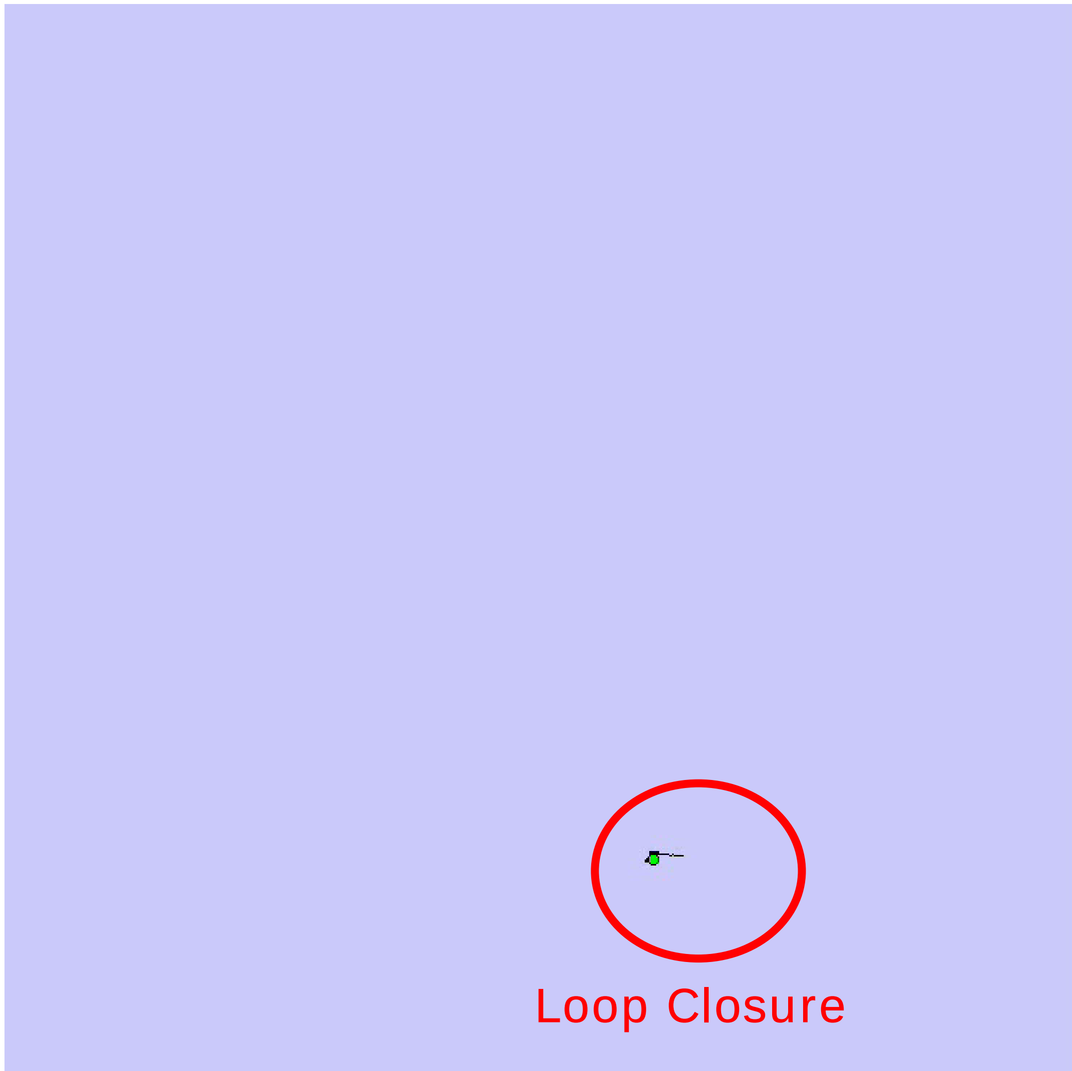
Blue path = odometry
Red path = estimated path

[End courtesy of Mike Montemerlo]

Motion Model for Scan Matching

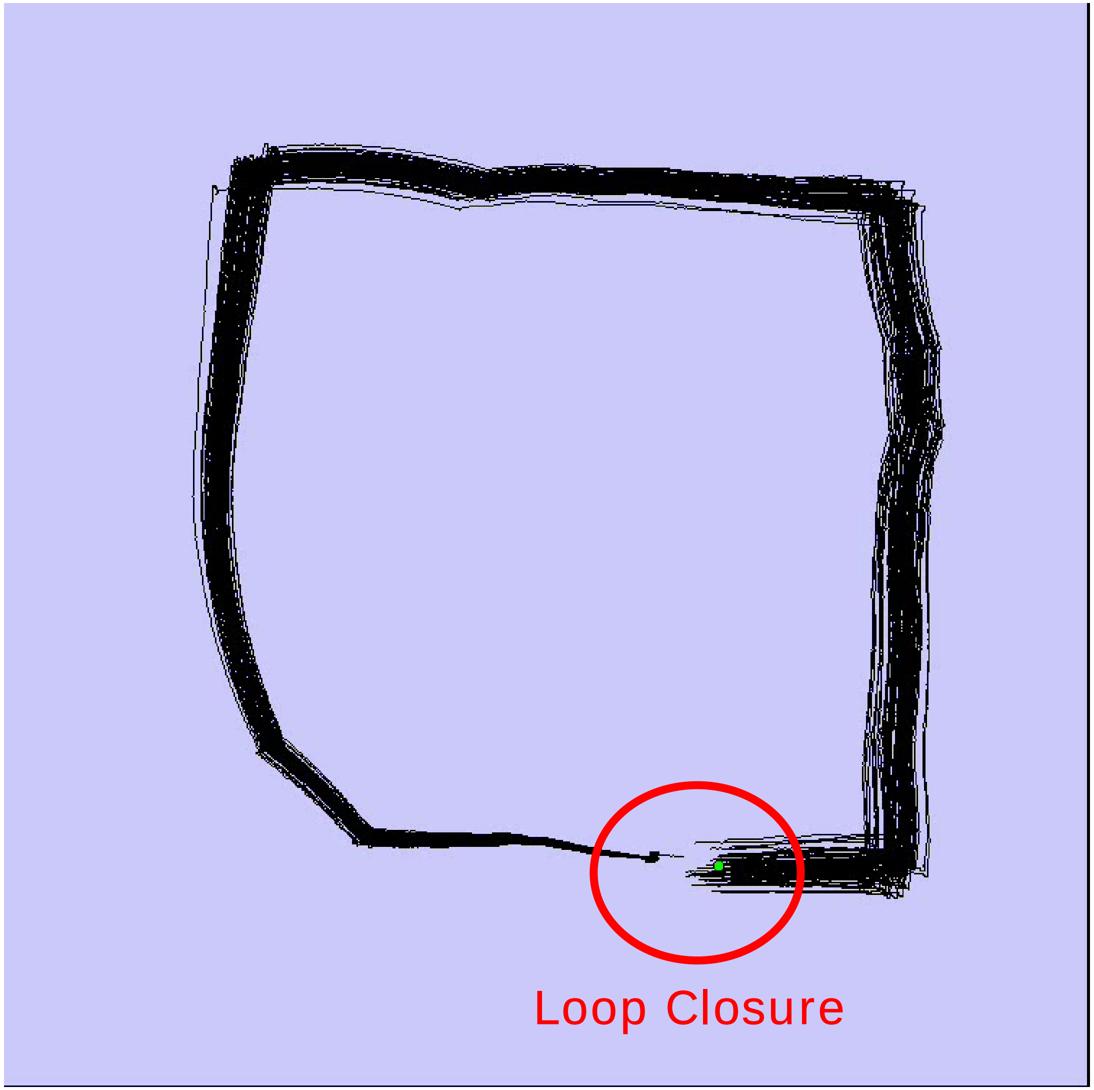


Rao-Blackwellized Mapping with Scan-Matching



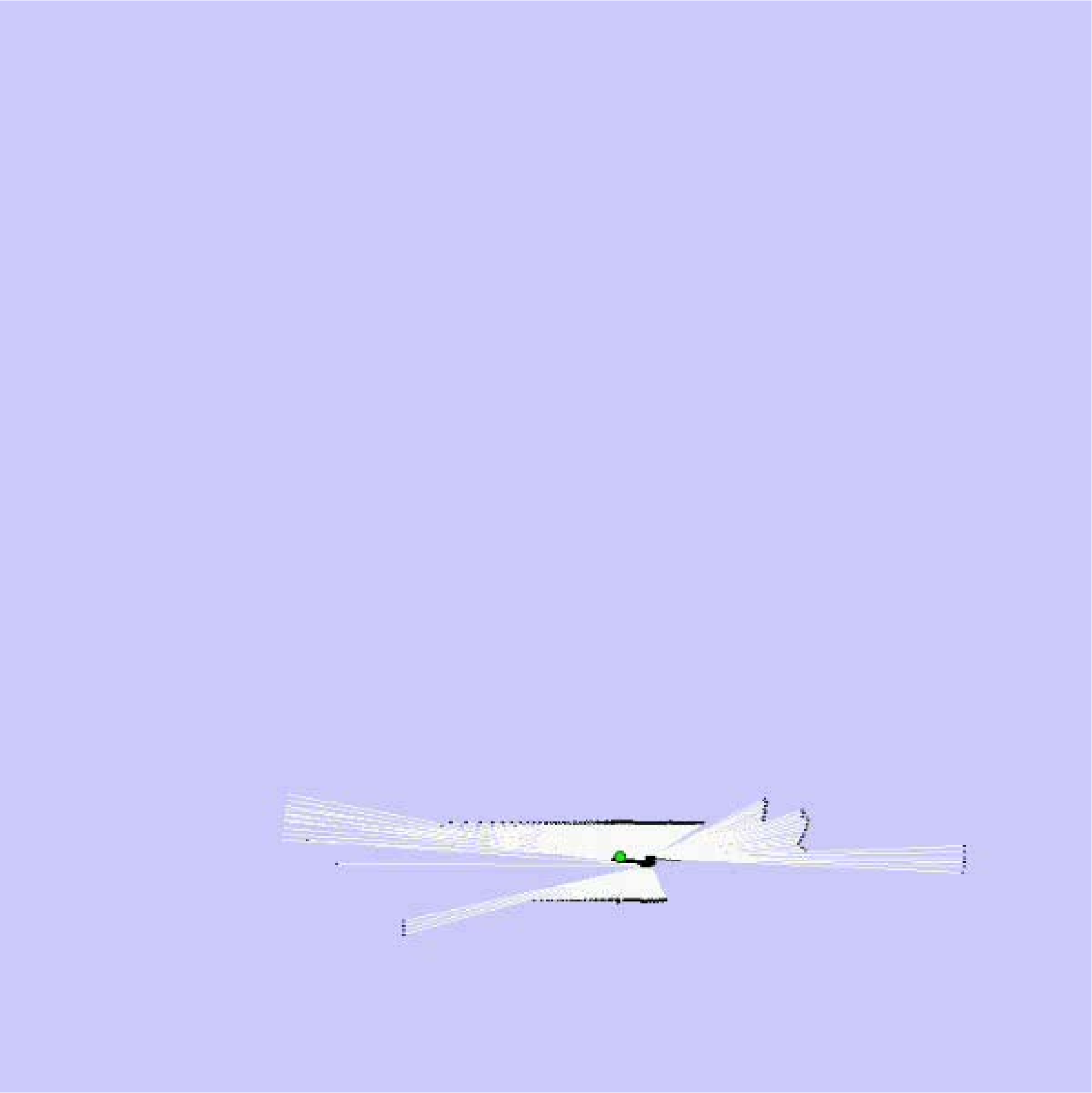
Map: Intel Research Lab Seattle

Rao-Blackwellized Mapping with Scan-Matching



Map: Intel Research Lab Seattle

Rao-Blackwellized Mapping with Scan-Matching



Map: Intel Research Lab Seattle

Example (Intel Lab)

- **15 particles**
- four times faster than real-time P4, 2.8GHz
- 5cm resolution during scan matching
- 1cm resolution in final map

