Some Projects in Autonomous Robotics

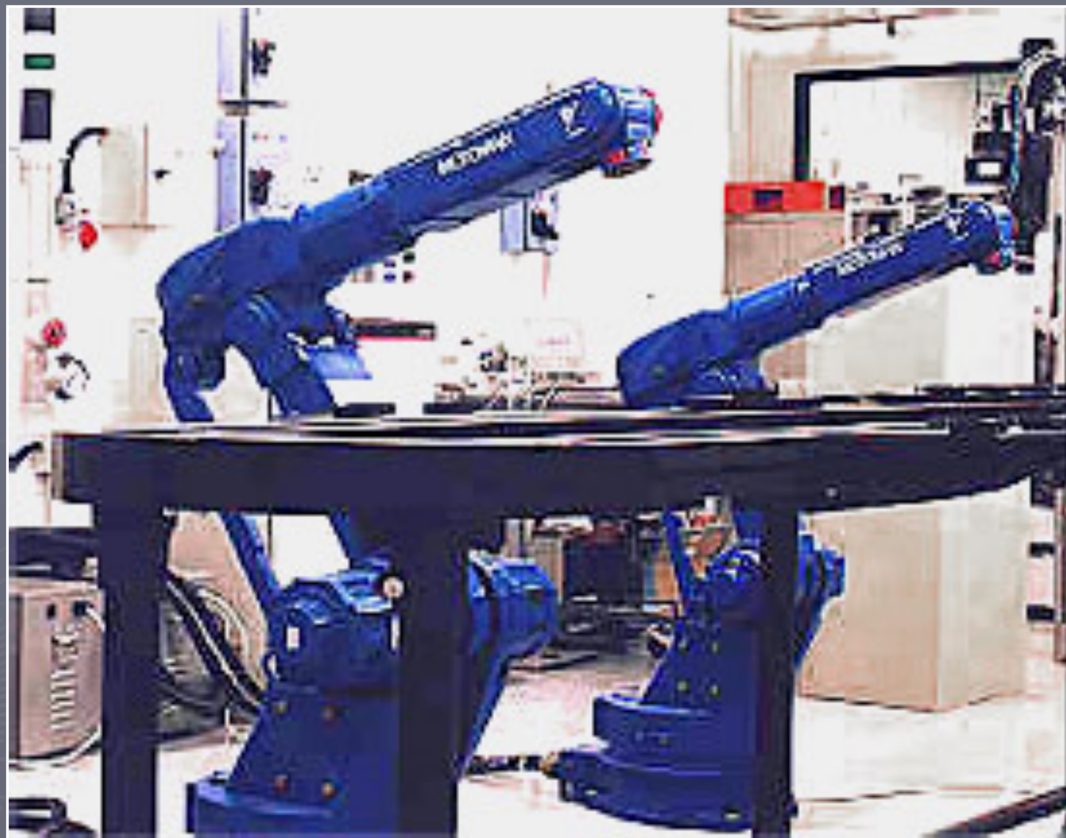
Dieter Fox

University of Washington
Department of Computer Science & Engineering

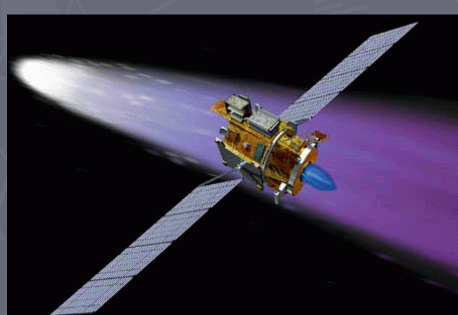
Robotics and State Estimation Lab
Intel Labs Seattle

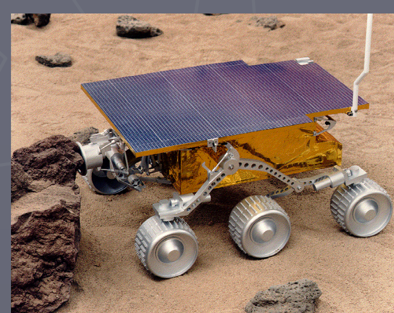
SA-1

Robotics Yesterday



Robotics Today

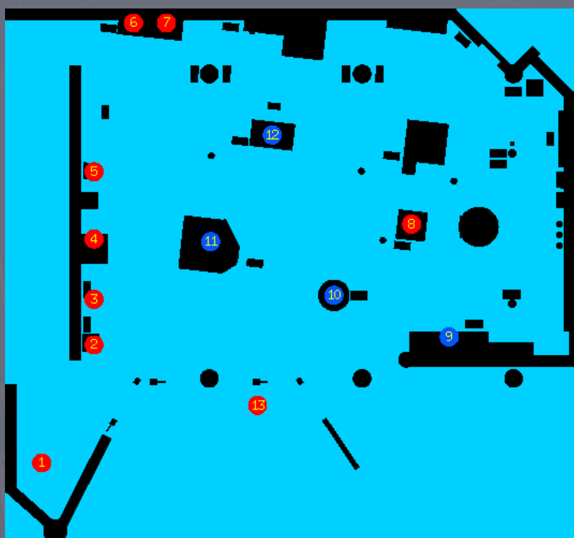




Robot Control

Sensor data

Control system



World model

Actions

Outline

- Overview
- **Localization and soccer playing**
- Exploration and map building
- Object recognition
- Discussion

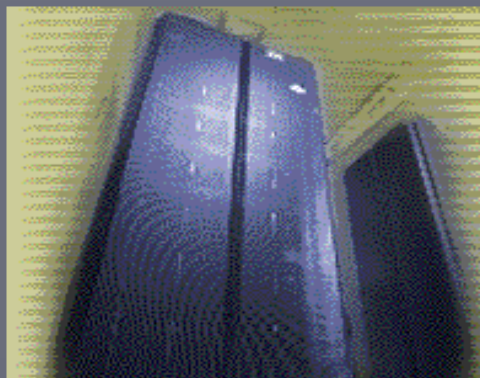
RoboCup Challenge

Design a team of robots that can play soccer!

- Dynamic, adversarial environments
- Real time control and decision making
- Multi-robot collaboration

RoboCup-99: Stockholm, Sweden Final

Challenges of RoboCup vs. Chess



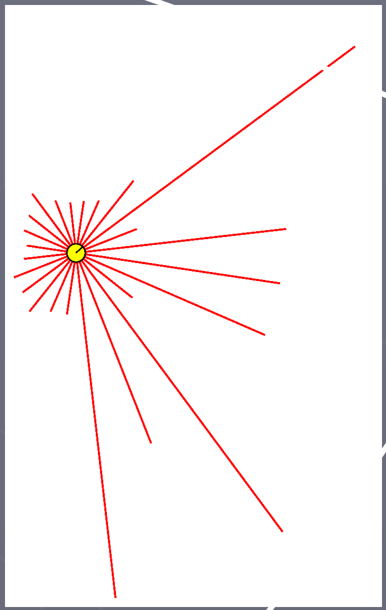
Deep Blue

Robot

- (Semi-) Static
- Deterministic
- Observable
- Turn-based

- Dynamic
- Stochastic
- Partially observable
- Real-time

Mobile Robot Localization



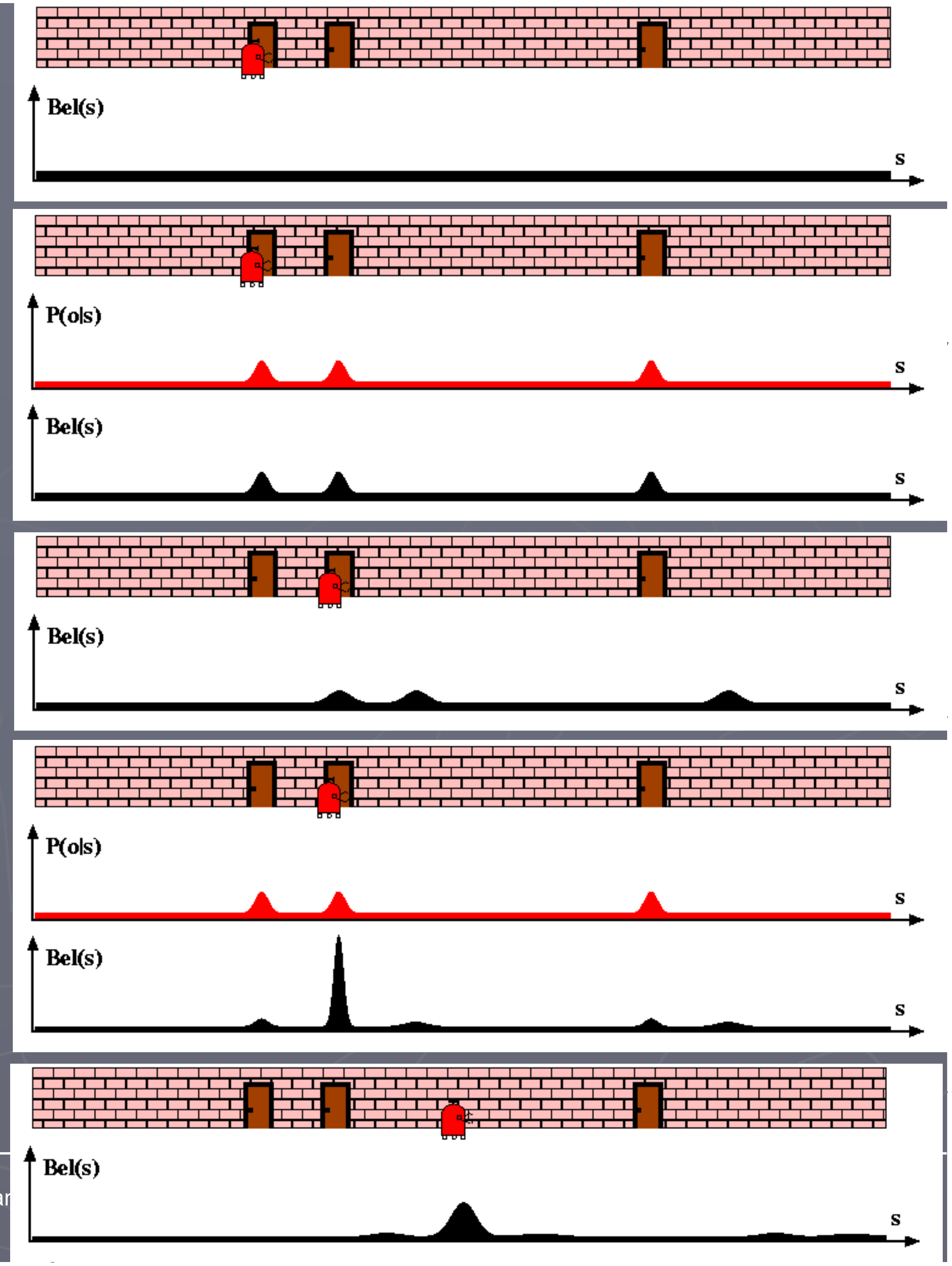
Where am I?

Bayes Filters for Robot Localization

- **Given:**
  - Stream of observations $z_{1:t}$ and control $u_{1:t}$
  - Sensor model $p(z_t \mid x_t)$
  - Action model $p(x_t \mid x_{t-1}, u_{t-1})$
  - Prior probability of the system state *p(x).*
- **Wanted:**
  - Estimate the state *x* of the dynamical system.
  - The posterior is estimated recursively:

$$p(x_t \mid z_{1:t}, u_{1:t-1}) = \eta \; p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1}) \; p(x_{t-1} \mid z_{1:t-1}, u_{1:t-2}) \; dx_{t-1}$$

Principle of Mobile Robot Localization

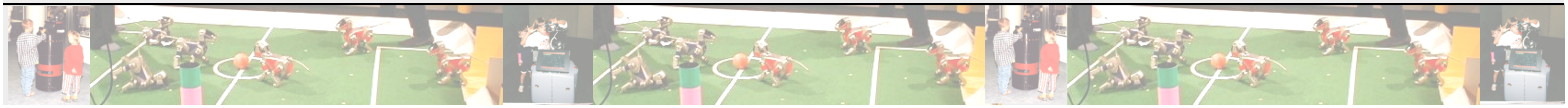


Particle Filters

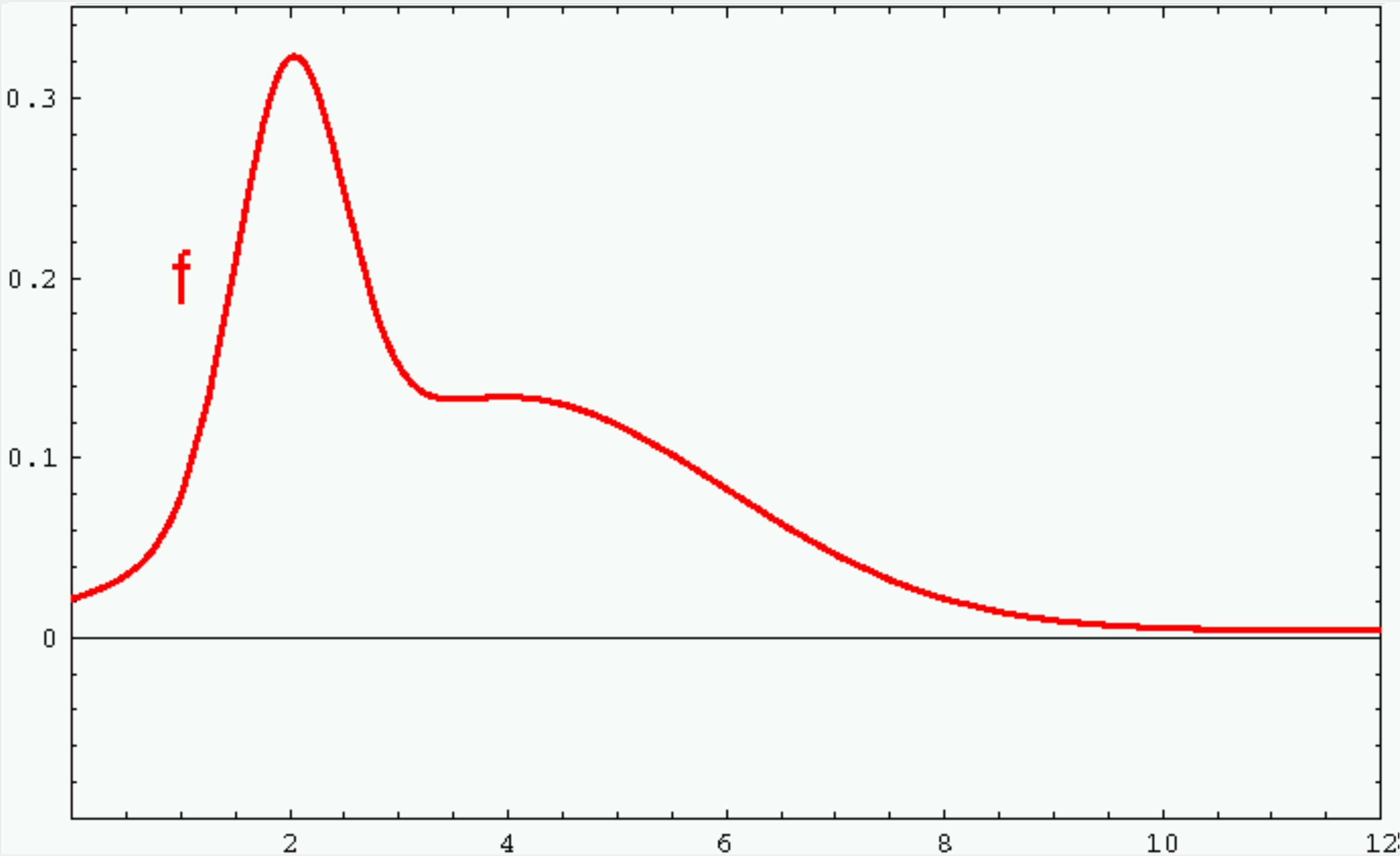
- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