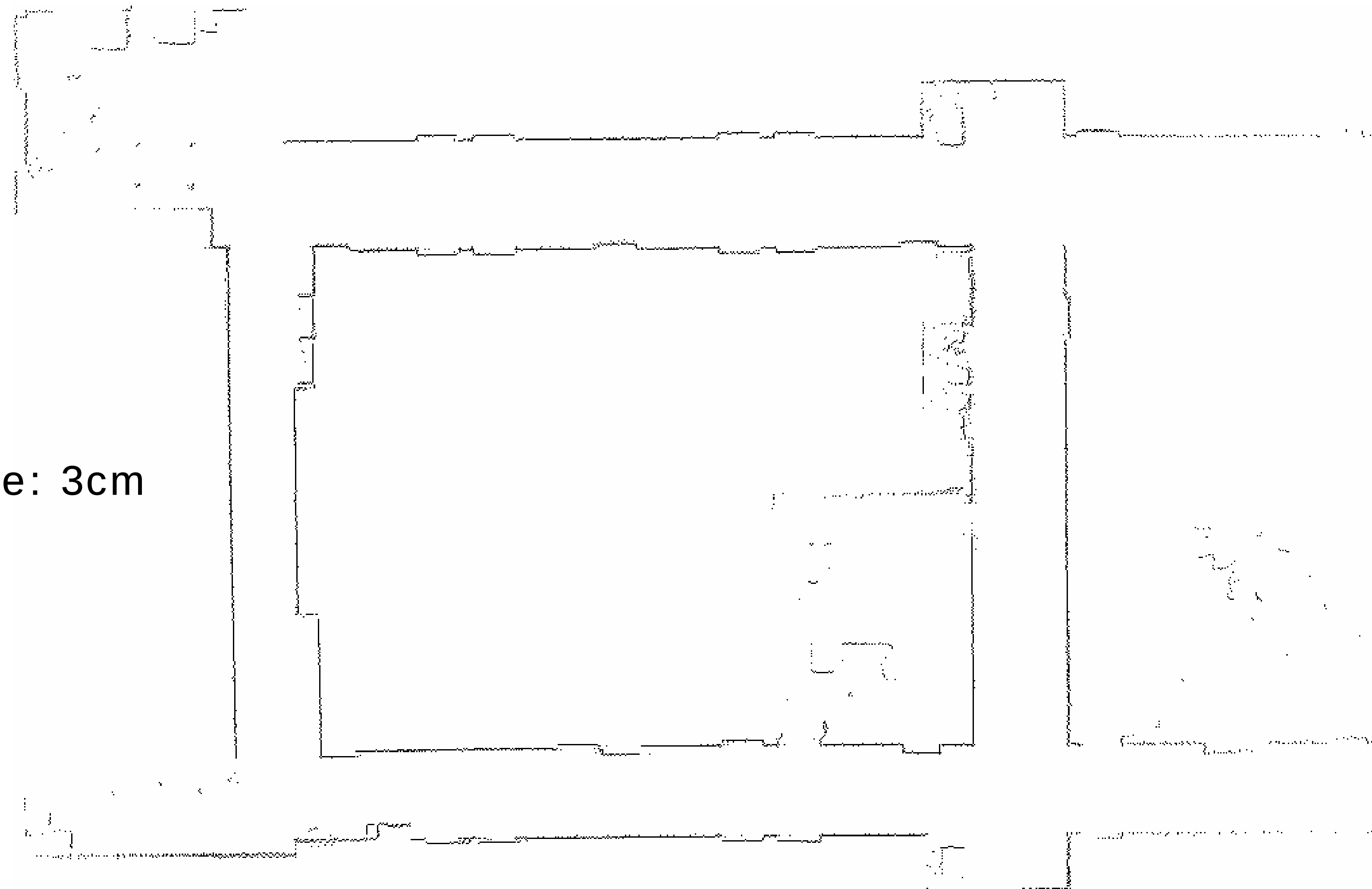
joint work with Giorgio Grisetti

Outdoor Campus Map

- **30 particles**
- $250x250m^2$
- 1.088 miles (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map

joint work with Giorgio Grisetti

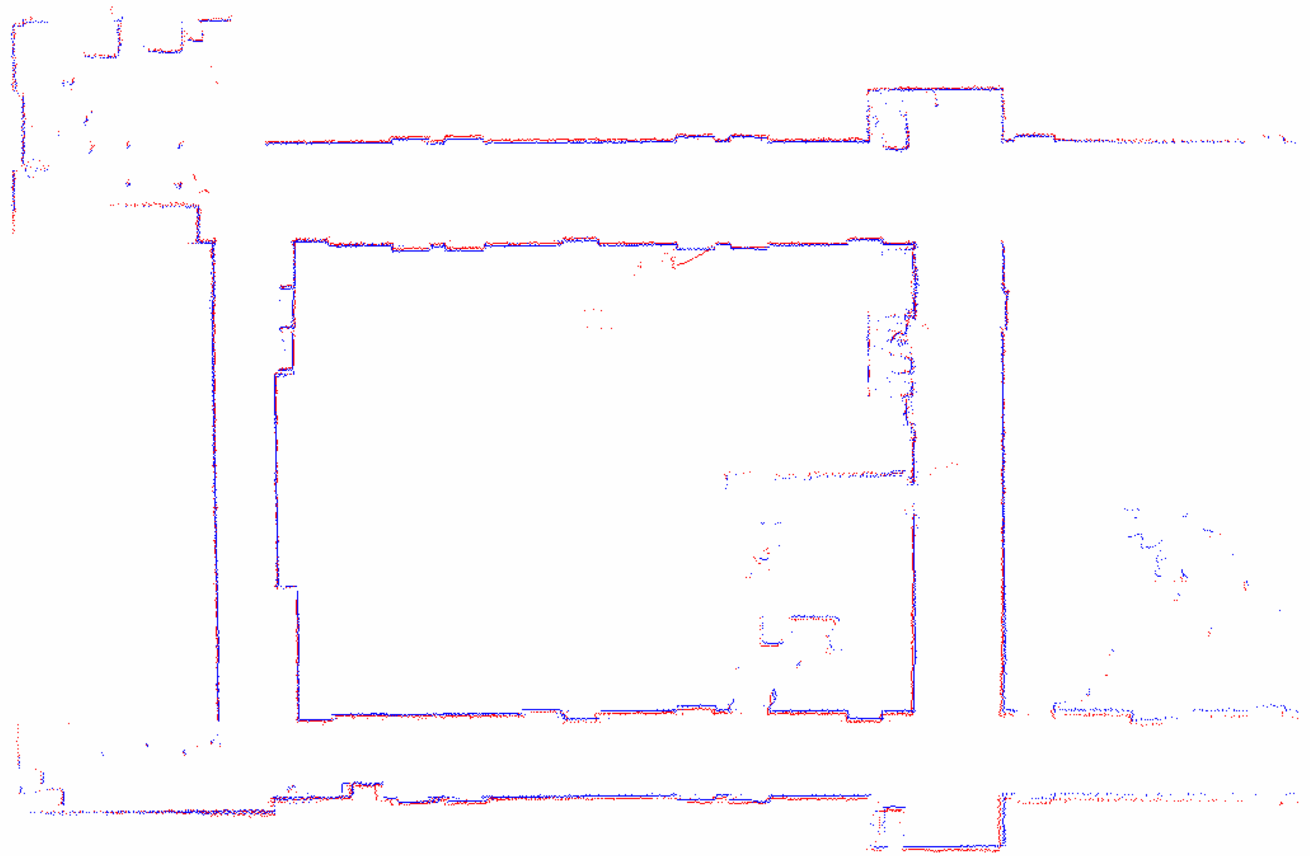
DP-SLAM [Eliazar & Parr]



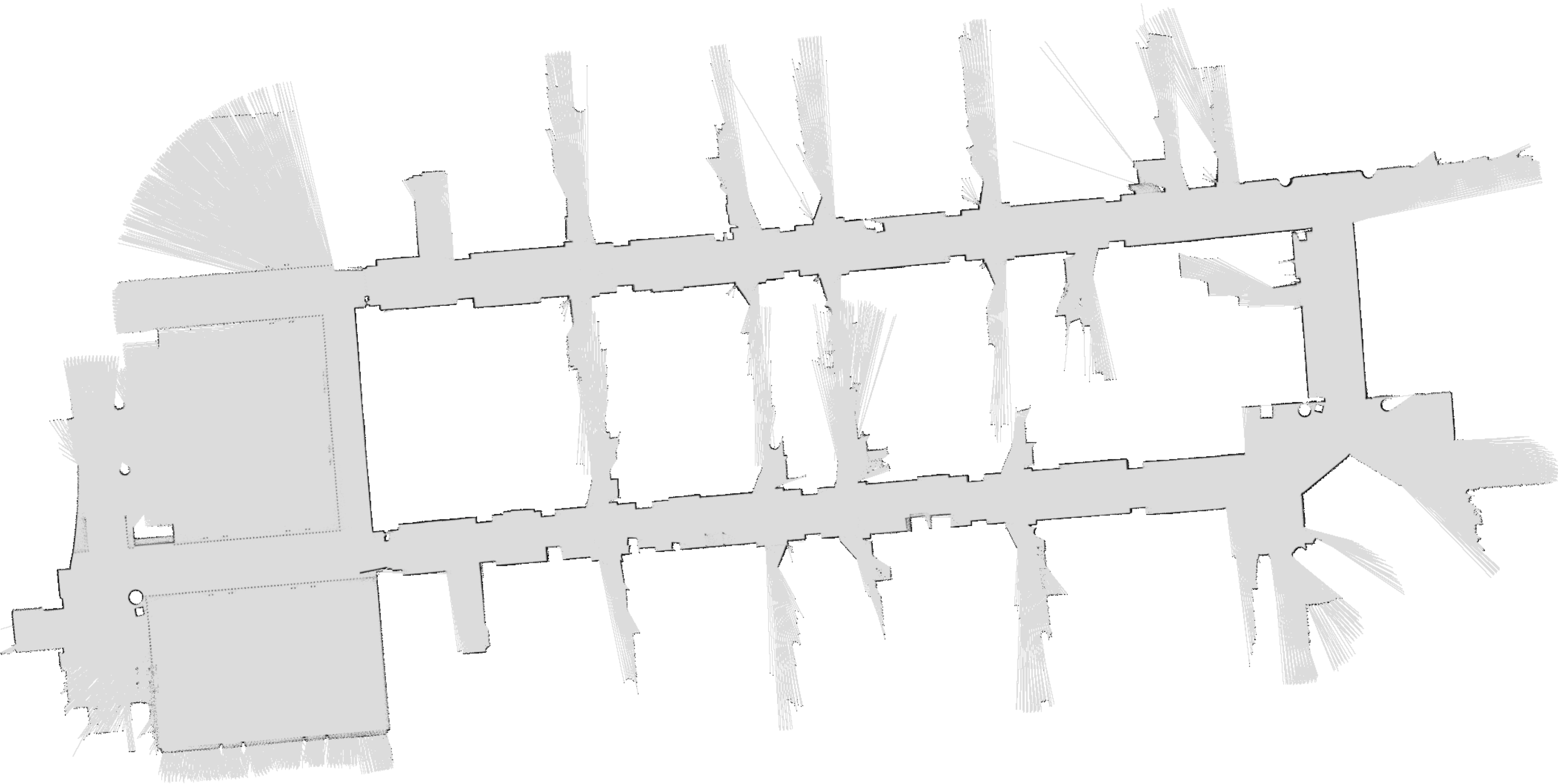
scale: 3cm

Runs at real-time speed on 2.4GHz Pentium 4 at 10cm/s

Consistency

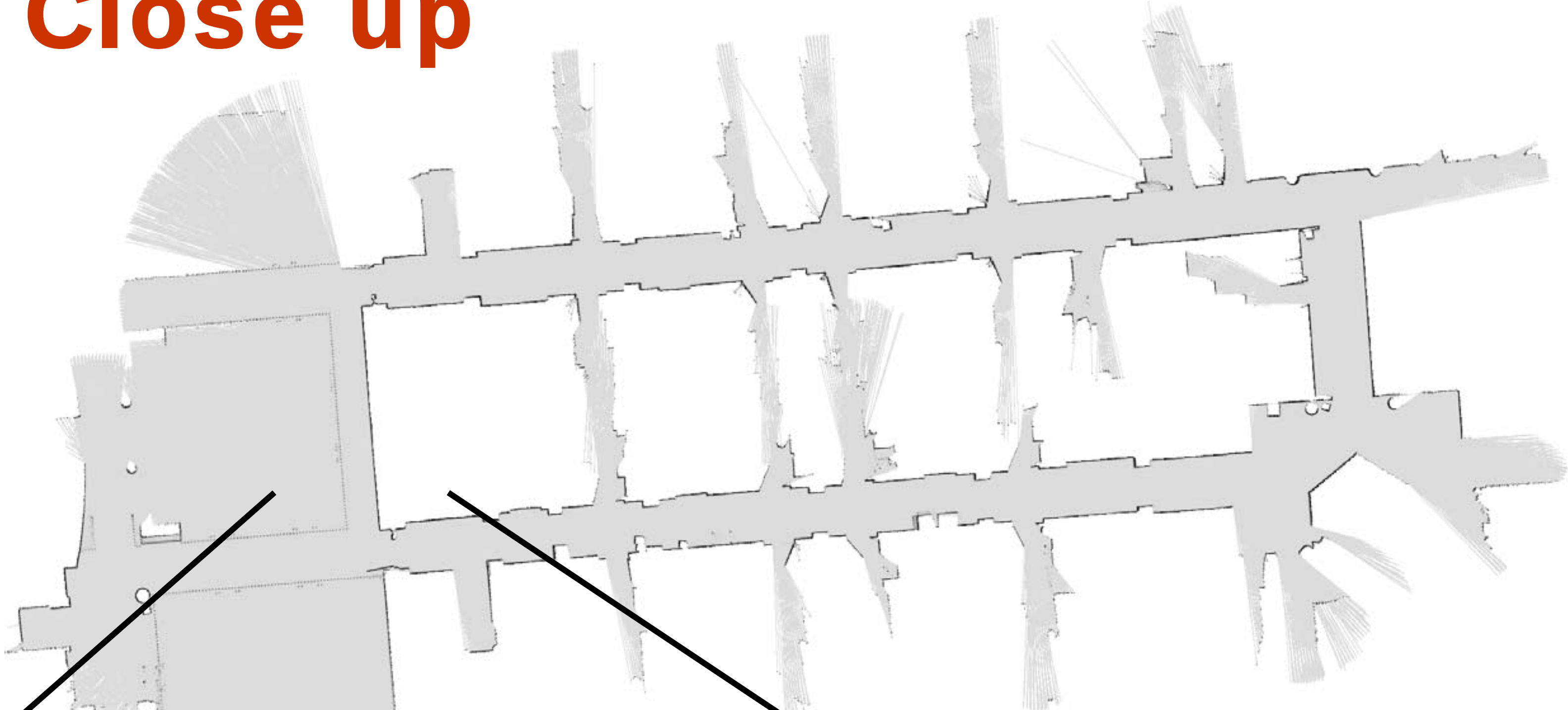


Results obtained with DP-SLAM 2.0 (offline)



Eliazar & Parr, 04

Close up

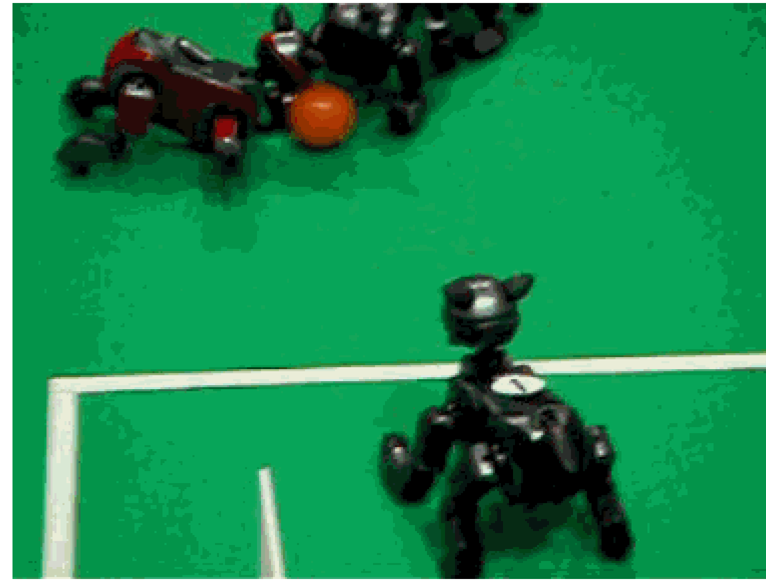


End courtesy of Eliazar & Parr

Fast-SLAM Summary

- Full and online version of SLAM
- Factorizes posterior into robot trajectories (particles) and map (EKFs).
- Landmark locations are independent!
- More efficient proposal distribution through Kalman filter prediction
- Data association per particle

Ball Tracking in RoboCup



- Extremely noisy (nonlinear) motion of observer
- Inaccurate sensing, limited processing power
- Interactions between target and

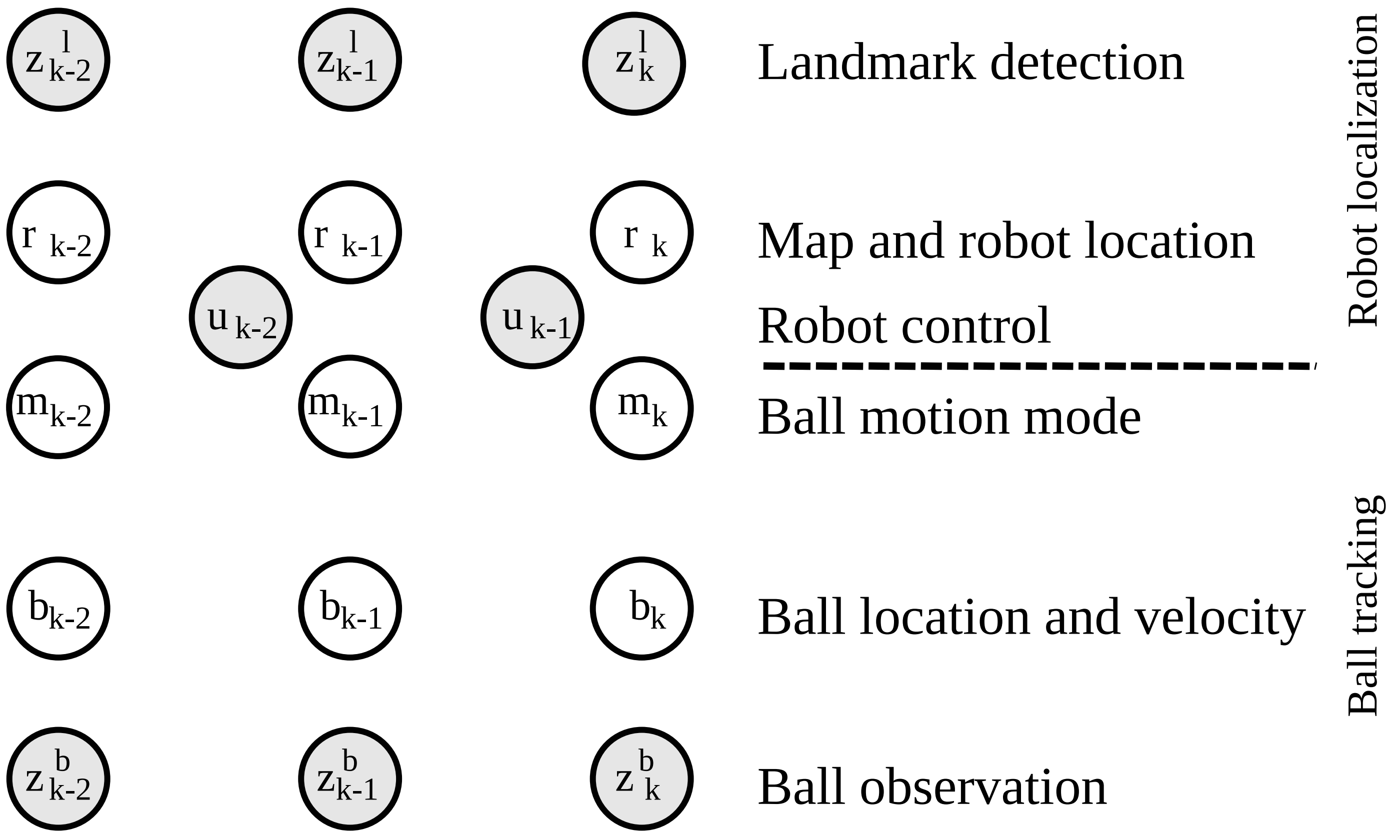
Goal: Unified framework for modeling the ball and its interactions.

t

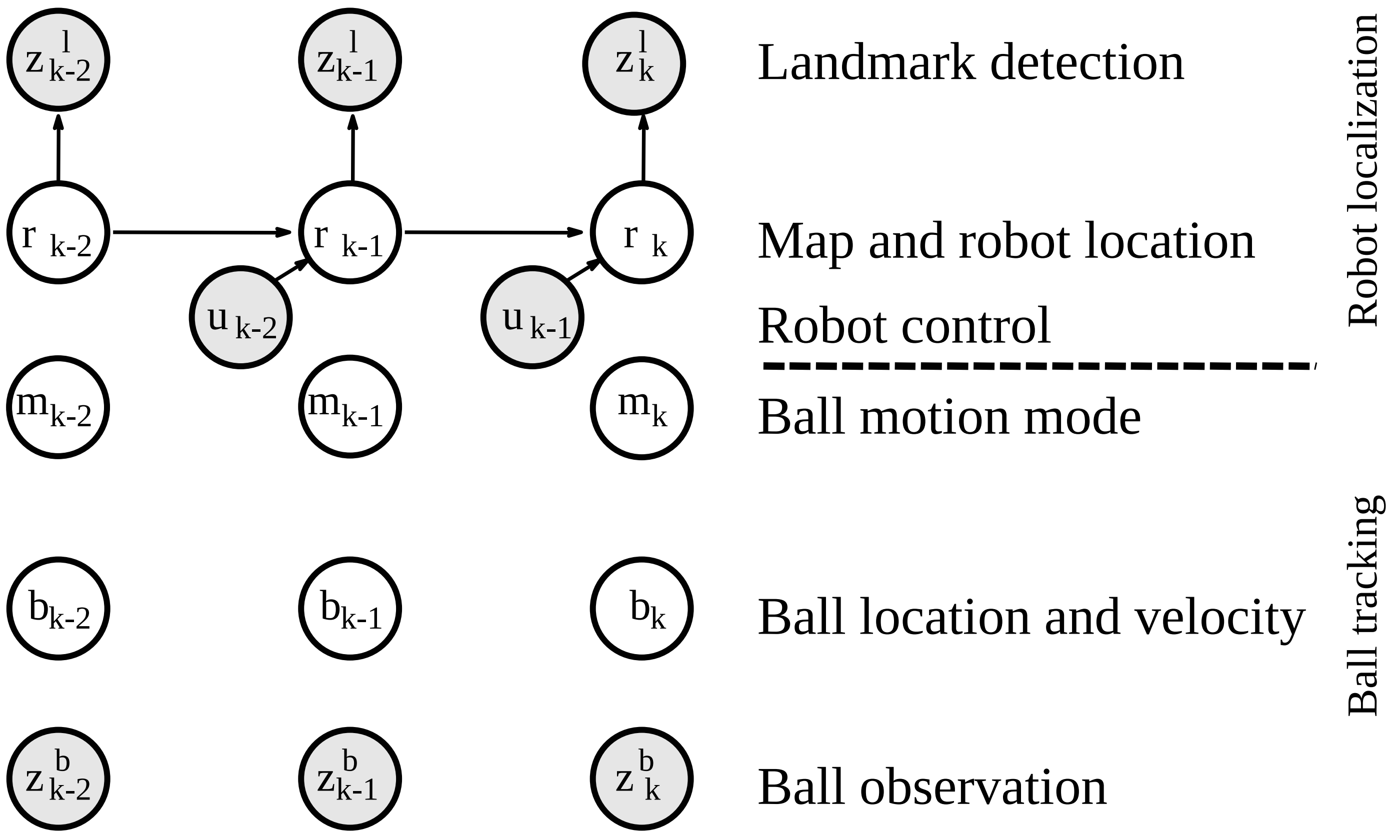
Tracking Techniques

- Kalman Filter
  - Highly efficient, robust (even for nonlinear)
  - Uni-modal, limited handling of nonlinearities
- Particle Filter
  - Less efficient, highly robust
  - Multi-modal, nonlinear, non-Gaussian
- Rao-Blackwellised Particle Filter, MHT
  - Combines PF with KF
  - Multi-modal, highly efficient