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

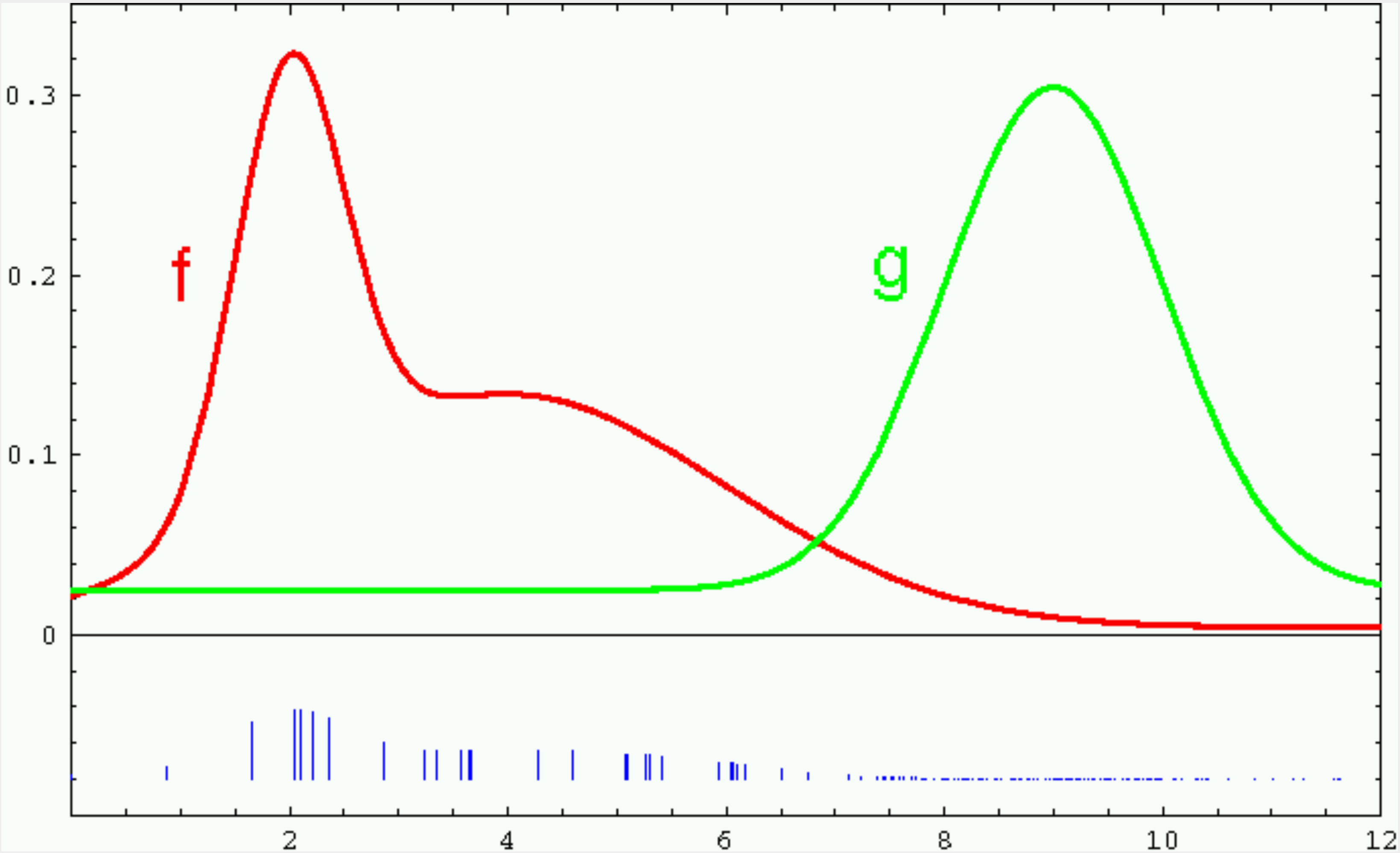
- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]d



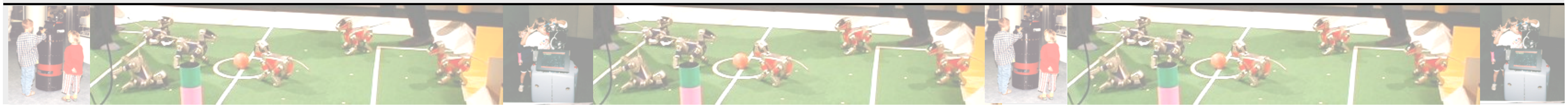
Sample-based Density Representation



Importance Sampling

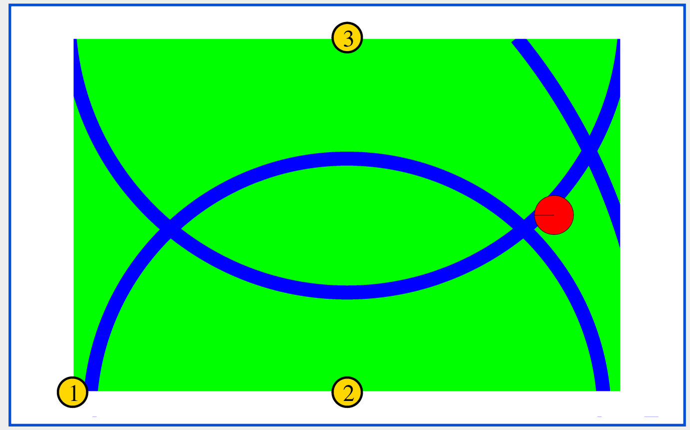


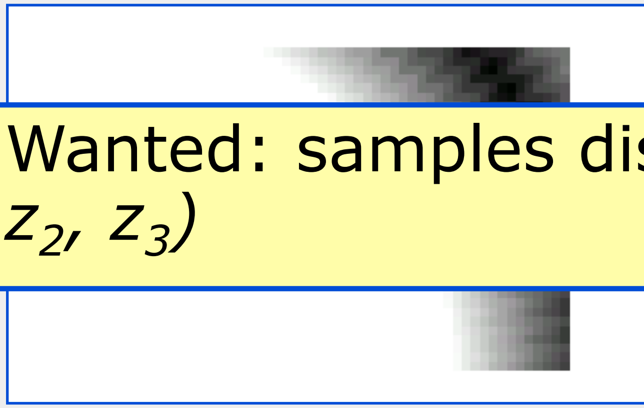
Weight samples: $w = f / g$

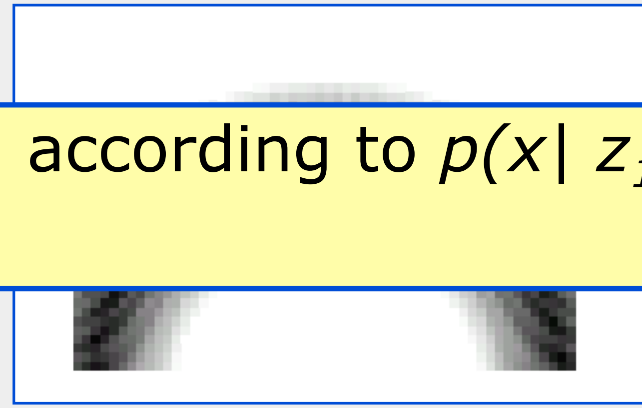


Importance Sampling with Resampling: Landmark Detection Example

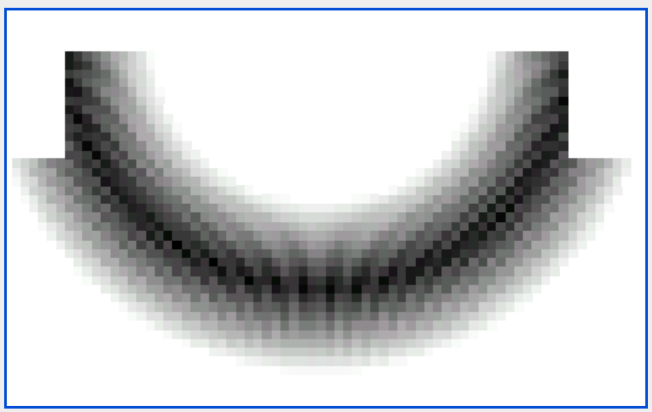
Distributions





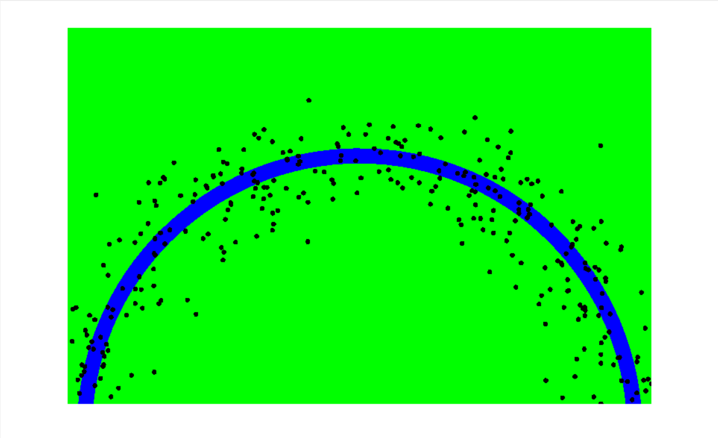


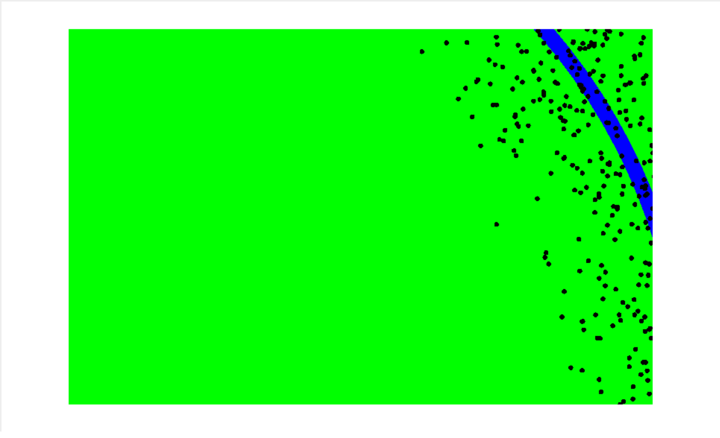
Wanted: samples distributed according to $p(x|\ z_1, z_2, z_3)$

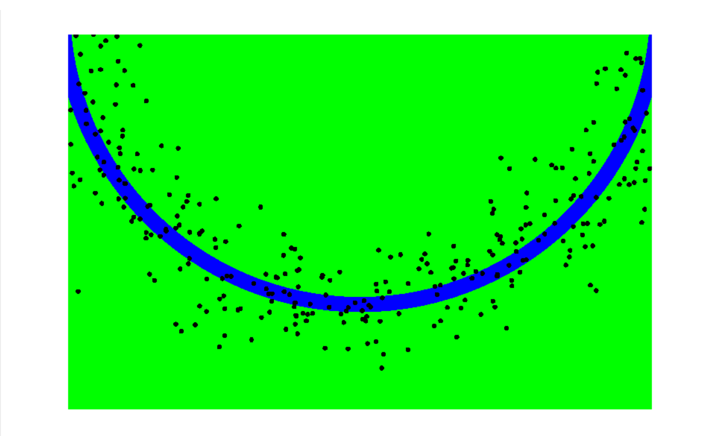


This is Easy!

We can draw samples from $p(x|z_l)$ by adding noise to the detection parameters.







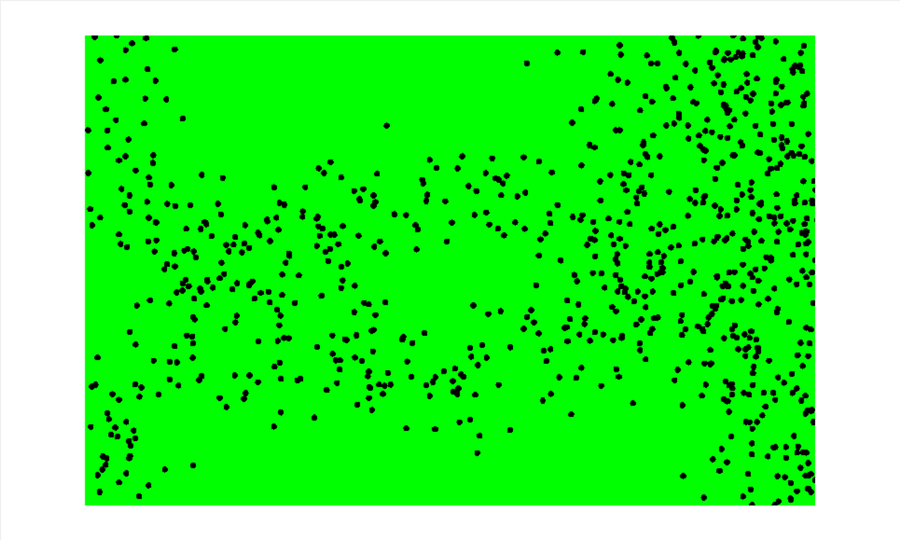
Importance Sampling with Resampling

$$\text{Target distribution f}: p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)}$$

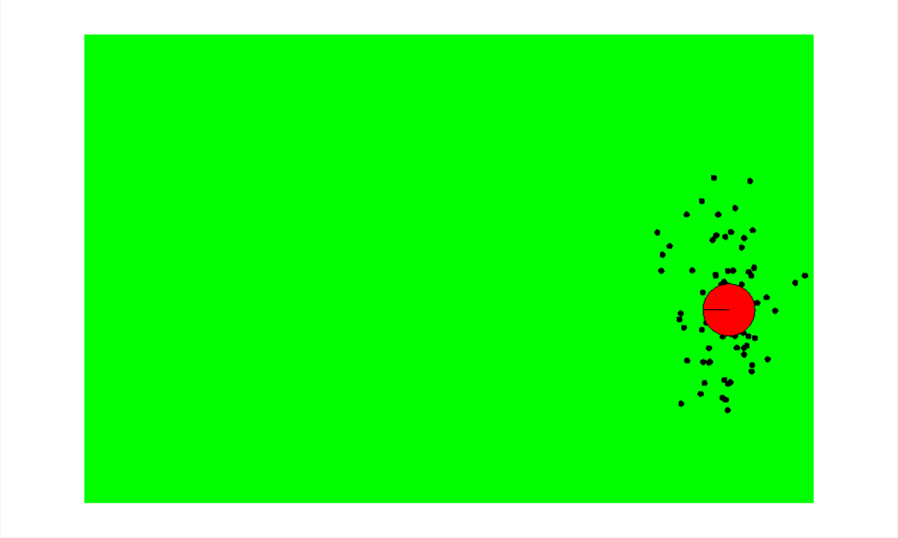
$$\text{Sampling distribution g}: p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)}$$

$$\text{Importance weights w}: \frac{f}{g} = \frac{p(x \mid z_1, z_2, \ldots, z_n)}{p(x \mid z_l)} = \frac{p(z_l) \prod_{k \neq l} p(z_k \mid x)}{p(z_1, z_2, \ldots, z_n)}$$