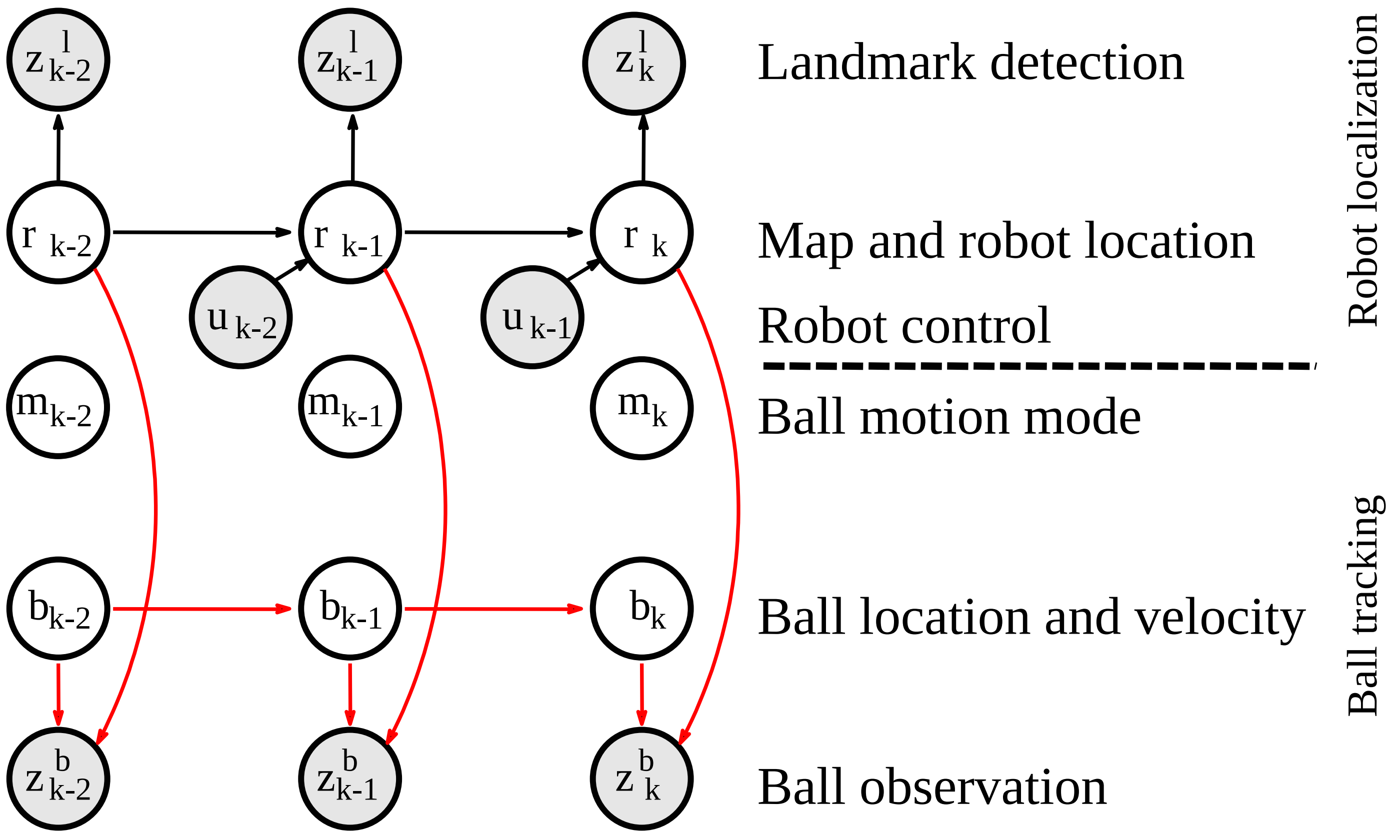
Dynamic Bayes Network for Ball Tracking



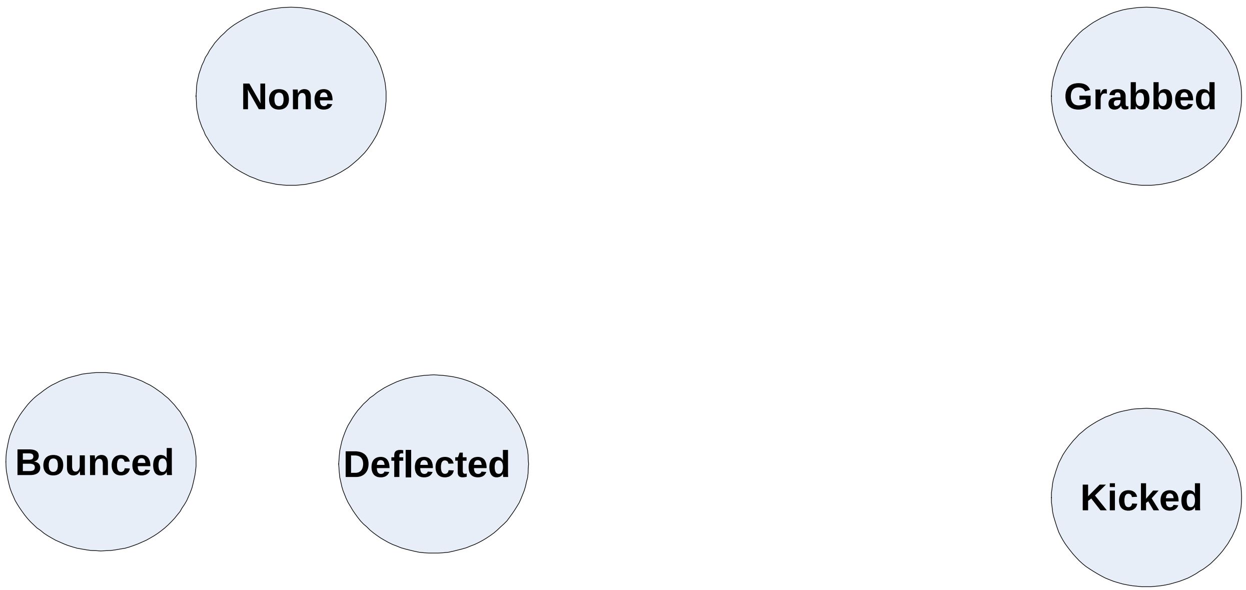
Robot Location



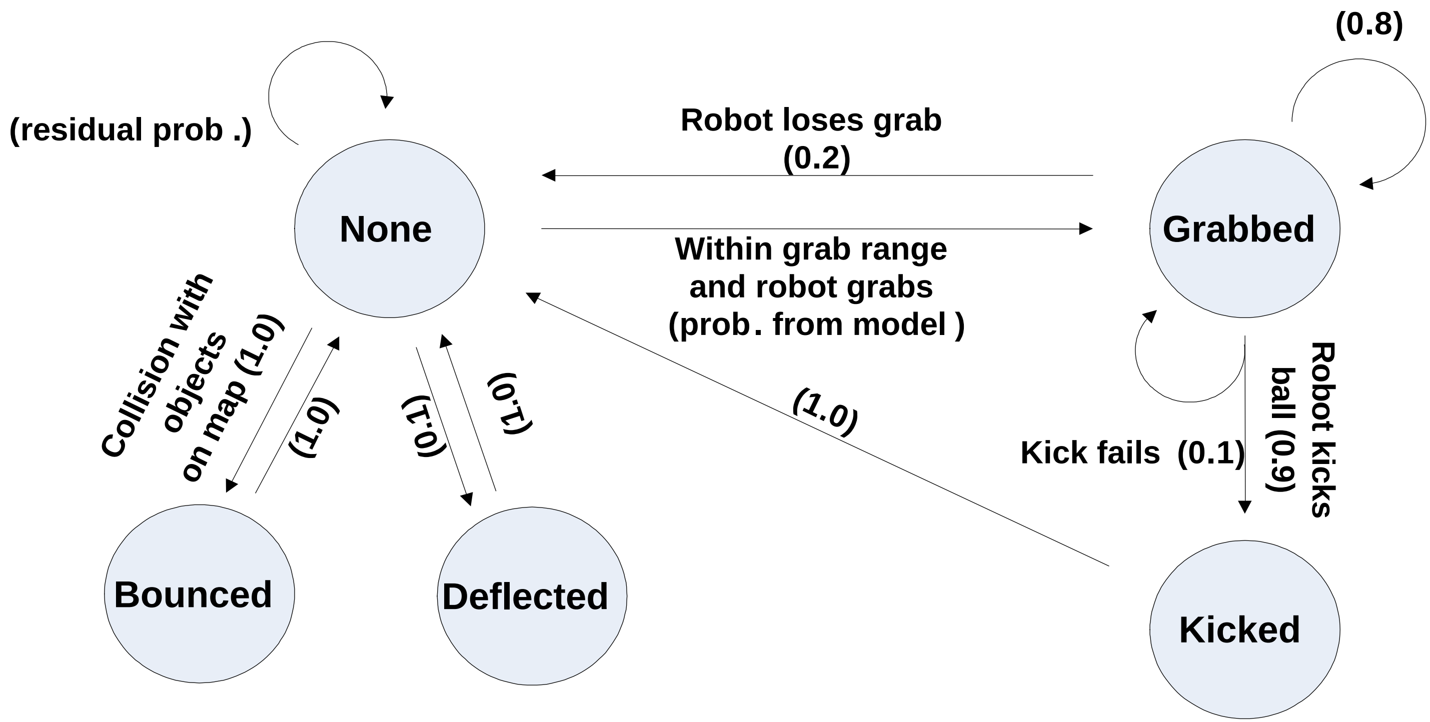
Robot and Ball Location (and velocity)



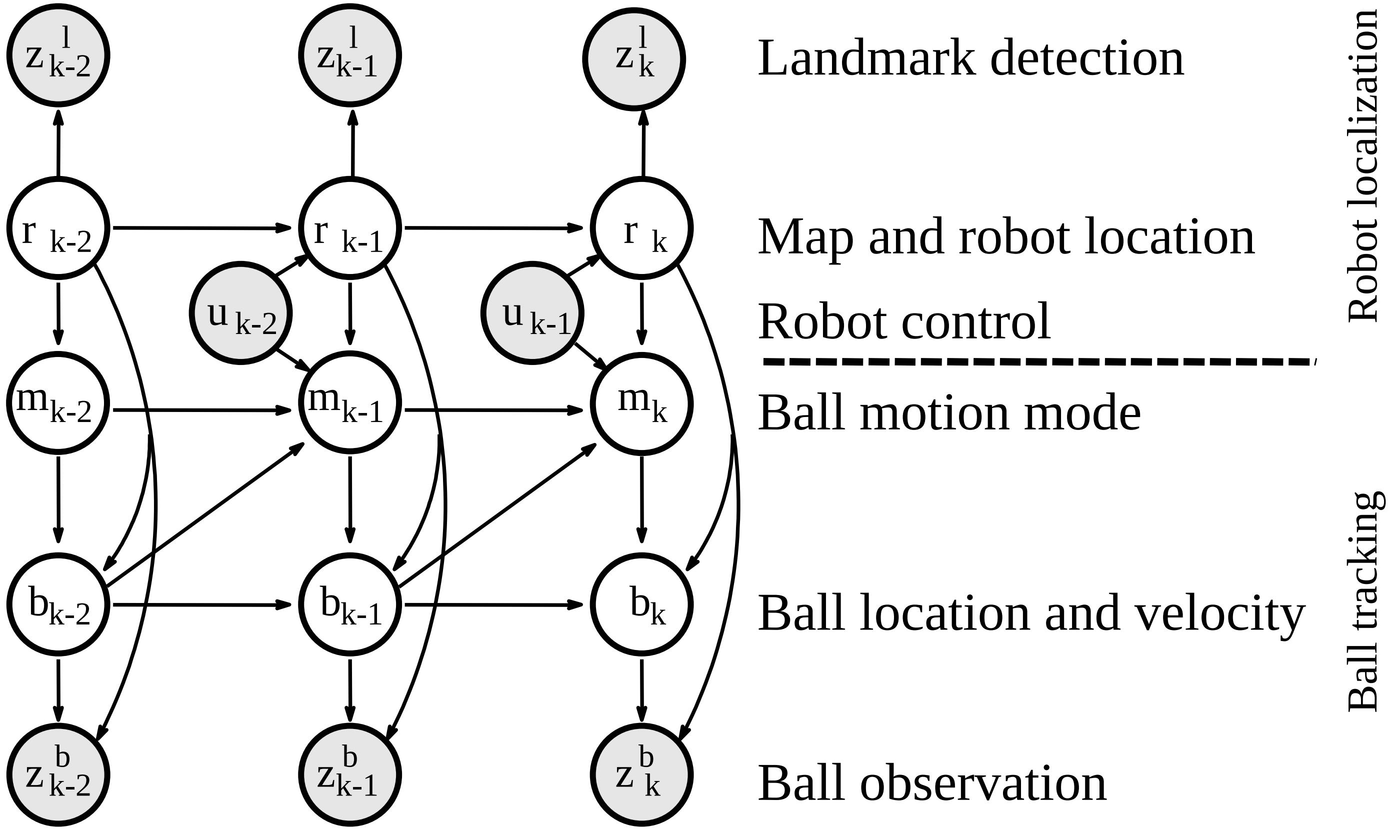
Ball-Environment Interactions



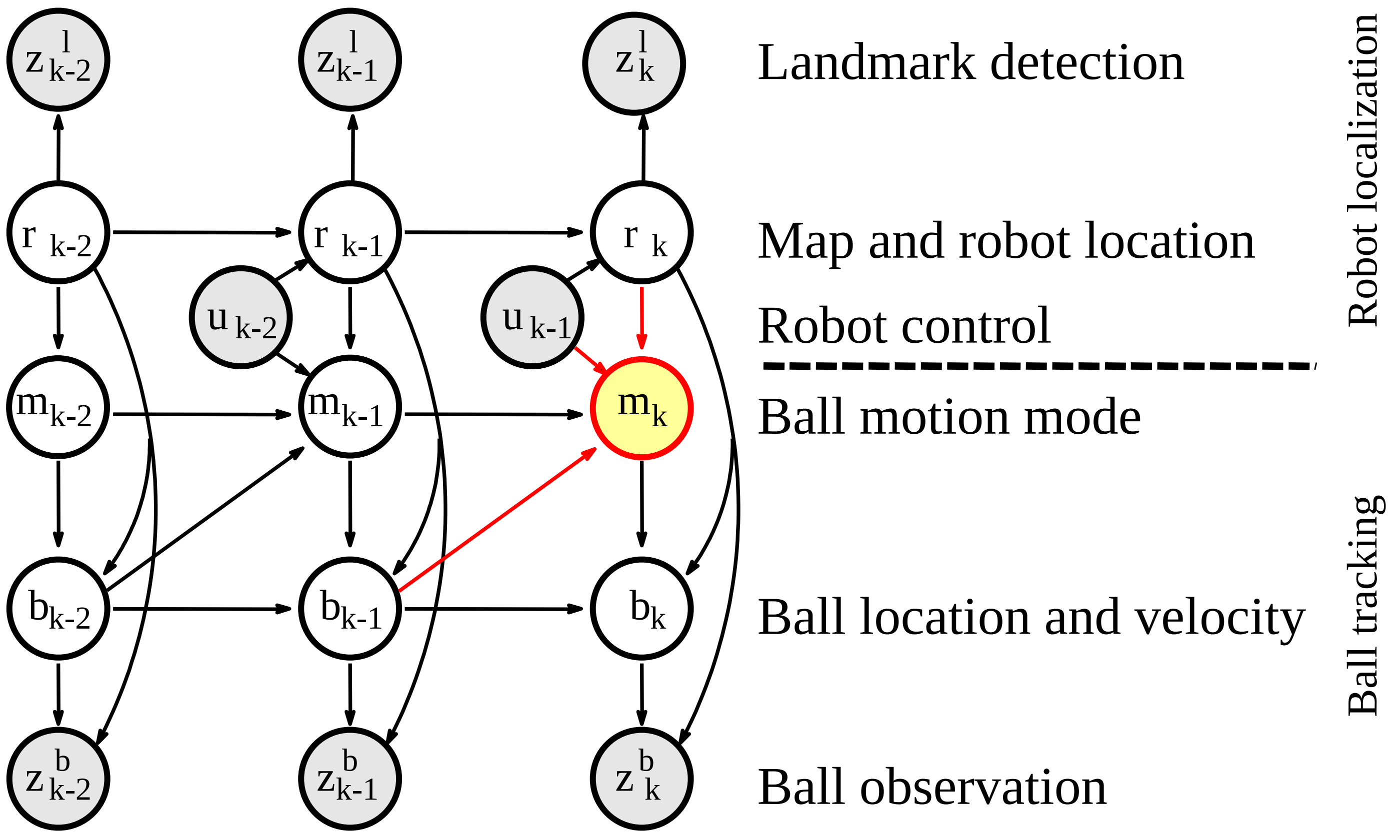
Ball-Environment Interactions



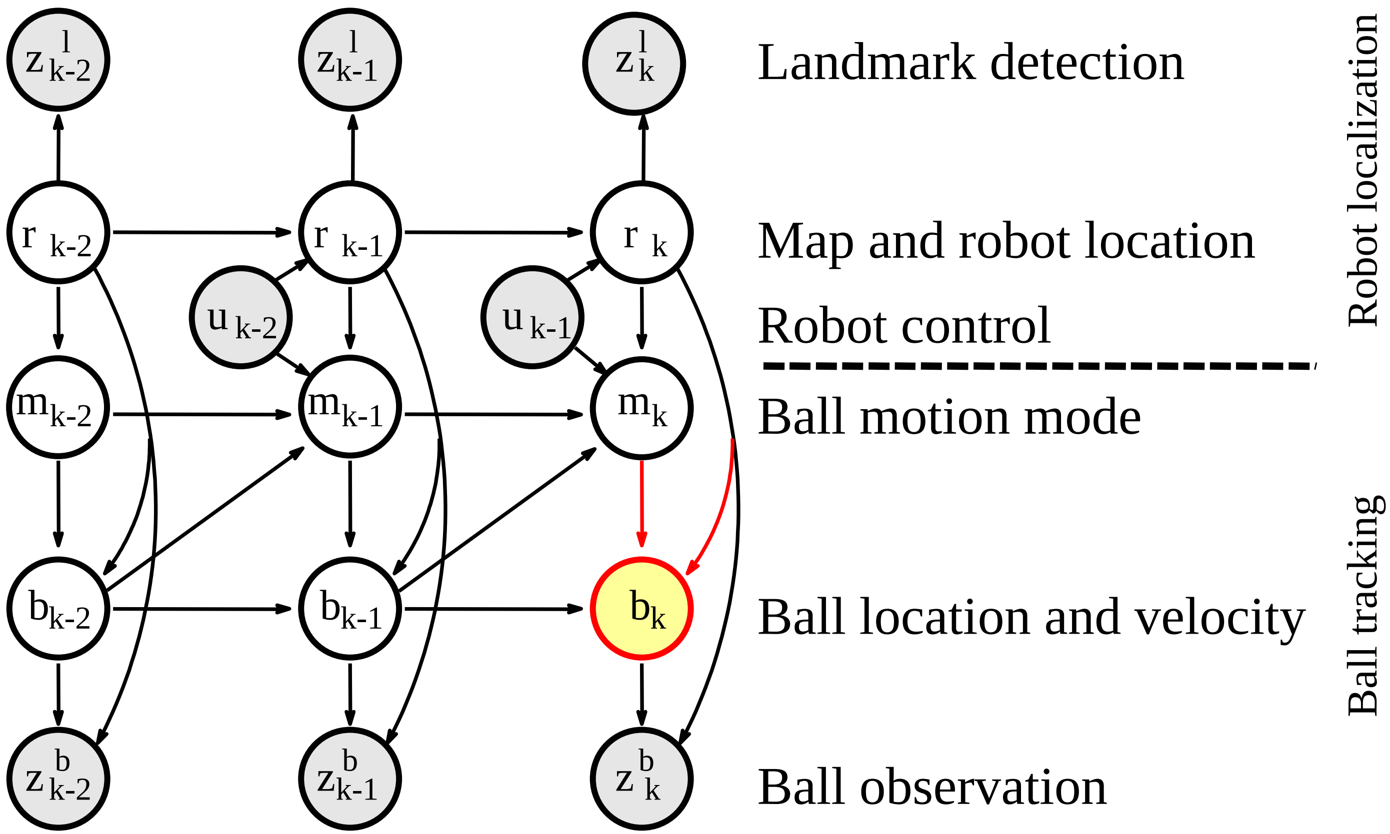
Integrating Discrete Ball Motion Mode



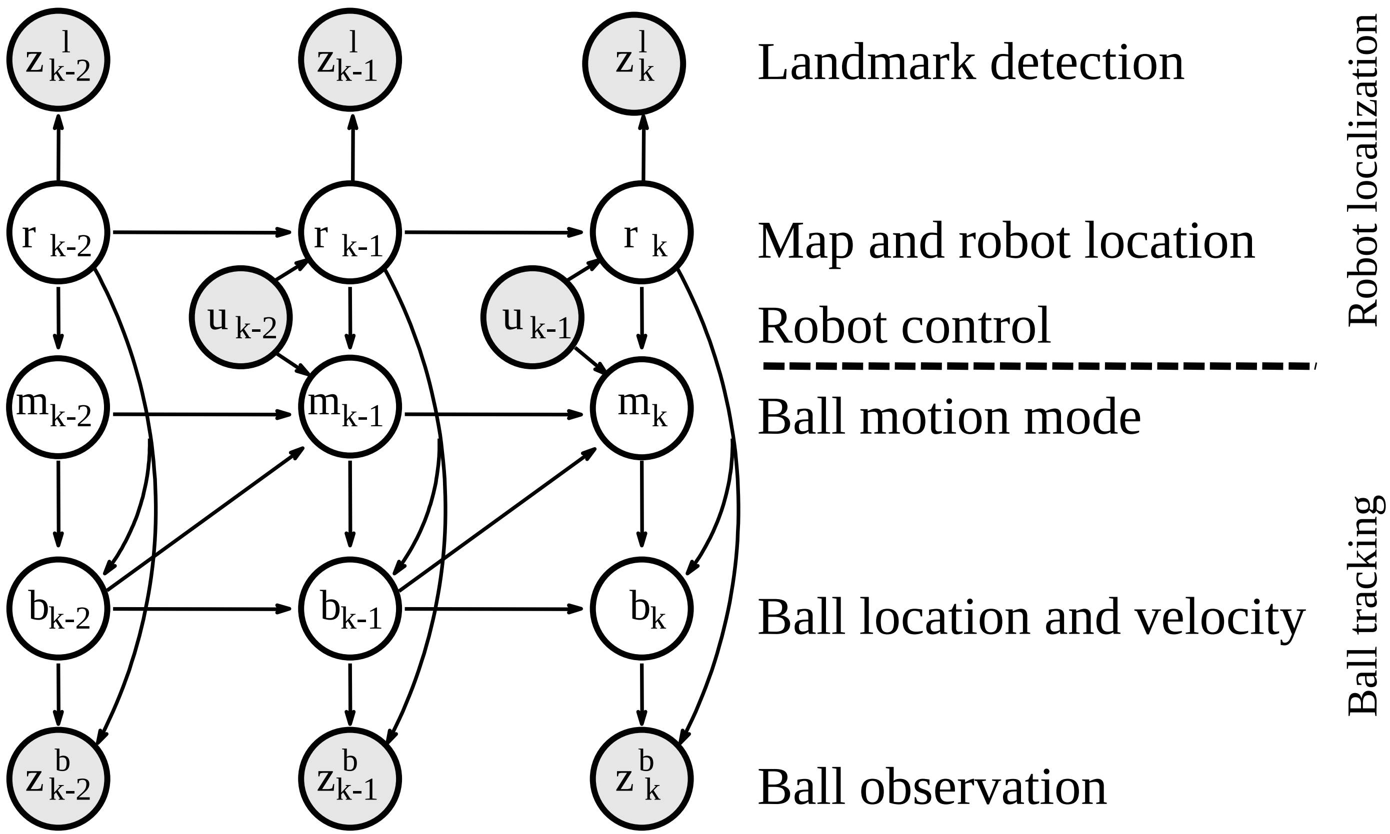
Grab Example (1)



Grab Example (2)



Inference: Posterior Estimation



$$\boxed{p(b_k, m_k, r_k \mid z^b_{1:k}, z^l_{1:k}, u_{1:k-1})}$$

Rao-Blackwellised PF for Inference

- Represent posterior by random samples
- Each sample

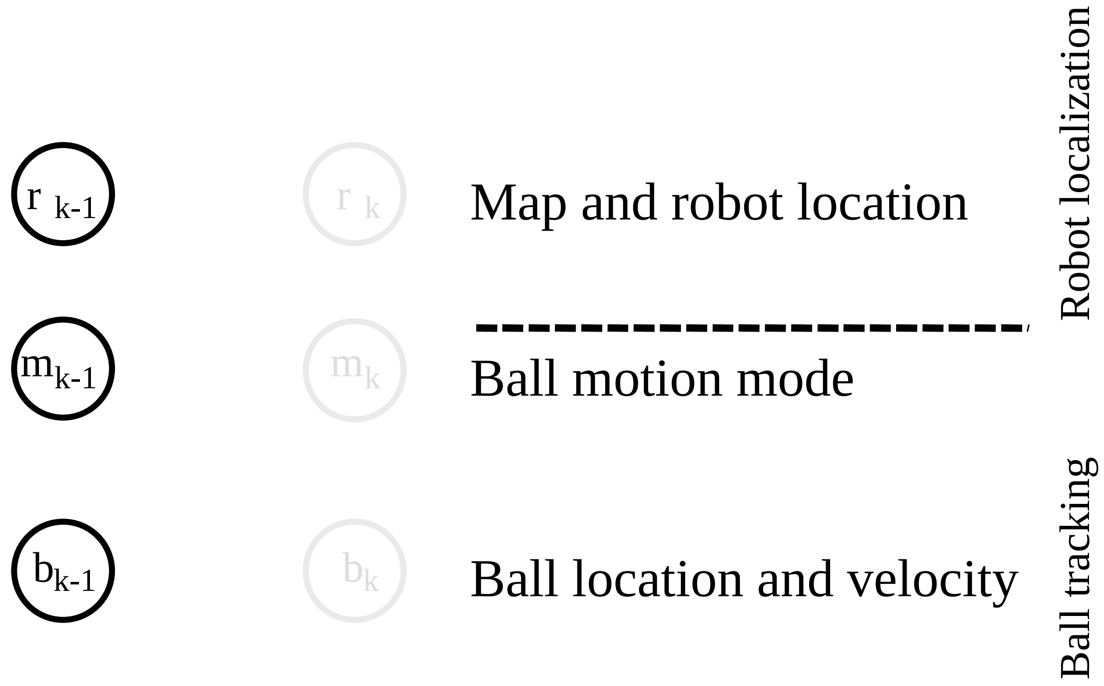
$$s_i = \langle r_i, m_i, b_i \rangle = \langle \langle x, y, \theta \rangle_i, m_i, \langle \mu, \Sigma \rangle_i \rangle$$

contains robot location, ball mode, ball Kalman filter

- Generate individual components of a particle stepwise using the factorization

$$p(b_k, m_{1:k}, r_{1:k} \mid z_{1:k}, u_{1:k-1}) =$$
$$p(b_k \mid m_{1:k}, r_{1:k}, z_{1:k}, u_{1:k-1})\, p(m_{1:k} \mid r_{1:k}, z_{1:k}, u_{1:k-1}) \cdot p(r_{1:k} \mid z_{1:k}, u_{1:k-1})$$