Importance Sampling with Resampling

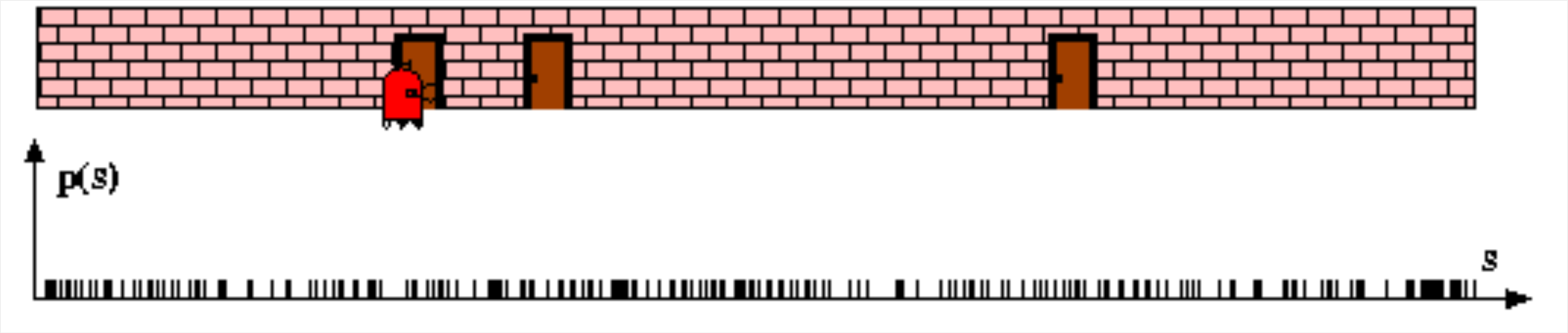


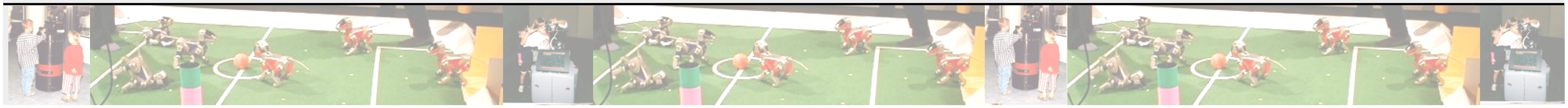
Weighted samples



After resampling

Particle Filters

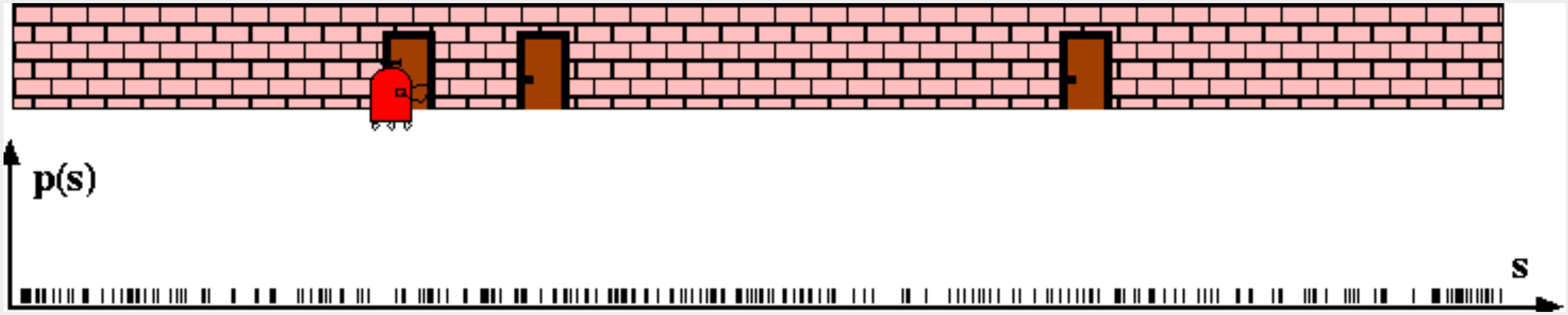


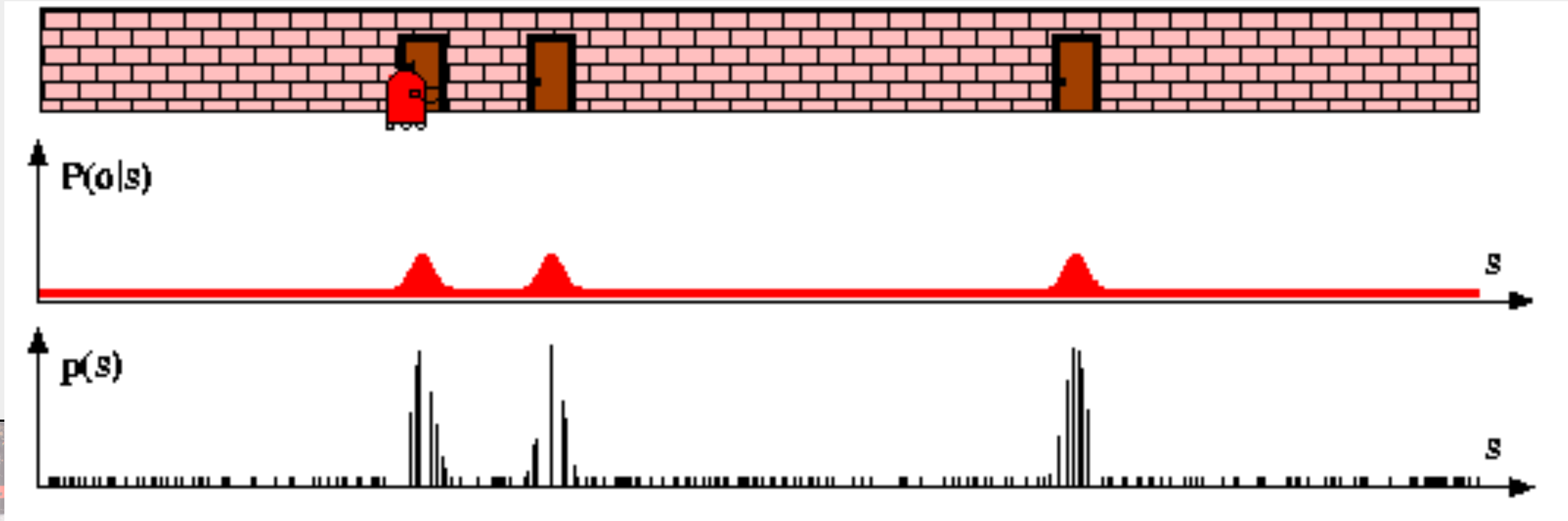


Sensor Information: Importance Sampling

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

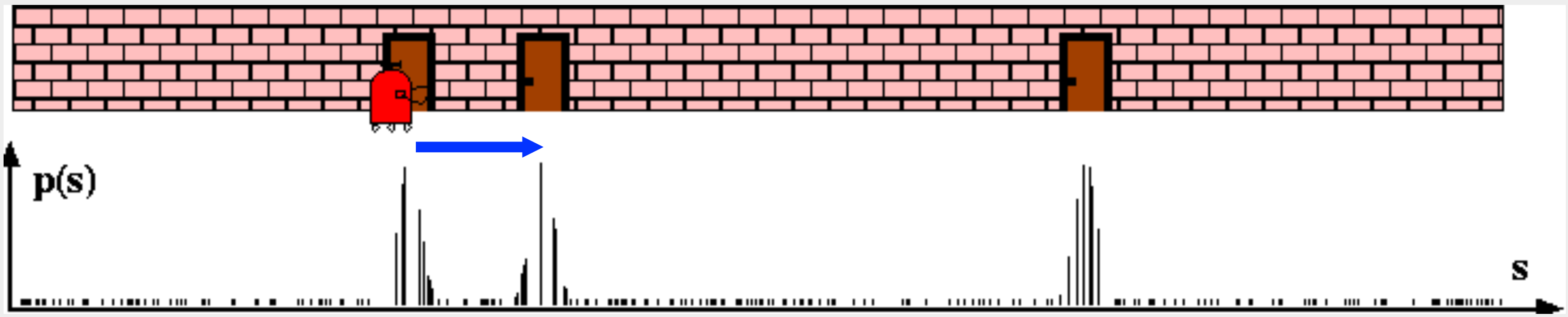
$$w \leftarrow \frac{\alpha\ p(z\,|\,x)\ Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z\,|\,x)$$

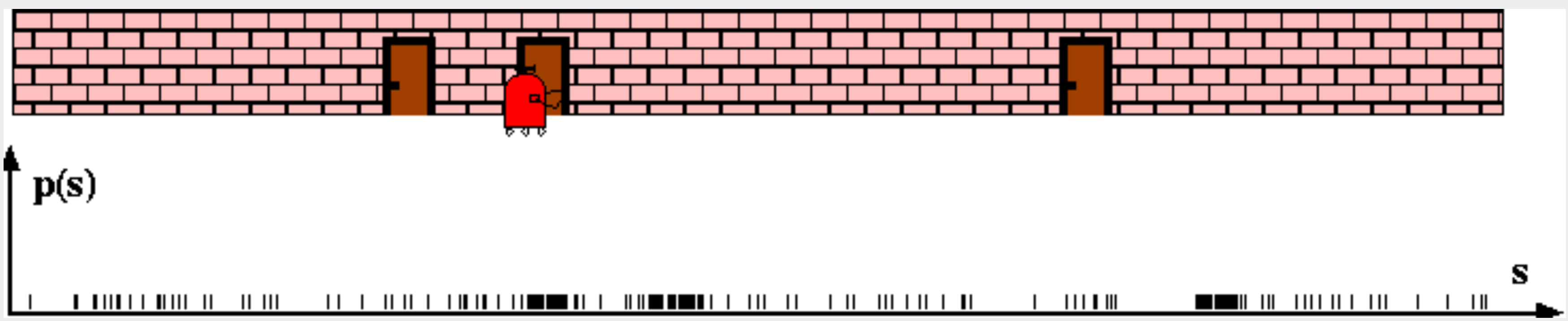


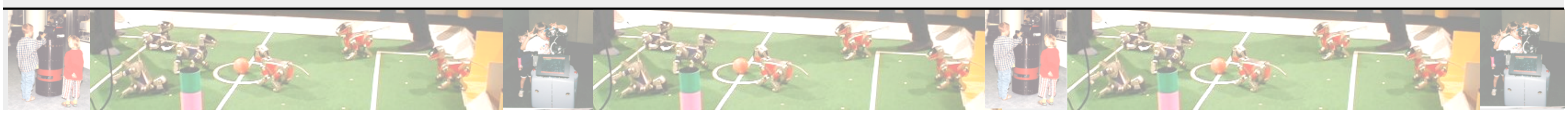


Robot Motion

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



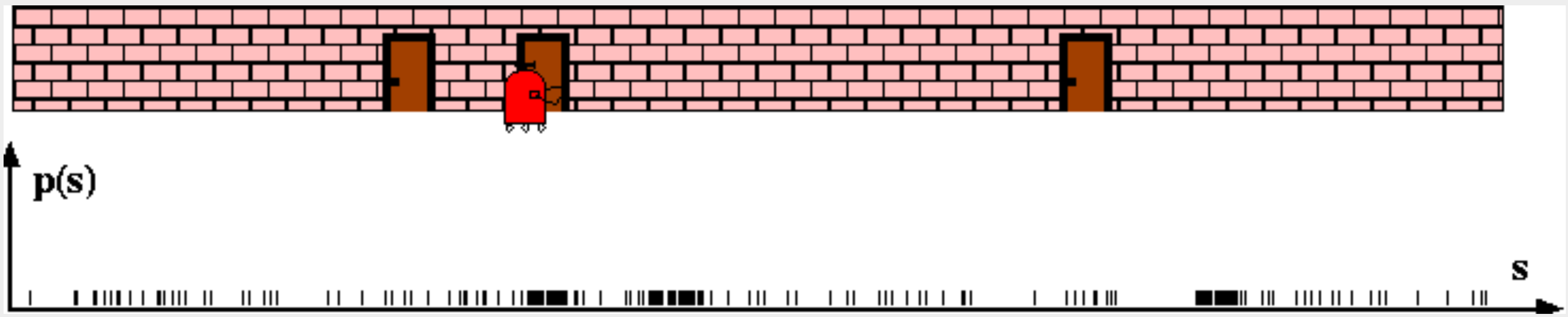


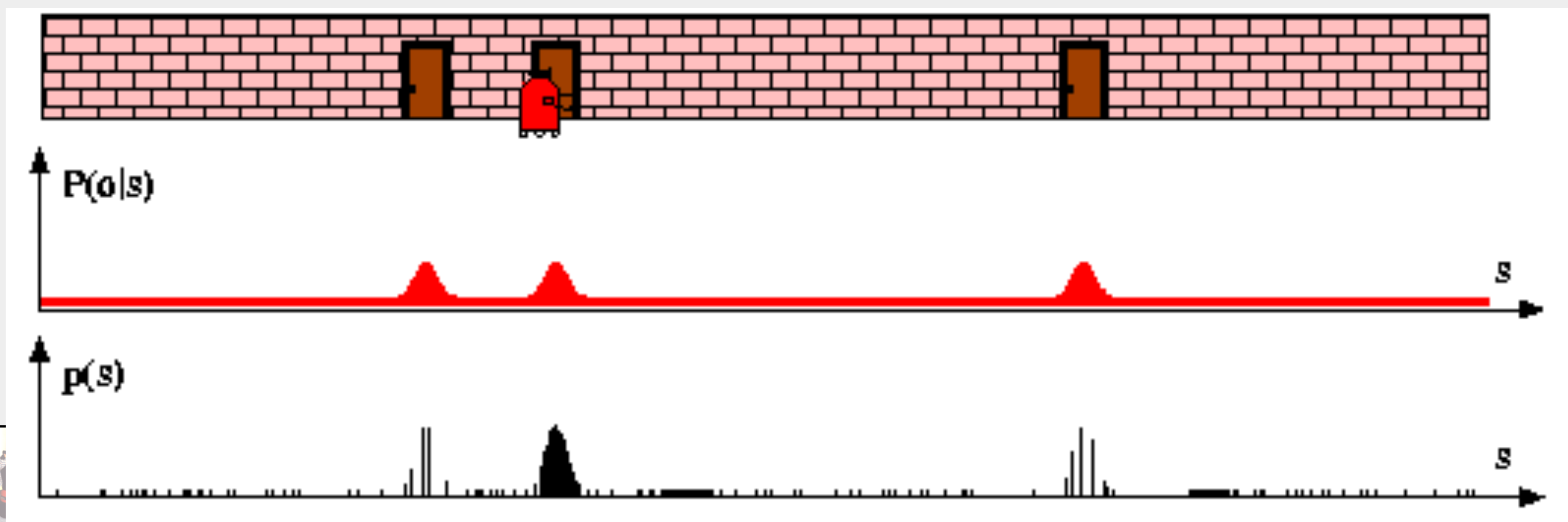


Sensor Information: Importance Sampling

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

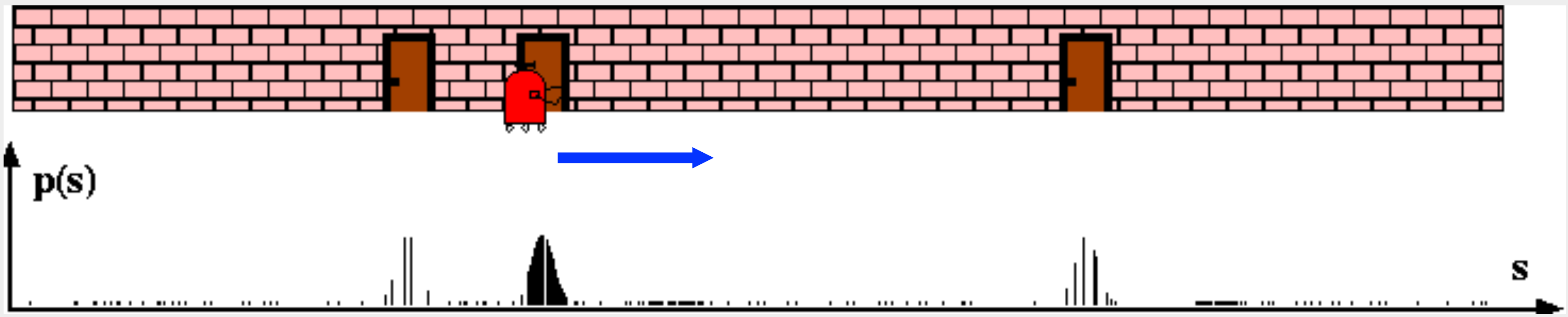
$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$

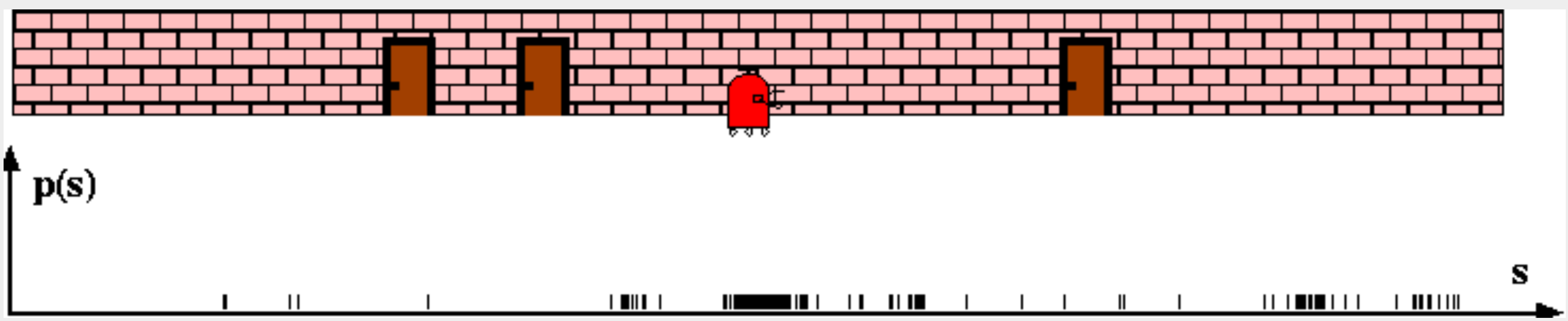


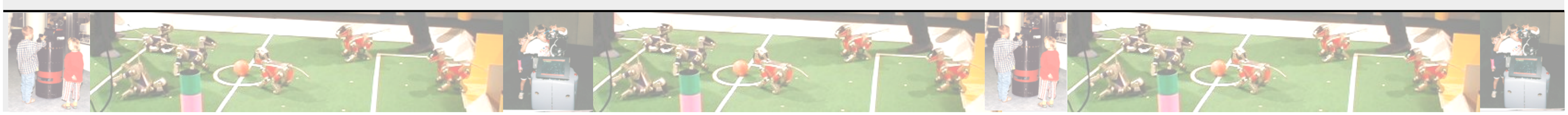


Robot Motion

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







Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Generate new samples***
4. Sample index *j(i)* from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weights***

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:

$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta\; p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel\;(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel\;(x_{t-1})} \propto p(z_t \mid x_t)$$

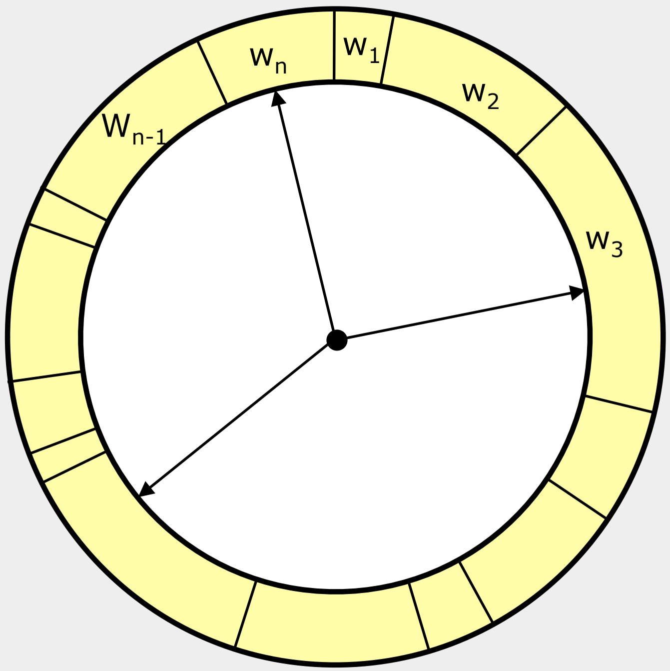
Resampling

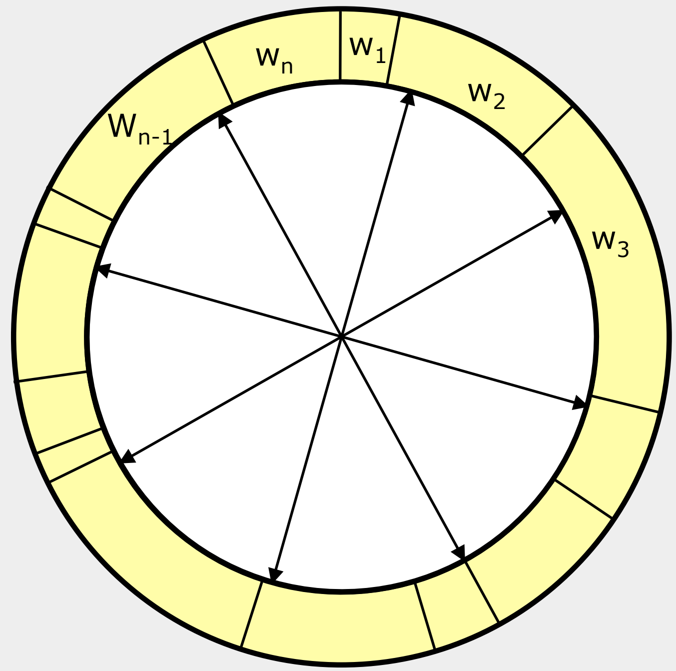
- **Given**: Set *S* of weighted samples.

- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.

- Typically done *n* times with replacement to generate new sample set *S'*.

Resampling





- Roulette wheel
- Binary search, log n

- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Resampling Algorithm

1. Algorithm **systematic_resampling**(*S*,*n*):

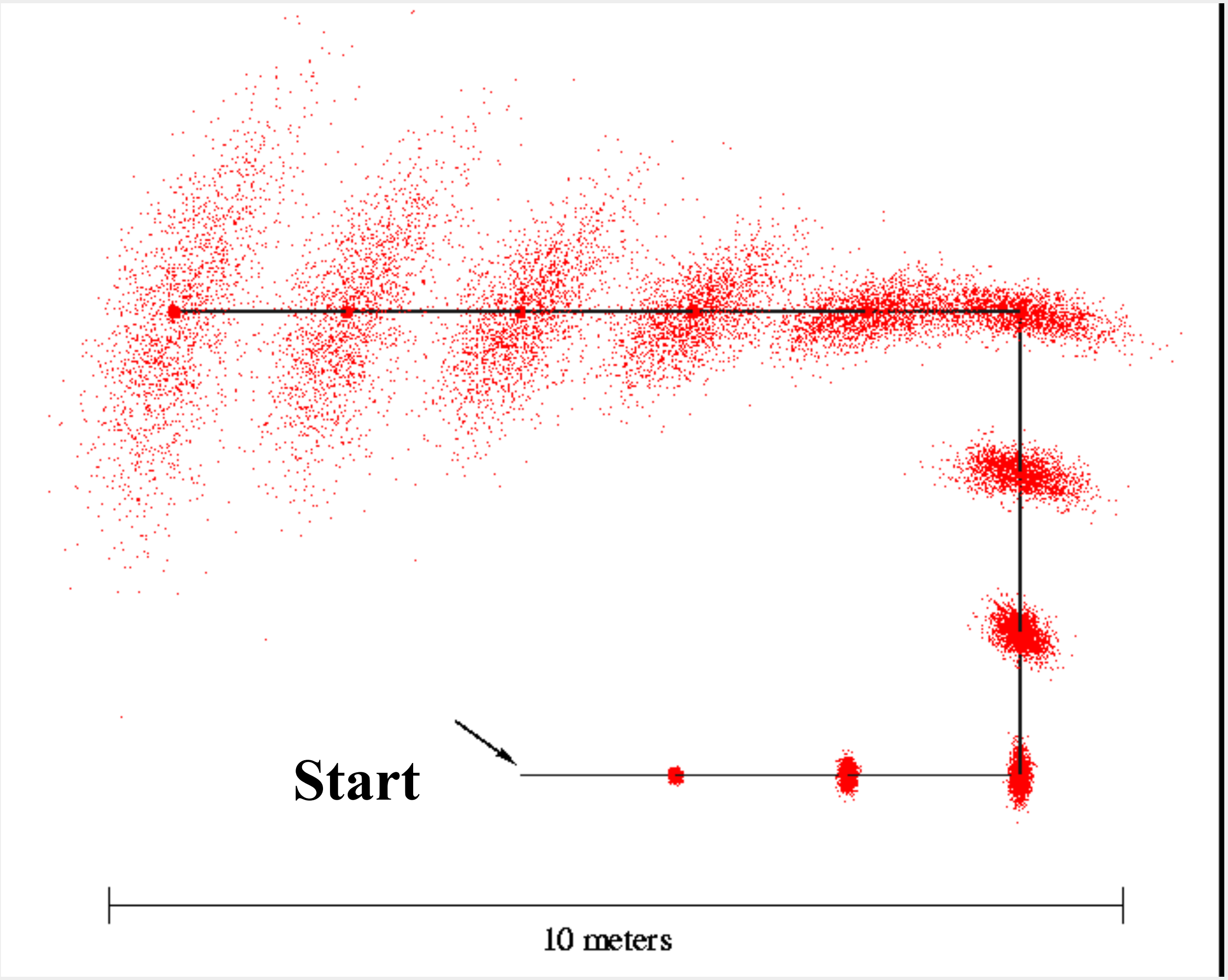
2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ — ***Generate cdf***
4. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ — ***Initialize threshold***

6. **For** $j = 1 \ldots n$ — ***Draw samples …***
7. $\quad$ **While** ($u_j > c_i$) — ***Skip until next threshold reached***
8. $\quad\quad i = i + 1$
9. $\quad S' = S' \cup \{< x^i, n^{-1} >\}$ — ***Insert***
10. $\quad u_j = u_j + n^{-1}$ — ***Increment threshold***