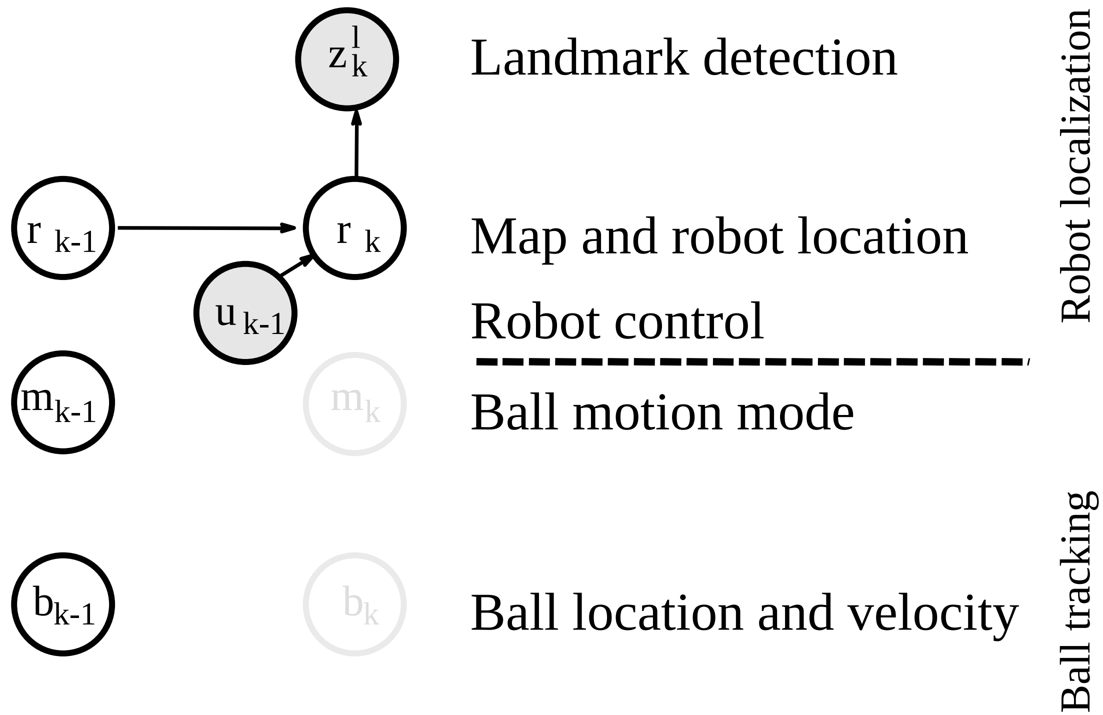
Rao-Blackwellised Particle Filter for Inference



- Draw a sample from the previous sample set:

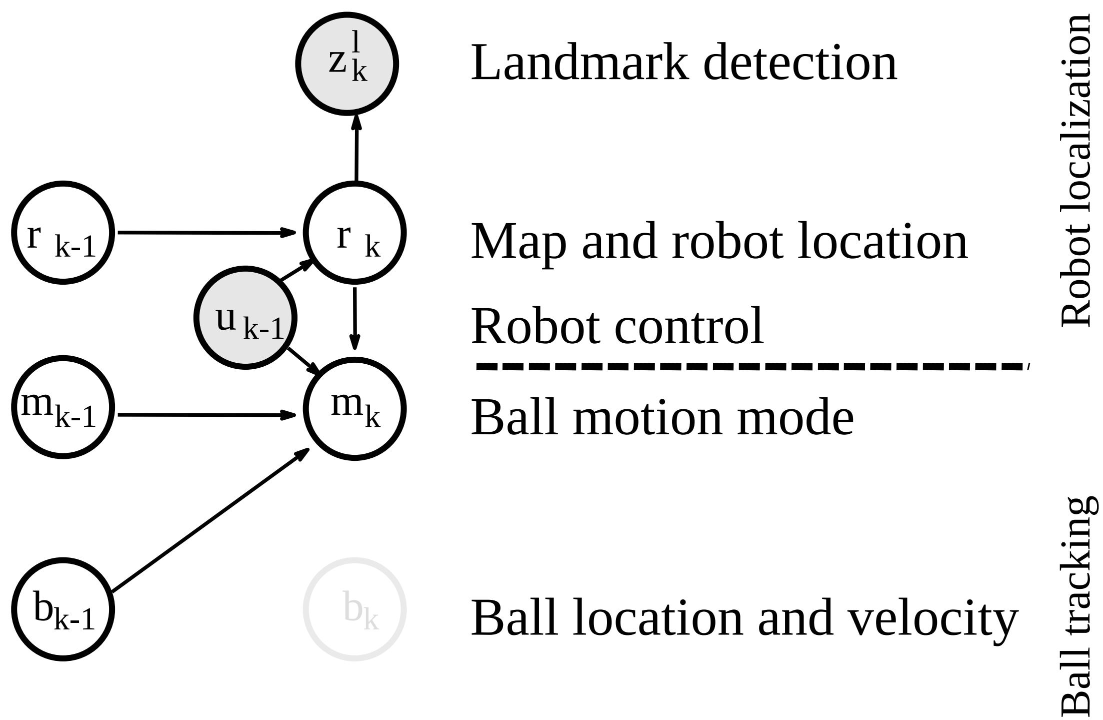
$$\left\langle r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)} \right\rangle$$

Generate Robot Location



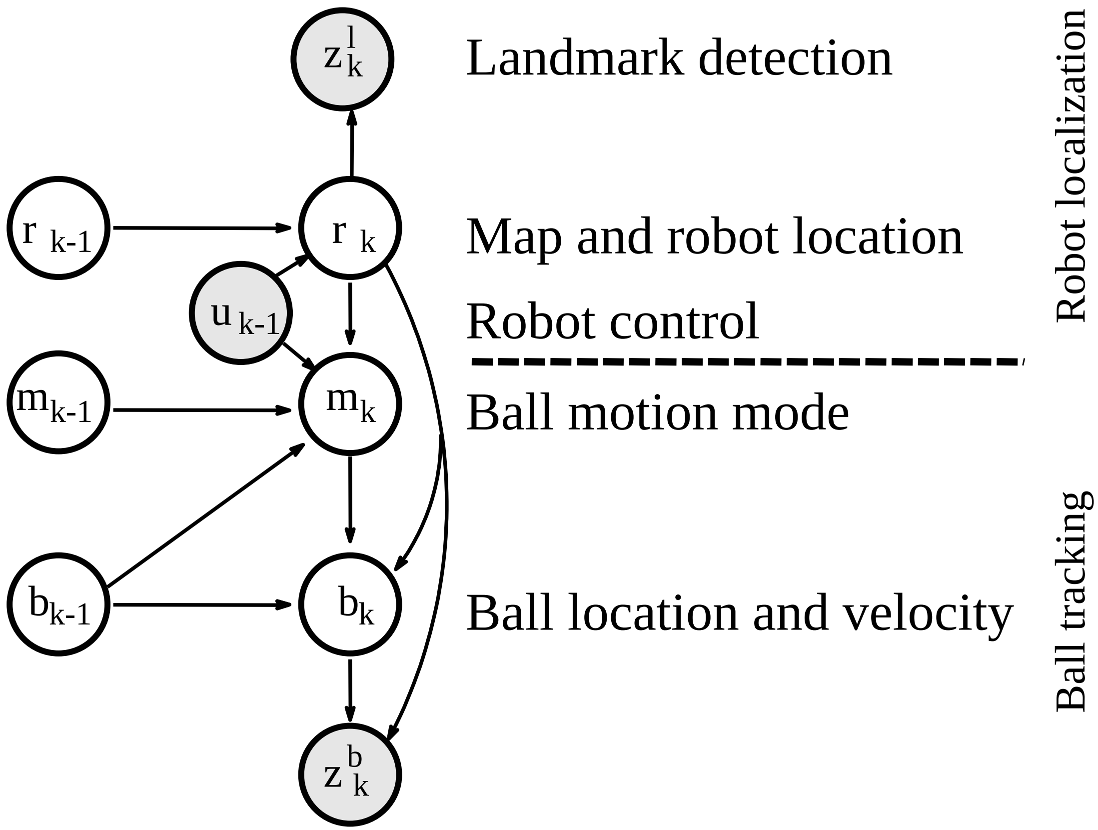
$$r_k^{(i)} \sim p(r_k \mid r_{k-1}^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \left\langle r_k^{(i)}, \_, \_ \right\rangle$$

Generate Ball Motion Model



$$m_k^{(i)} \sim p(m_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, z_k, u_{k-1}) \quad \Rightarrow \quad \left\langle r_k^{(i)}, m_k^{(i)}, \_ \right\rangle$$

Update Ball Location and Velocity



$$b_k^{(i)} \sim p(b_k \mid r_k^{(i)}, m_k^{(i)}, b_{k-1}^{(i)}, z_k) \quad \Rightarrow \quad \left\langle r_k^{(i)}, m_k^{(i)}, b_k^{(i)} \right\rangle$$

Importance Resampling

- Weight sample by

$$w_k^{(i)} \propto p(z_k^l \mid r_k^{(i)})$$

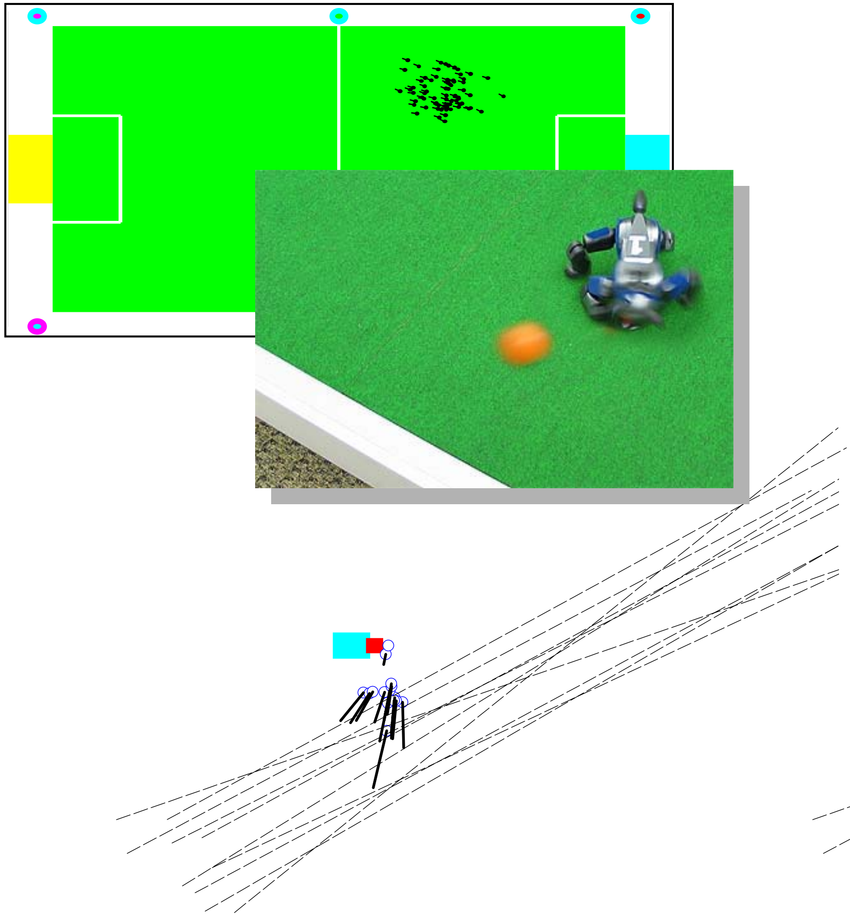
if observation is landmark detection and by