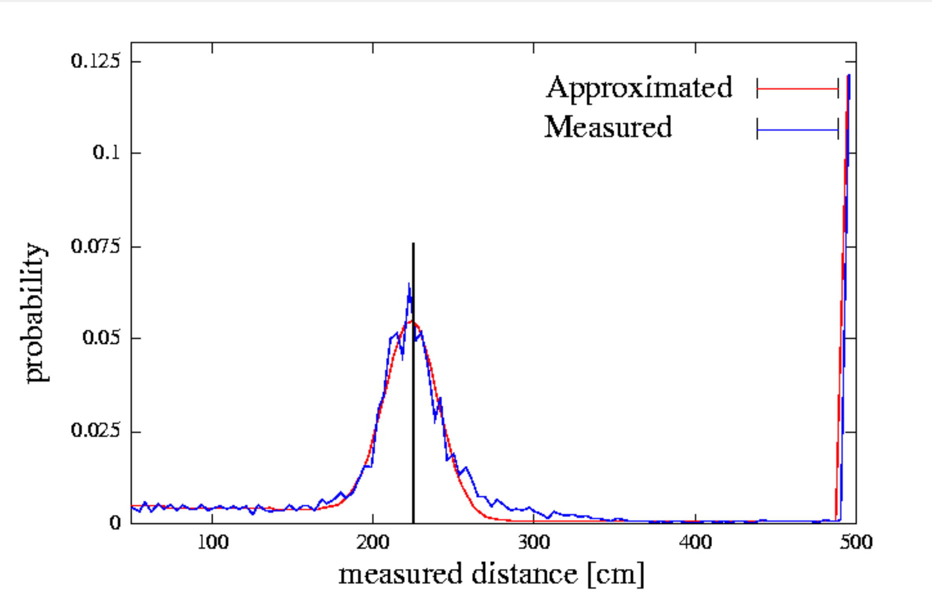
11. **Return** *S'*

Also called **stochastic universal sampling**

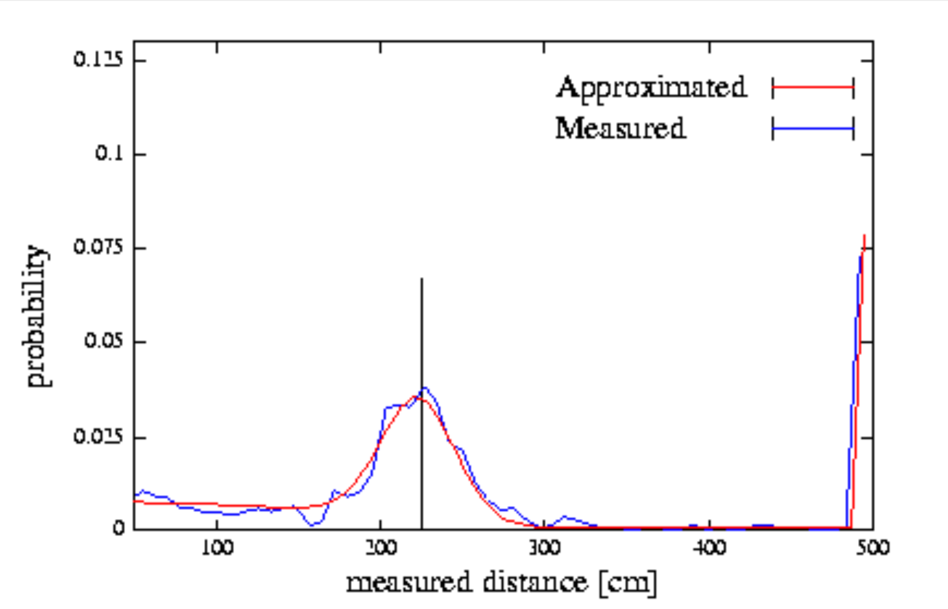
Motion Model Reminder



Proximity Sensor Model Reminder

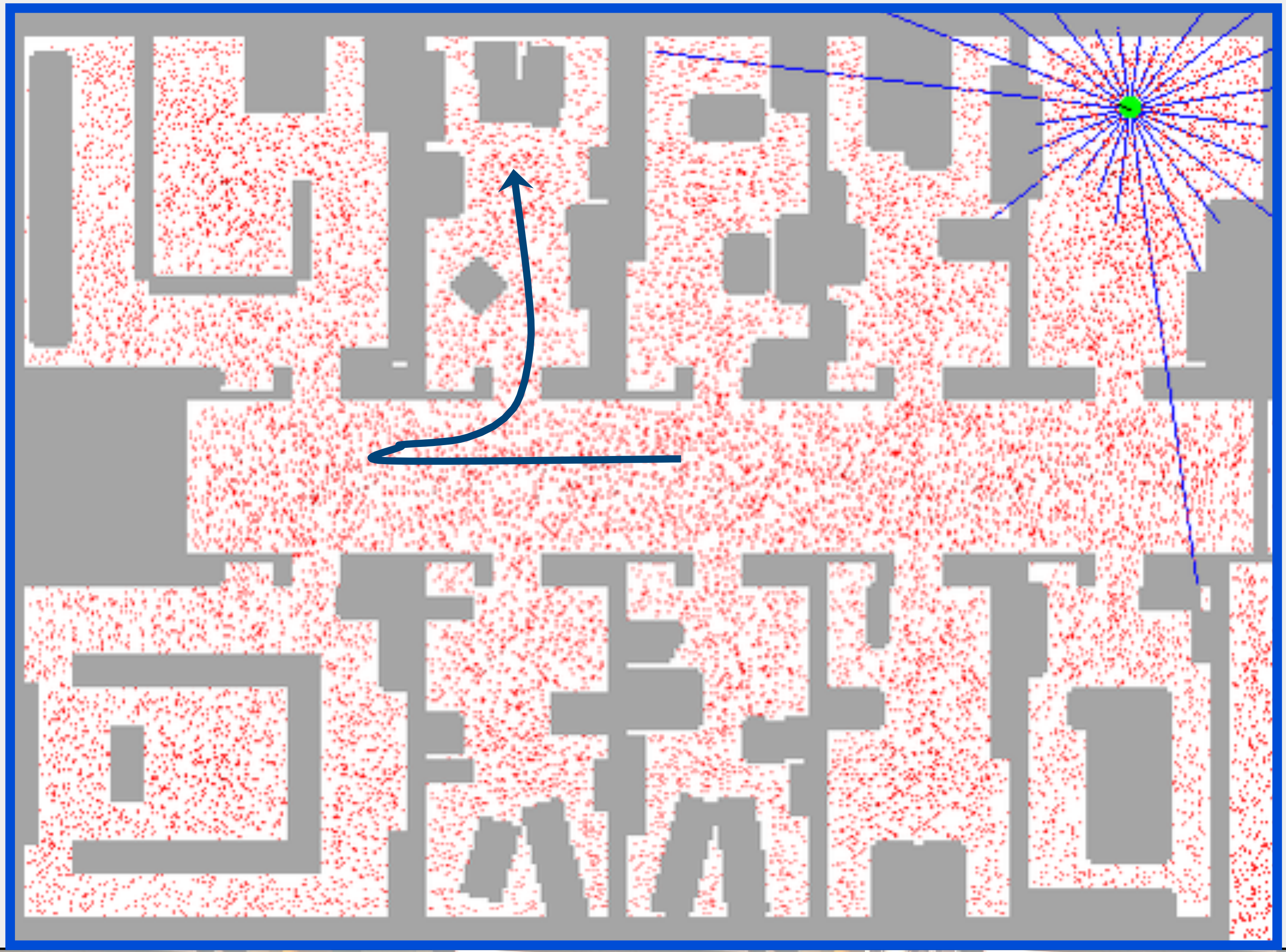


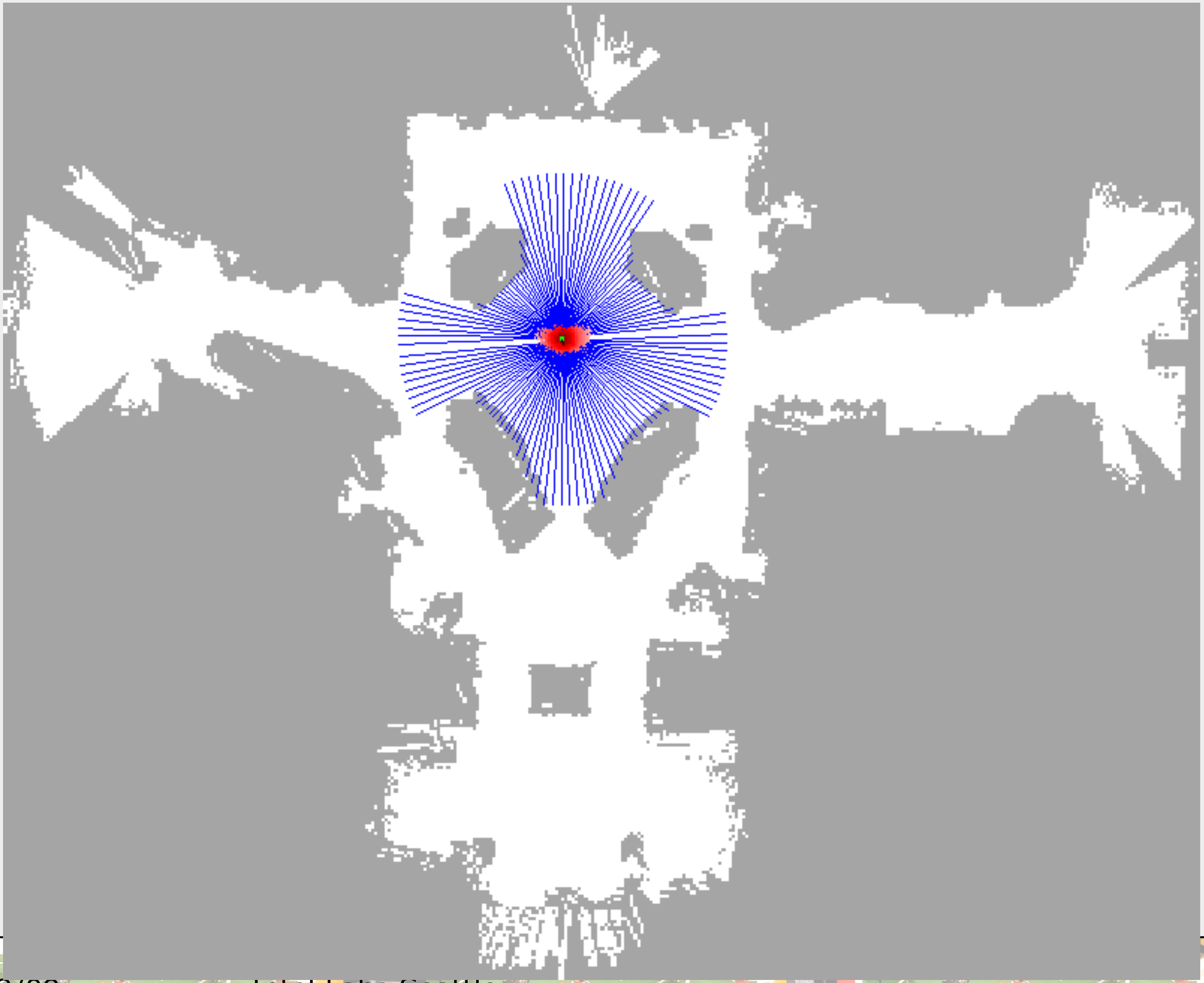
Laser sensor



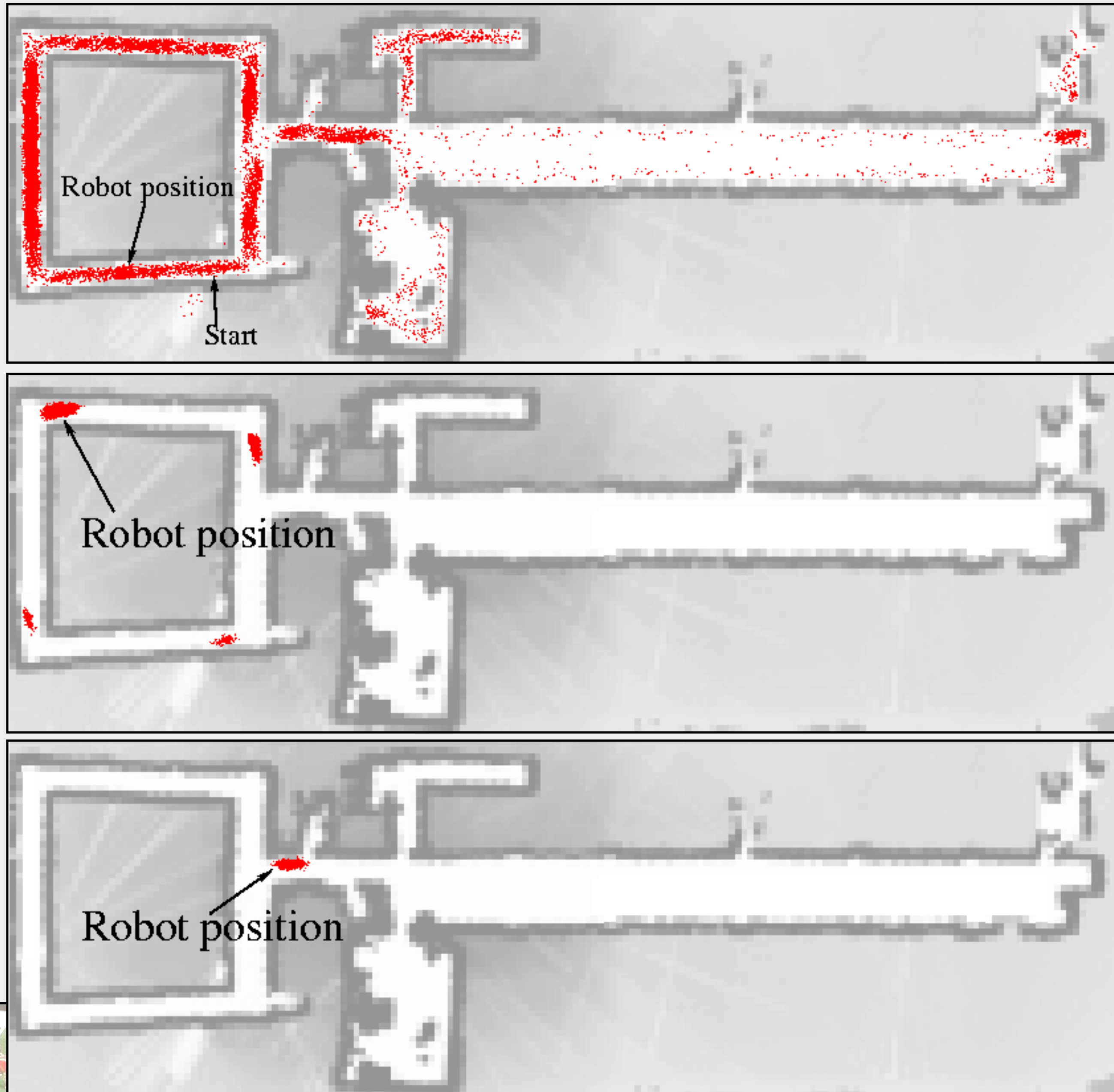
Sonar sensor

Sample-based Localization (sonar)





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- d), the KL-distance between the true posterior and the sample-based approximation is less than e.

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

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

Adaptive Particle Filter Algorithm

Algorithm **adaptive_particle_filter**(S_{t-1}, u_{t-1} z_t, $\Delta, \varepsilon, \delta$):

$S_t = \emptyset, \quad \alpha = 0, \quad n = 0, \quad k = 0, \quad b = \emptyset$

Do ***Generate new samples***

    Sample index *j(n)* from the discrete distribution given by w_{t-1}

    Sample x_t^n from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(n)}$ and u_{t-1}

    $w_t^n = p(z_t \mid x_t^n)$ ***Compute importance weight***

    $\eta = \eta + w_t^n$ ***Update normalization factor***

    $S_t = S_t \cup \{< x_t^n, w_t^n >\}$ ***Insert***

    If (x_t^n falls into an empty bin *b*) ***Update bins with support***

        k=k+1, b = non-empty

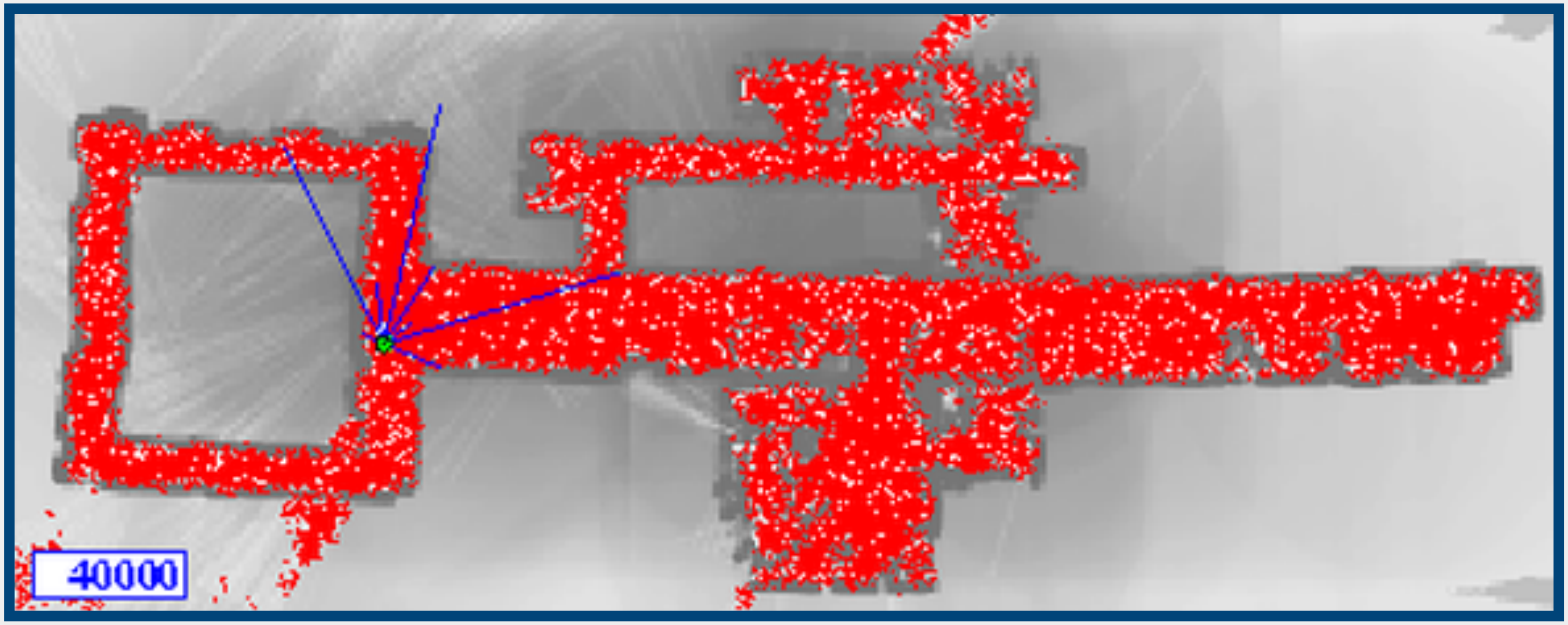
    n=n+1

While ($n < \frac{1}{2\varepsilon} \mathrm{X}^2(k-1, 1-\delta)$)

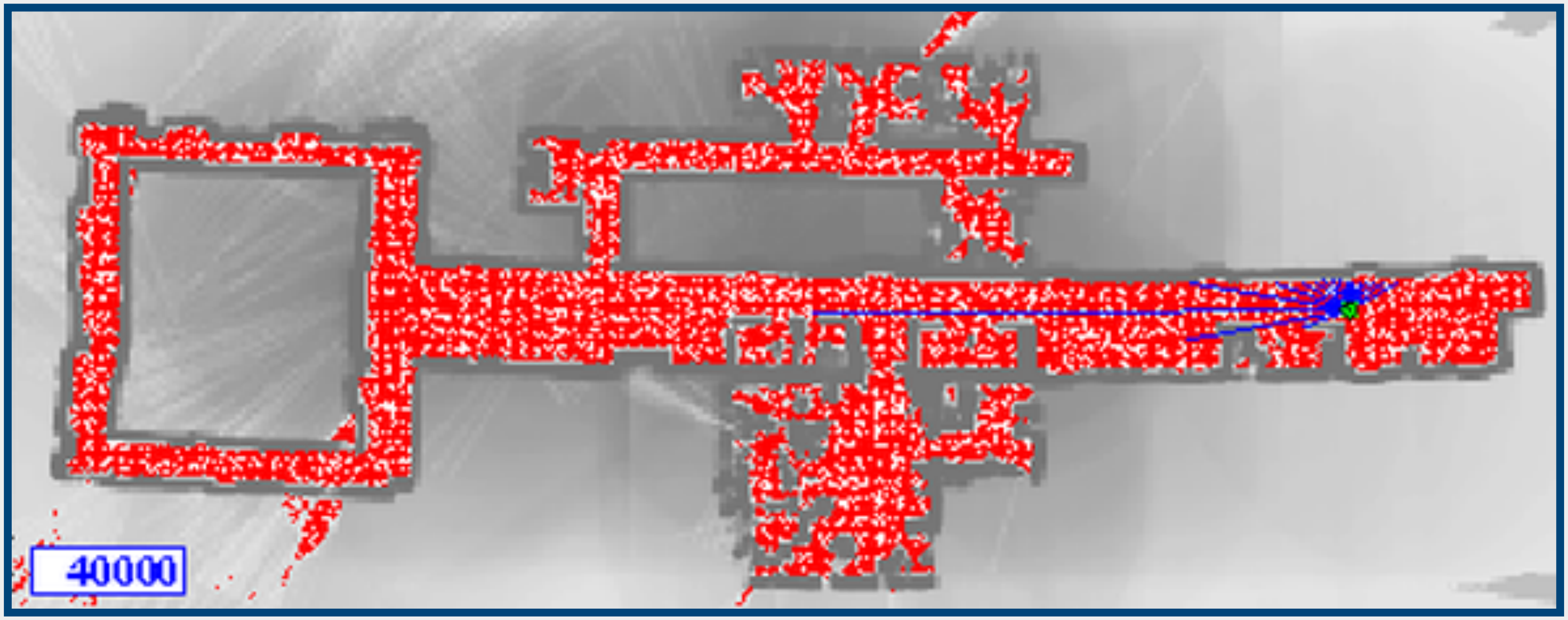
For $i = 1 \ldots n$

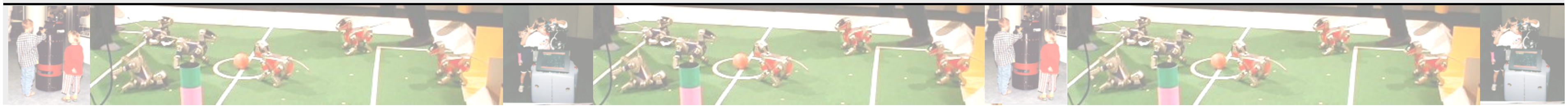
    $w_t^i = w_t^i / \eta$ ***Normalize weights***

Example Run Sonar

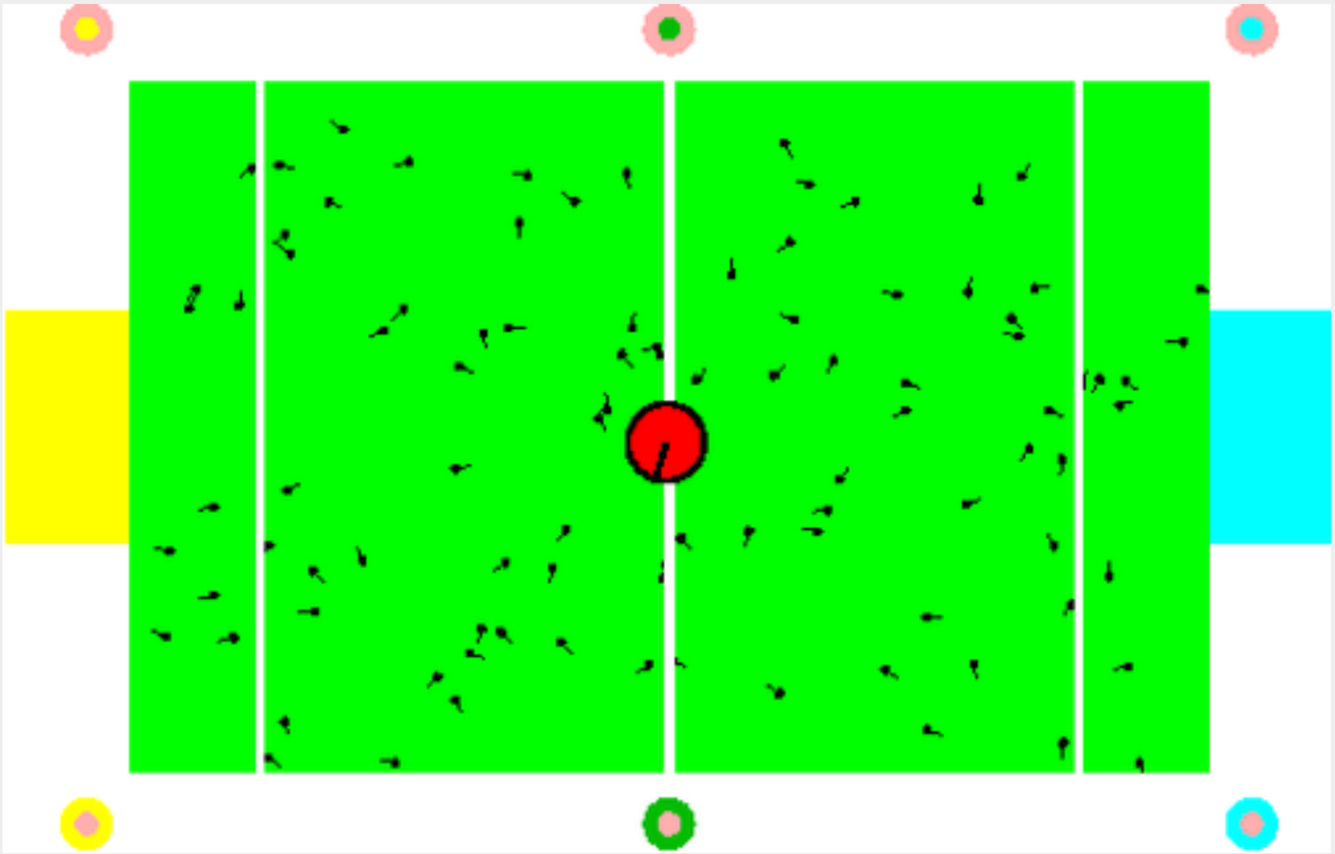


Example Run Laser

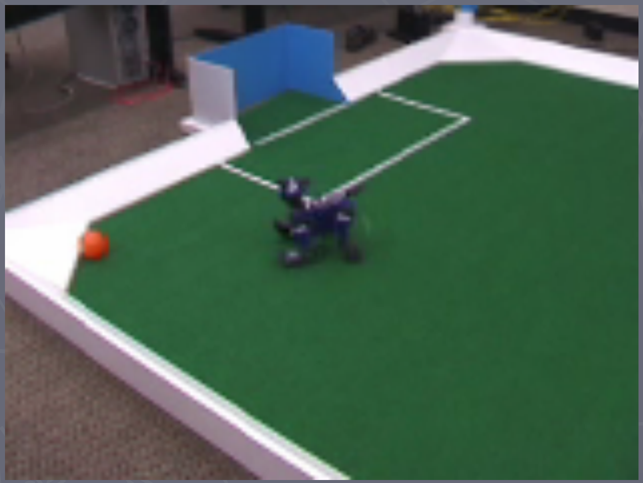


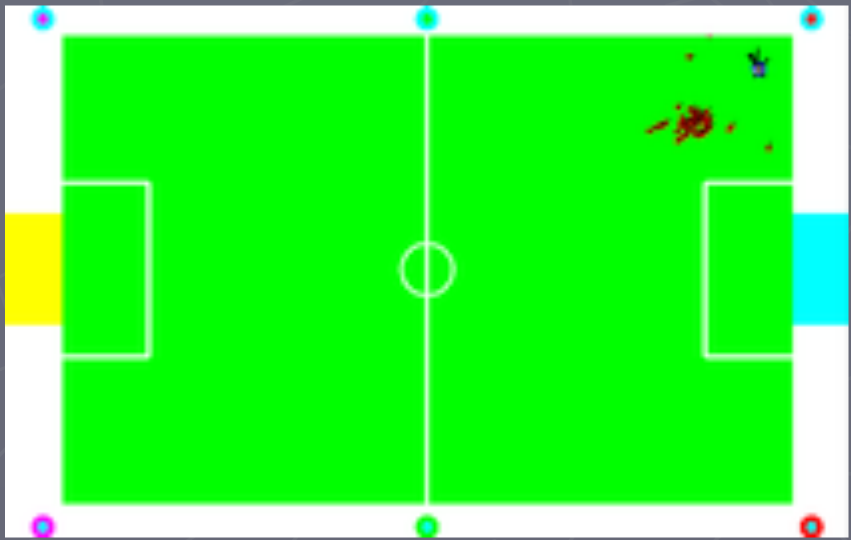


Localization for AIBO robots

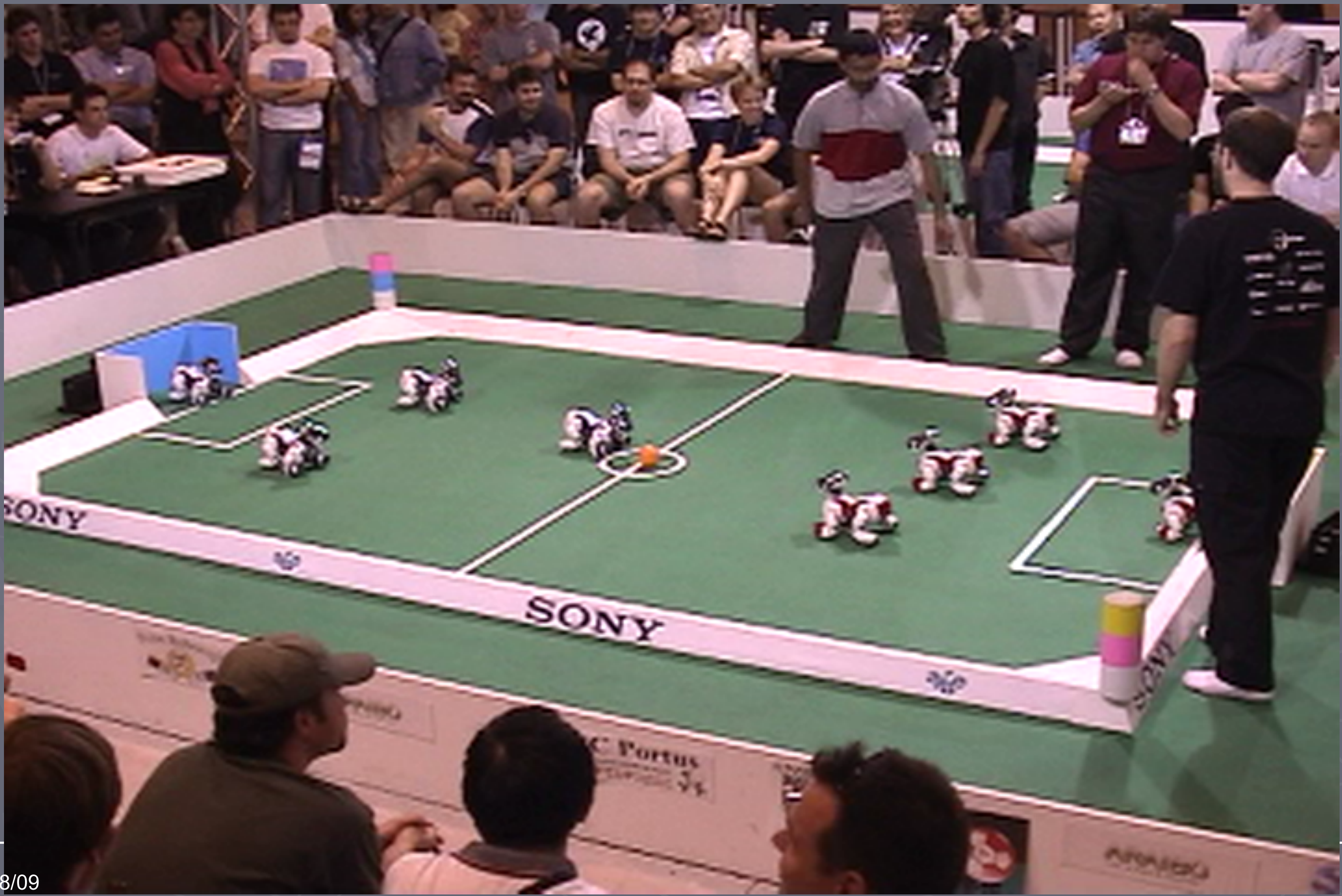


Ball Tracking





RoboCup 2004



Outline

- Overview
- Playing soccer with robots
- Exploration and map building
- Object recognition
- Discussion

Mapping the Allen Center: Raw Data

Mapping the Allen Center

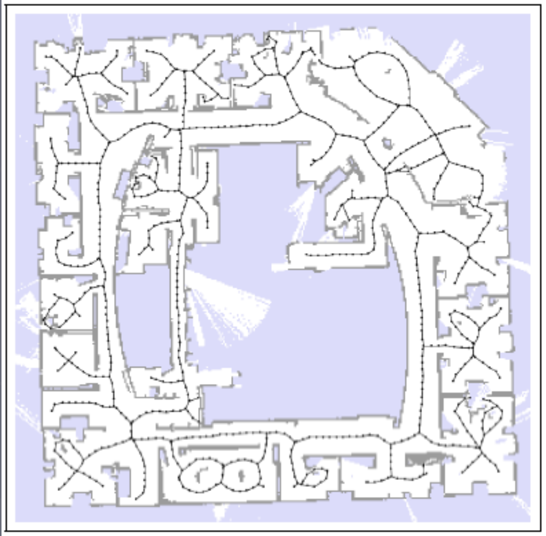
$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \ldots dx_{t-1}$$

Coordinated exploration with three robots
from unknown start locations

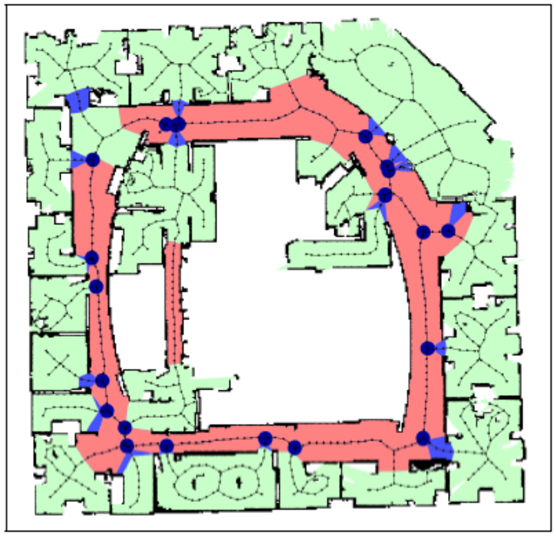
The robots are fully autonomous.
All computation is performed on–board.