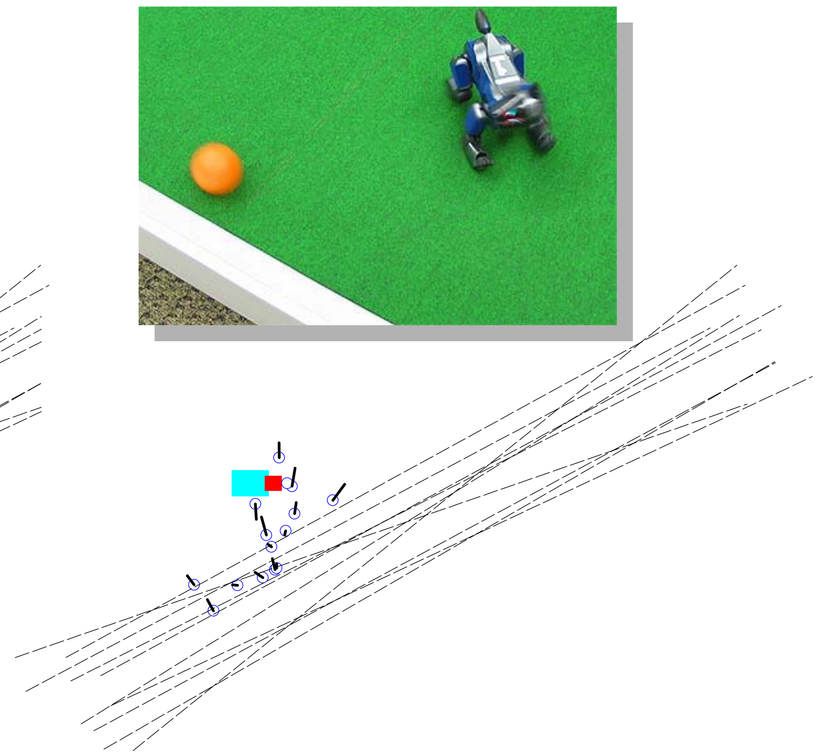
$$w_k^{(i)} \propto \sum_{M_k} p(z_k^b \mid M_k, r_k^{(i)}, b_{k-1}^{(i)}) p(M_k \mid r_k^{(i)}, m_{k-1}^{(i)}, b_{k-1}^{(i)}, u_{k-1})$$

if observation is ball detection.

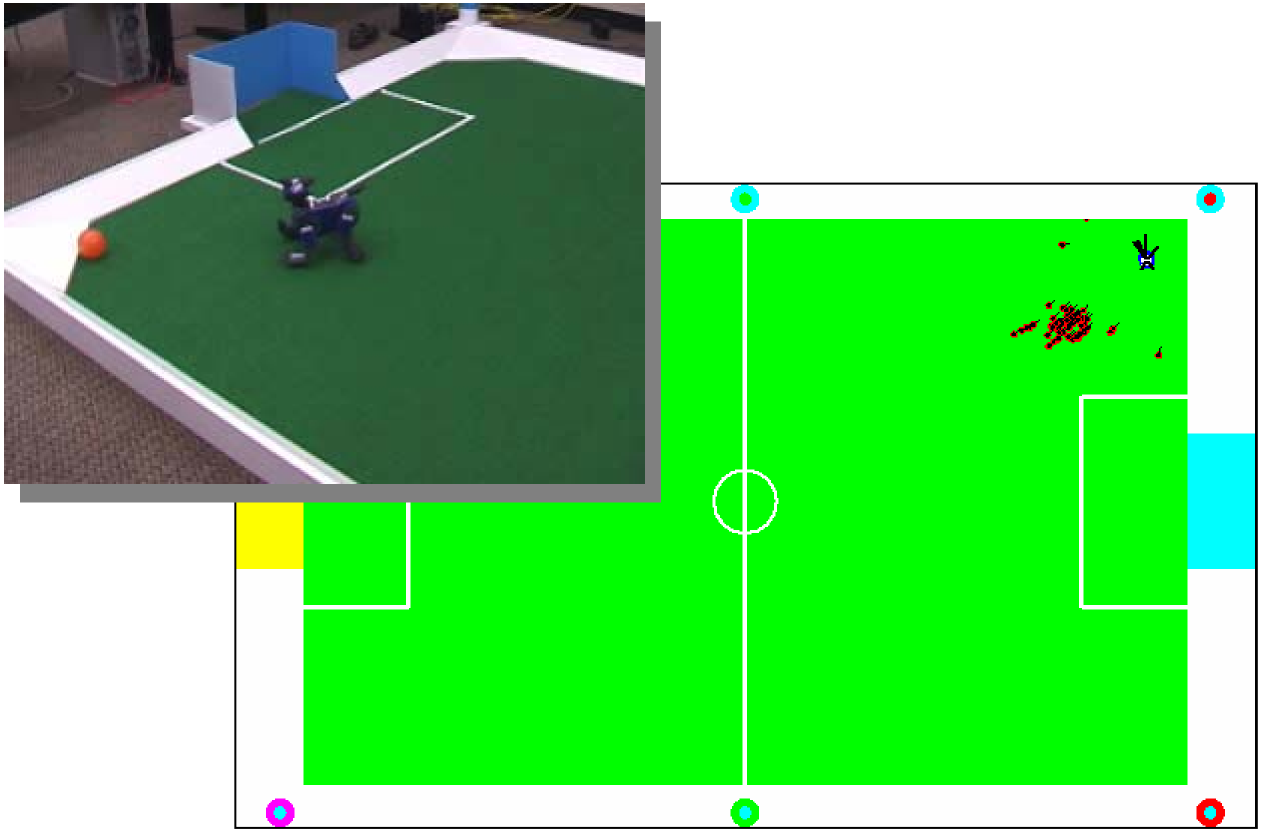
- Resample

Ball-Environment Interaction



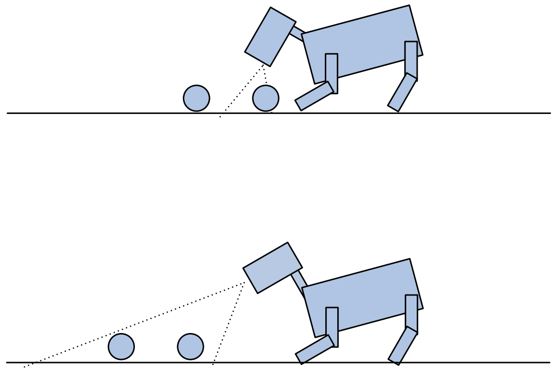


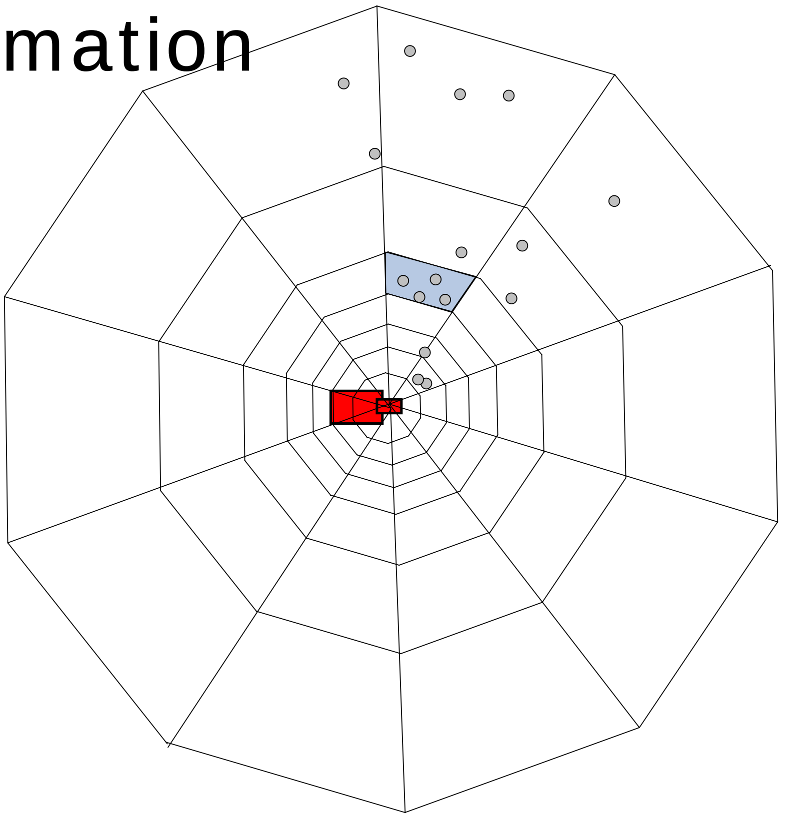
Ball-Environment Interaction



Tracking and Finding the Ball

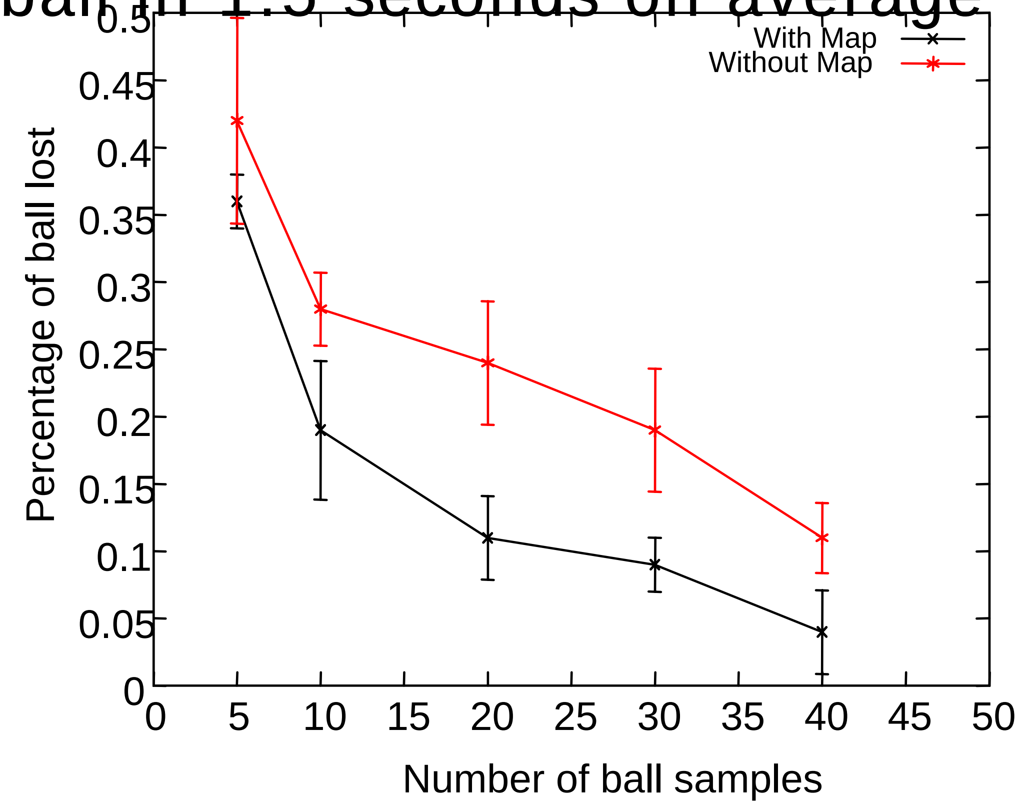
- Cluster ball samples by discretizing pan / tilt angles
- Uses negative information