Shown is the perspective of one robot

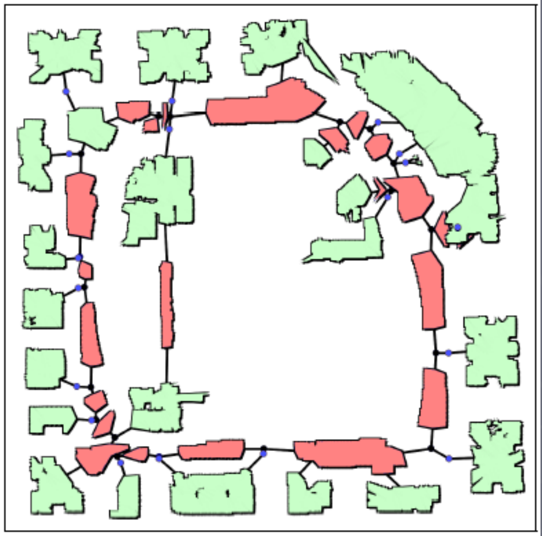
Semantic Mapping



Occupancy map



Spatial Labeling



Topological Representation

- Learn parameters from labelled maps, apply to new one
- Accuracy: 91.2%

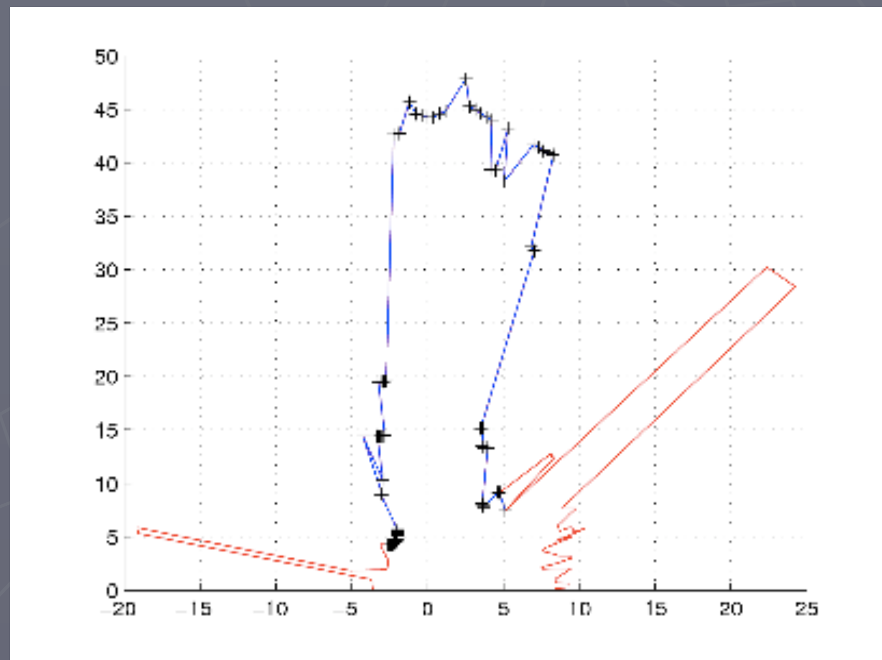
$$p(\mathbf{x} \mid \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp\left\{ \sum_{c \in C} \mathbf{w}_c^{\mathrm{T}} \mathrm{f}_c(\mathbf{x}_c, \mathbf{z}_c) \right\}$$

Outline

- ► Overview
- ► Playing soccer with robots
- ► Exploration and map building
- ► Object recognition
- ► Discussion

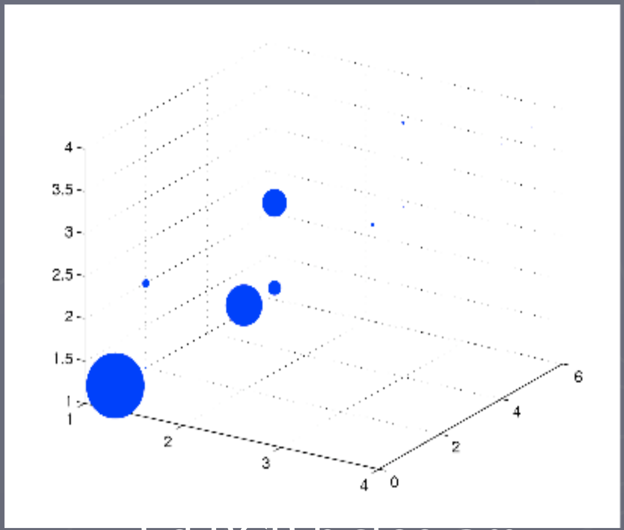
Data

Geometric Features



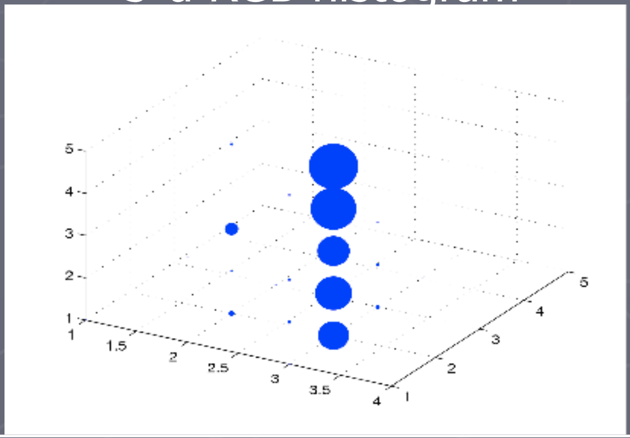
$$f_{geometric} = \begin{bmatrix} \text{distances to neighbors} \\ \text{angles to neighbors} \\ \text{\#max range neighbors} \end{bmatrix}$$