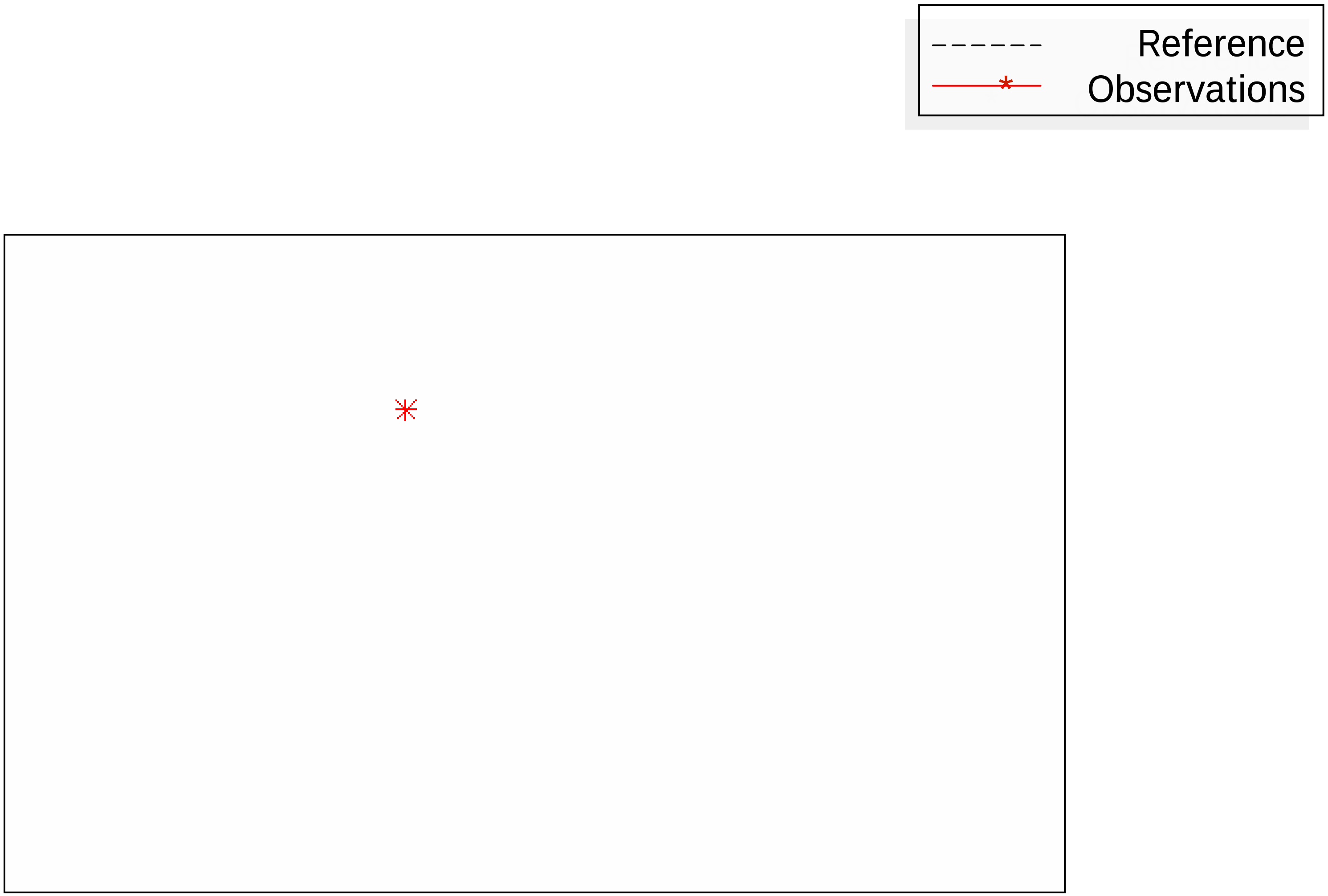


Experiment: Real Robot

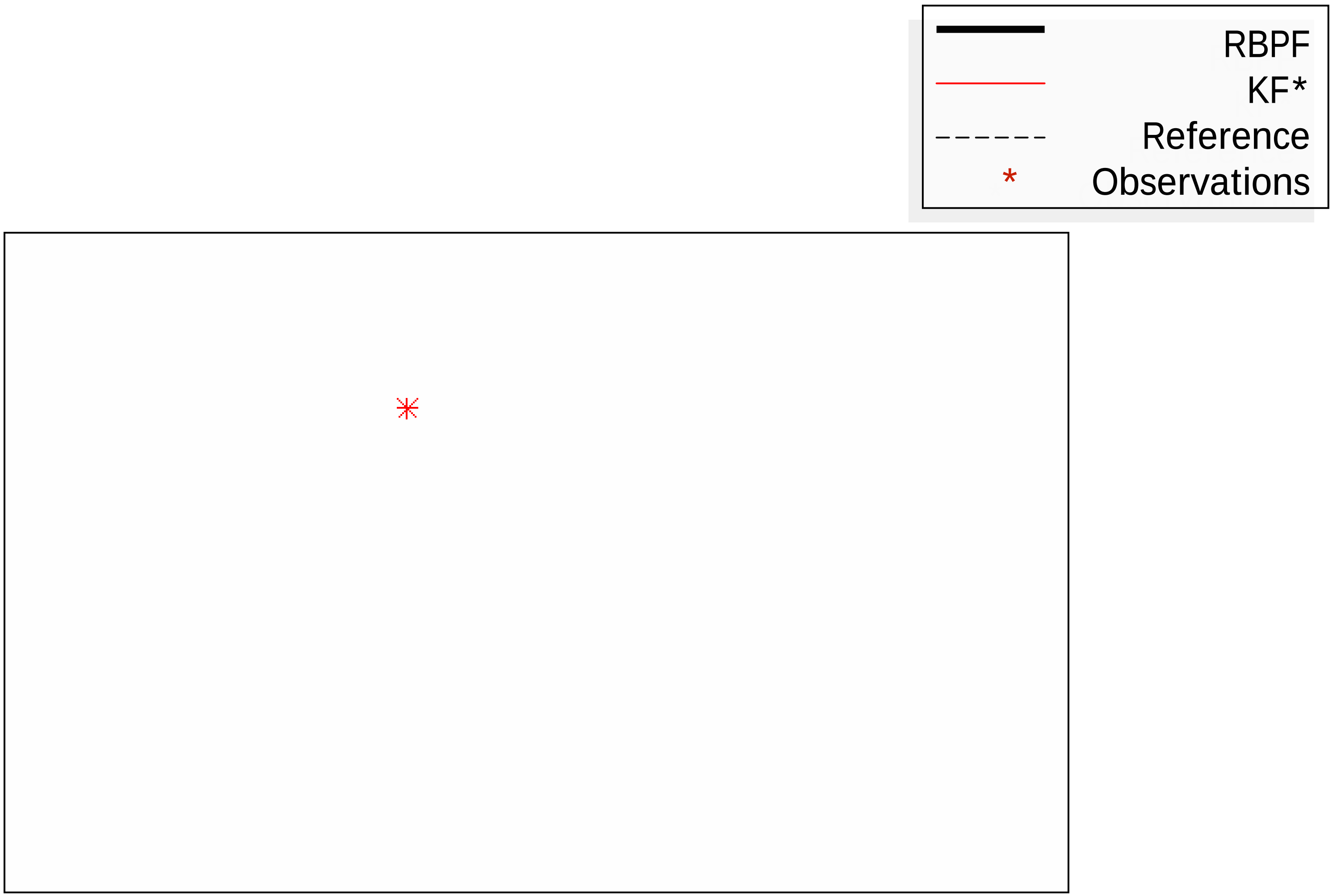
- Robot kicks ball 100 times, tries to find it afterwards
- Finds ball in 1.5 seconds on average



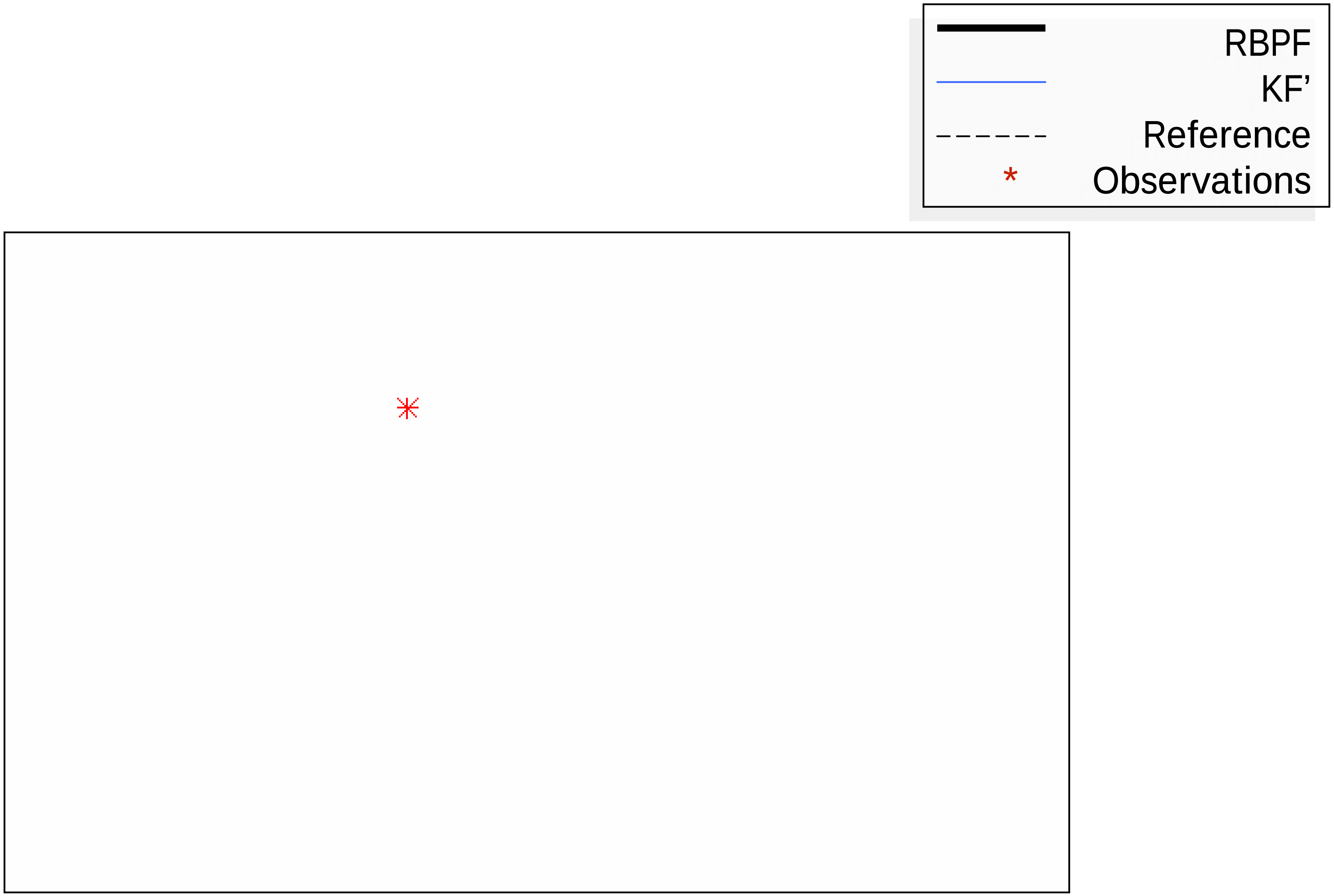
Simulation Runs



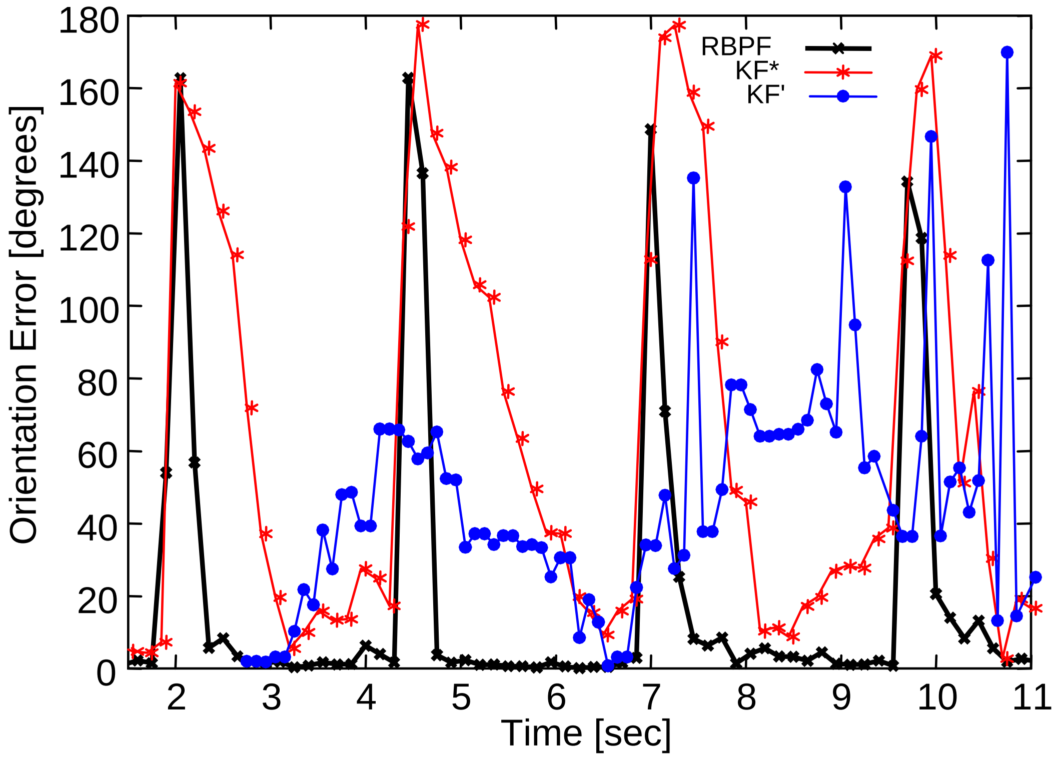
Comparison to KF* (optimized for straight motion)



Comparison to KF$'$ (inflated prediction noise)



Orientation Errors



Conclusions

- Bayesian filters are the most successful technique in robotics (vision?)
- Many instances (Kalman, particle, grid, MHT, RBPF, ...)
- Special case of dynamic Bayesian networks
- Recently: hierarchical models