Visual Features



3-d RGB histogram

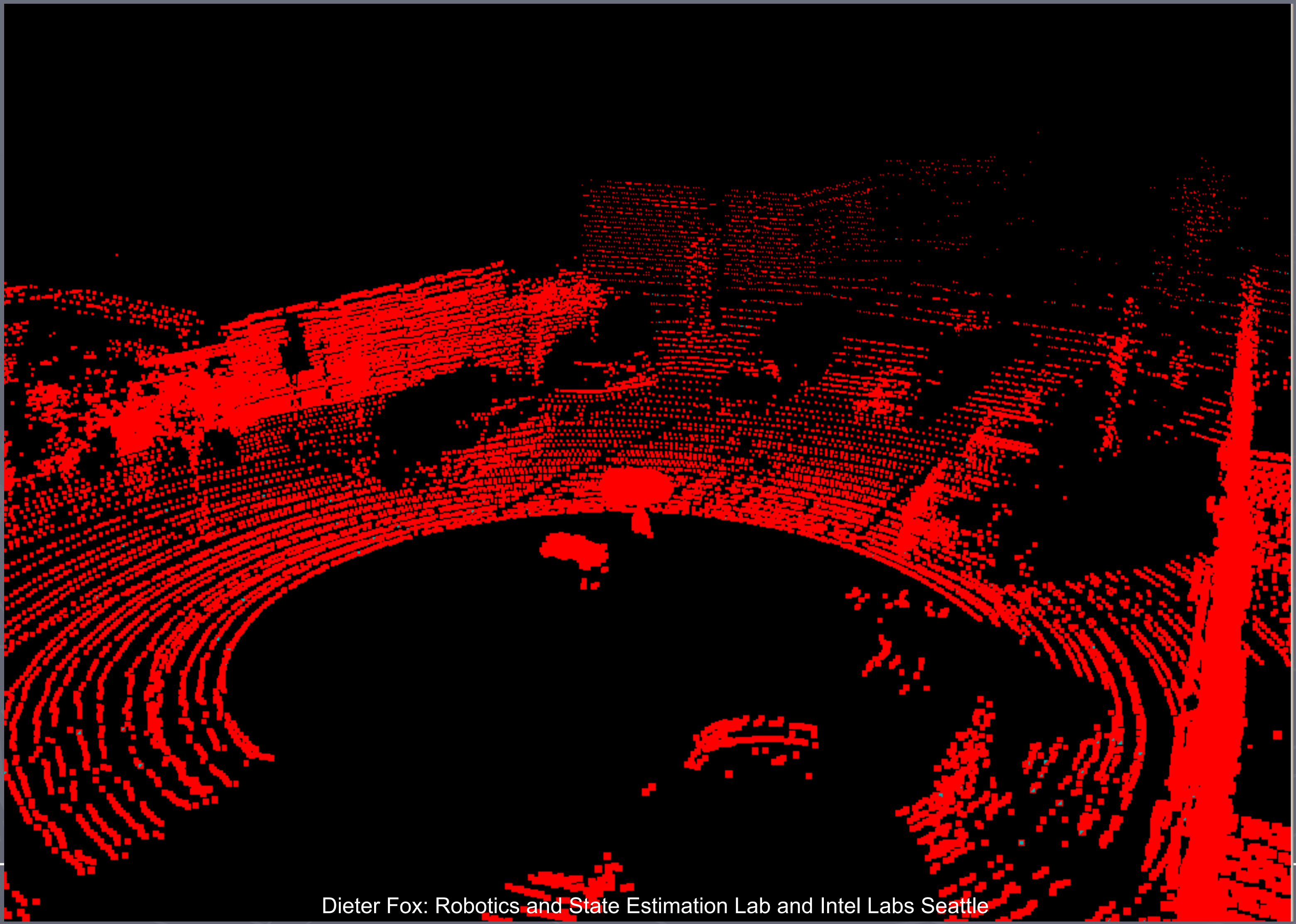
steerable pyramid

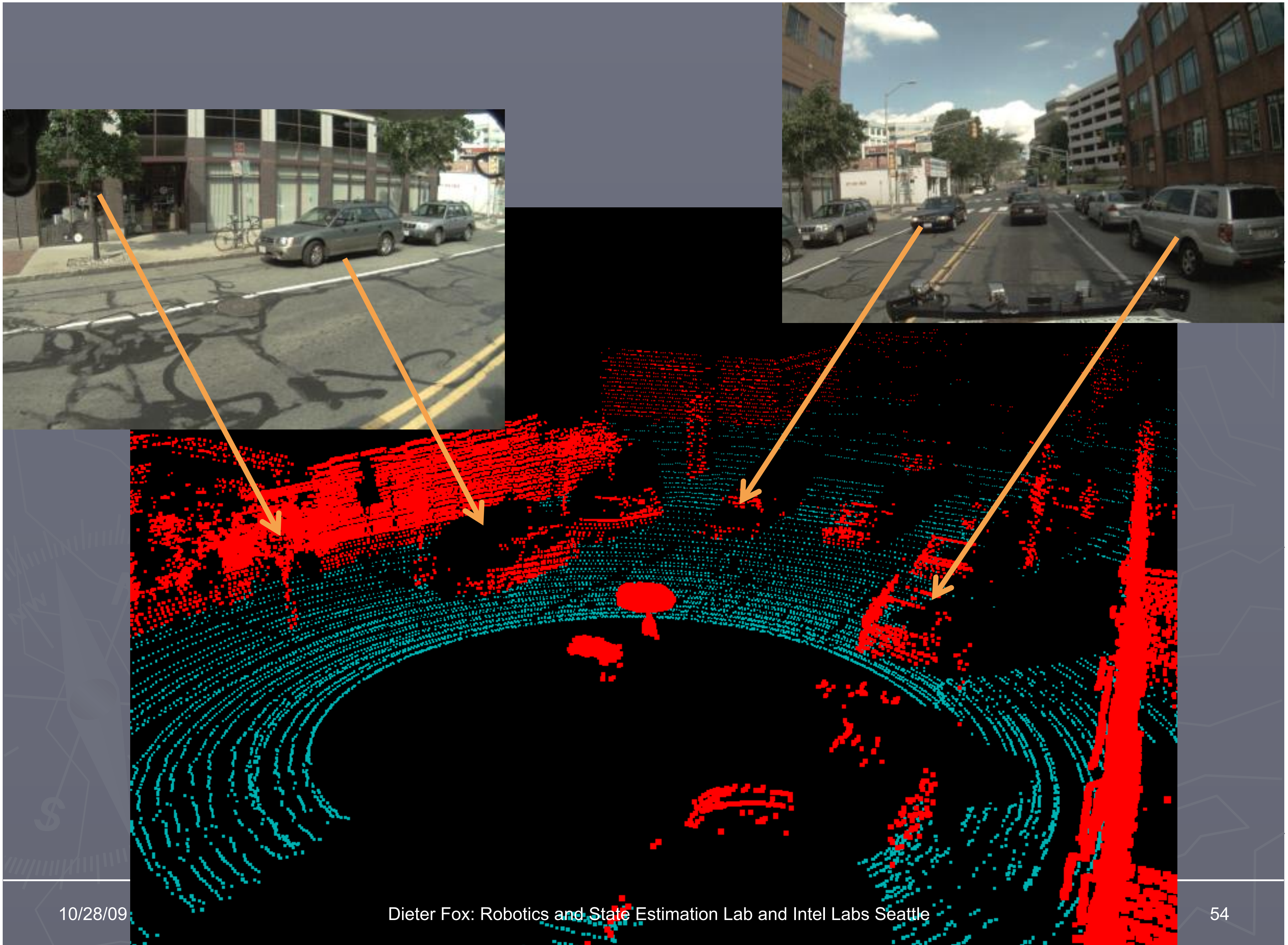


3-d HSV histogram

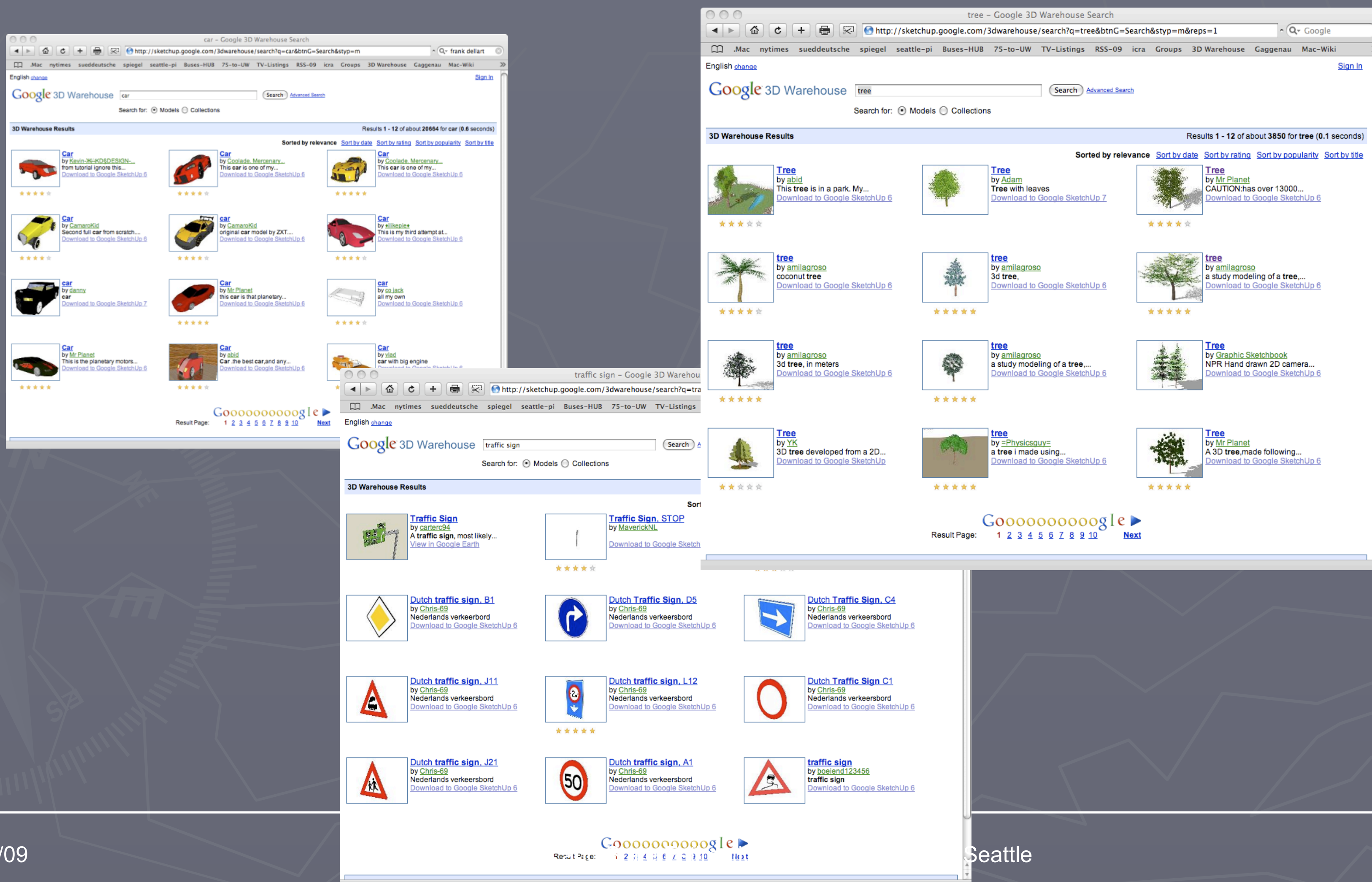
Example Trace

Going 3D

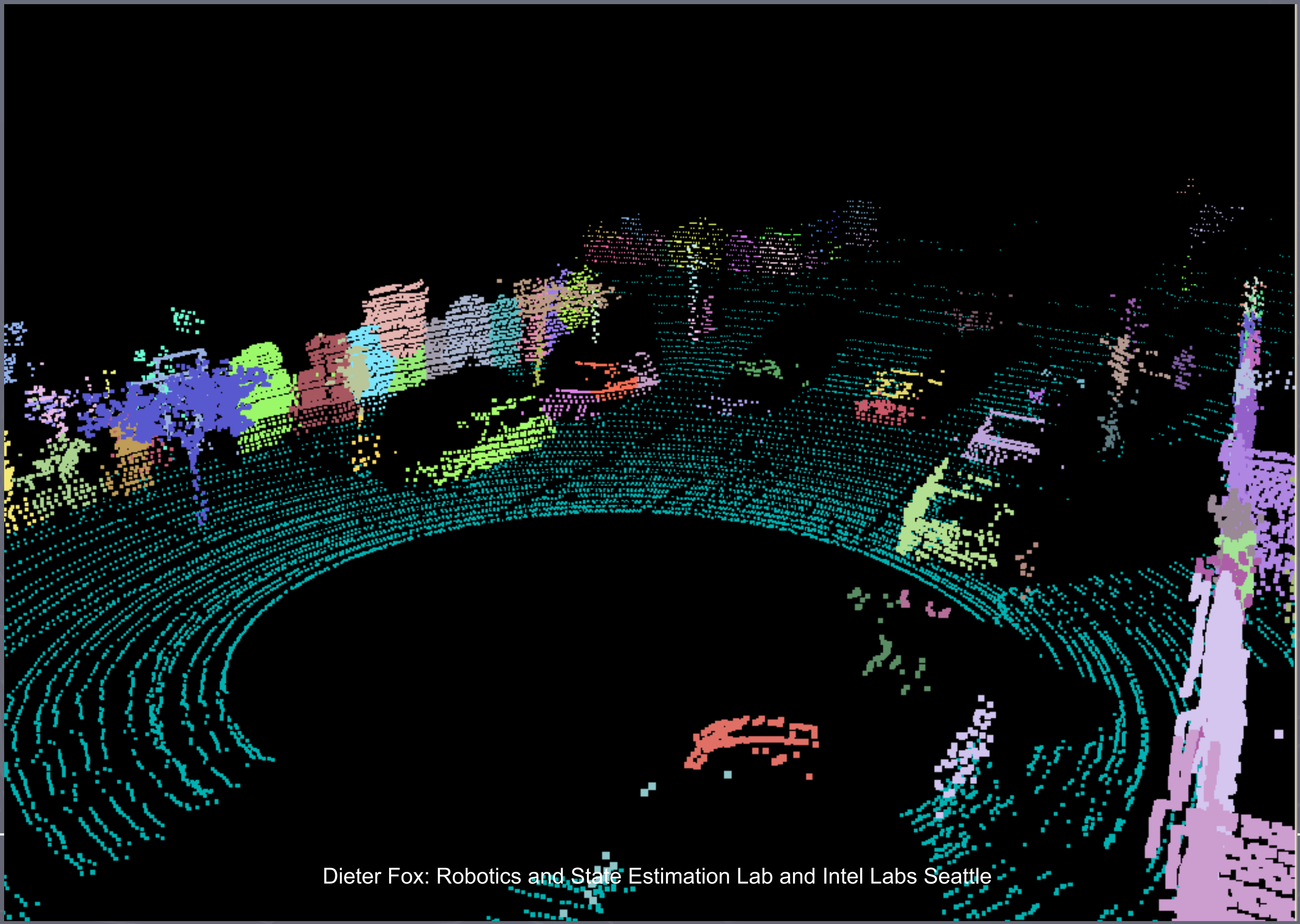




Google 3D-Warehouse: Contains Thousands of Labeled 3D Sketchup Models



Segmentation

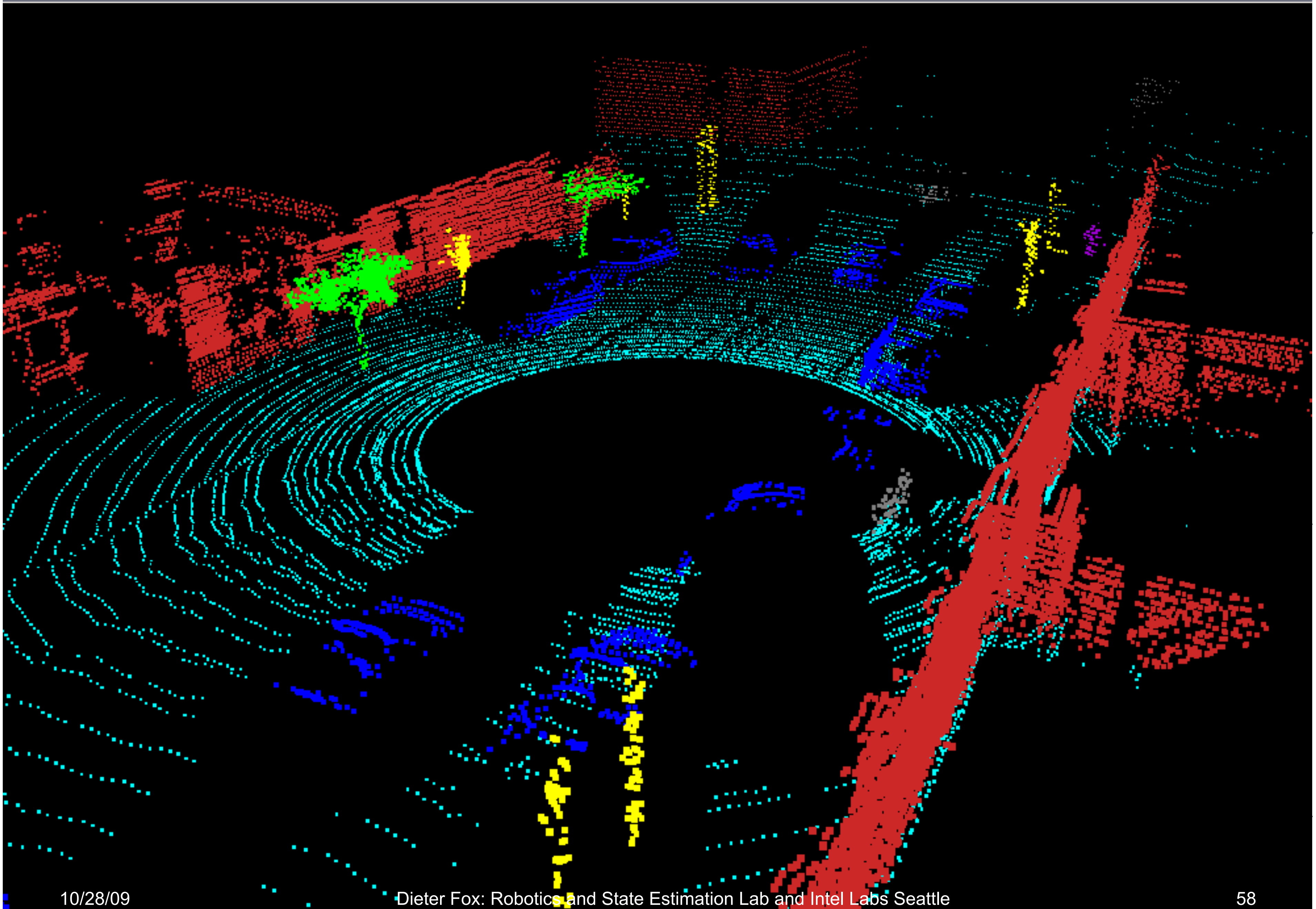


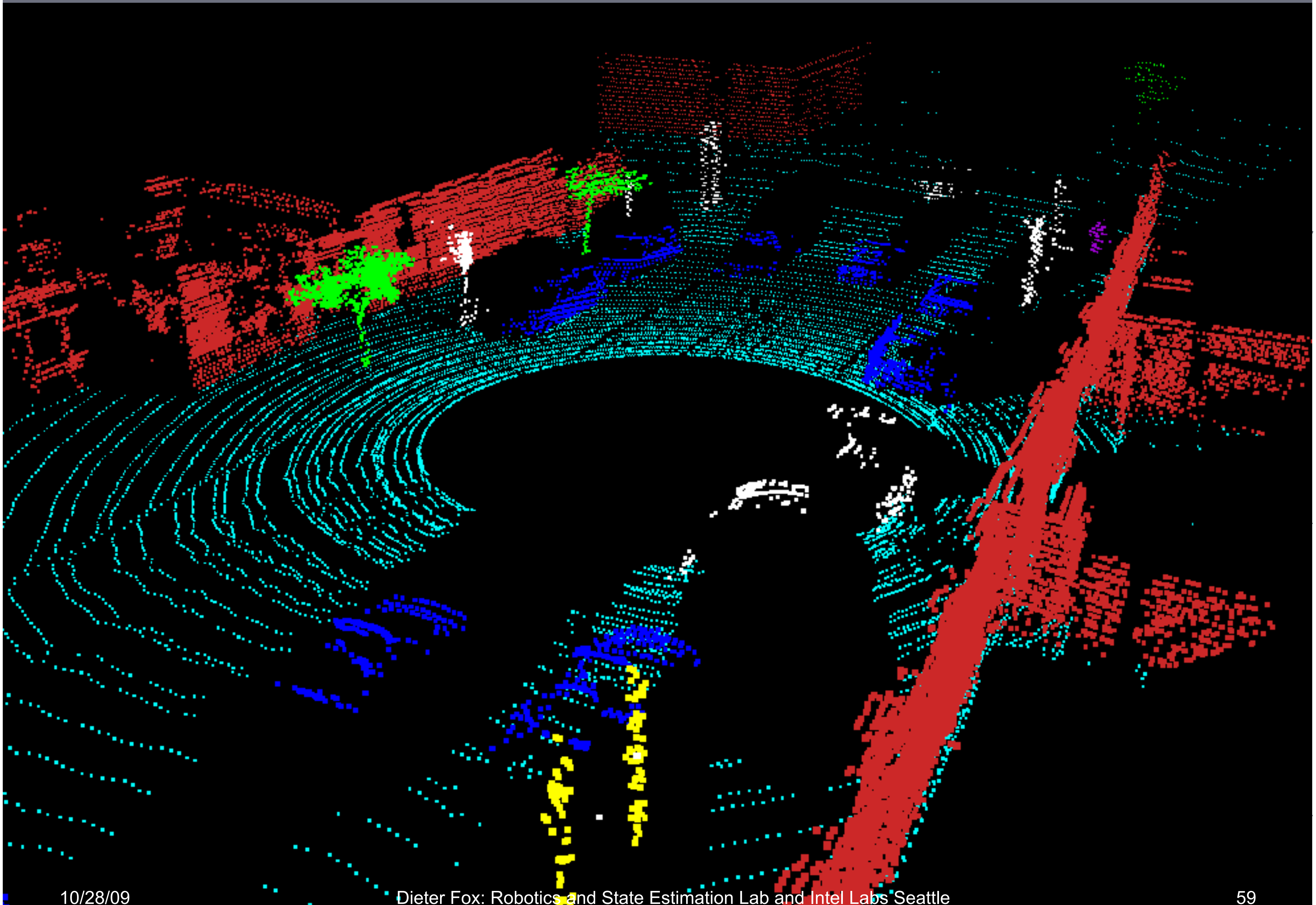
Exemplar Matches

Car

Person

Tree





Outline

- Overview
- Playing soccer with robots
- Exploration and map building
- Object recognition
- More recent stuff

Some Thoughts

- **Robots will**
  - operate in less and less structured environments (military, factory, home, health care, cars, …)
  - interact and share space with humans
- Robustness must increase while cost must go down
- **Key drivers for affordable and robust robots**
  - novel sensing technologies
  - advanced statistical estimation and learning algorithms that can handle uncertainty
- **Focus will shift from mechanics to silicon ☺**

3D Laser Mapping

- 3D point clouds enhanced with visual information
- Navteq, Google, Microsoft, …
- Velodyne: > 50,000 USD

Autonomous Parking

Courtesy W. Burgard

- Velodyne: key sensor in DARPA Urban Challenge

Fast Object Instance Recognition

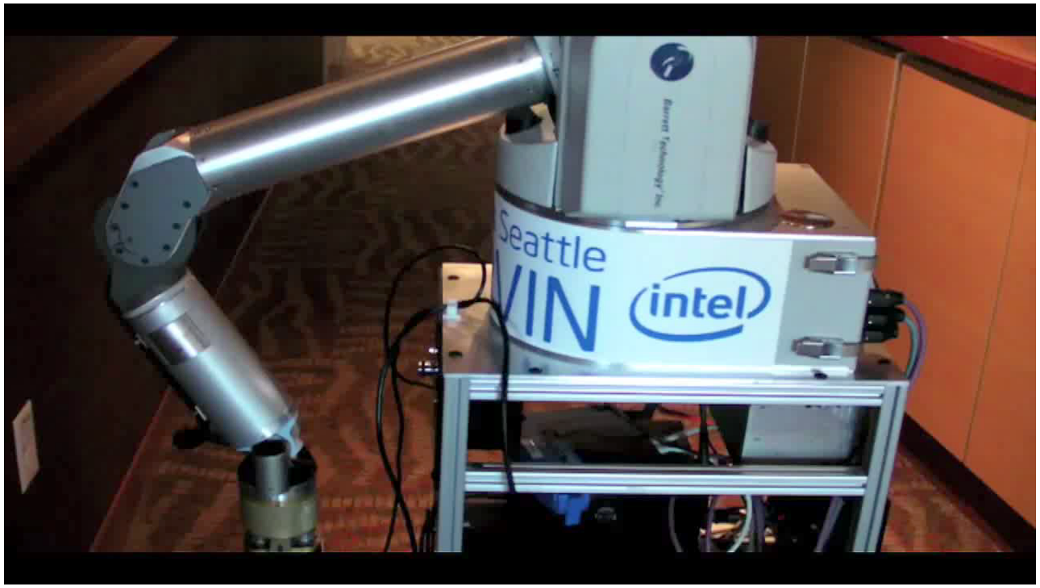
[Romea-Berenson-Srinivasa-Ferguson: ICRA-09]

Mobile Manipulation

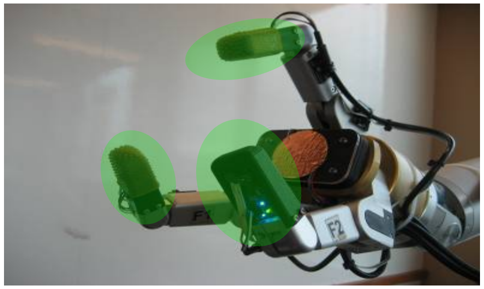
[Berenson-Srinivasa-Ferguson-Kuffner: ICRA-09, Srinivasa-etAl: ARJ-10]

Electric Field Sensing / Pretouch

[Mayton-LeGrand-Smith: ICRA-10, IROS-07]



- Finger tips measure electric field
- Field changes provide information about nearby objects
- Inspired by electric field sensing in fish



Where we are

- **We have**
  - very robust algorithms for mapping and navigation
  - rapidly progressing manipulation and object recognition capabilities
- **Success mostly based on**
  - algorithmic advances: statistical estimation and machine learning
    - require substantial processing power
  - laser range finders
    - still very expensive (2D: 5K, 3D: 50K)
    - cameras cheap but not yet robust enough
- **Still limited representations of environments**
  - Insufficient reasoning about semantic places, objects, and people

RGB-D: Adding Depth to Color

- Soon we'll have cheap depth cameras with high resolution and accuracy

- Key industry drivers: Gaming, entertainment

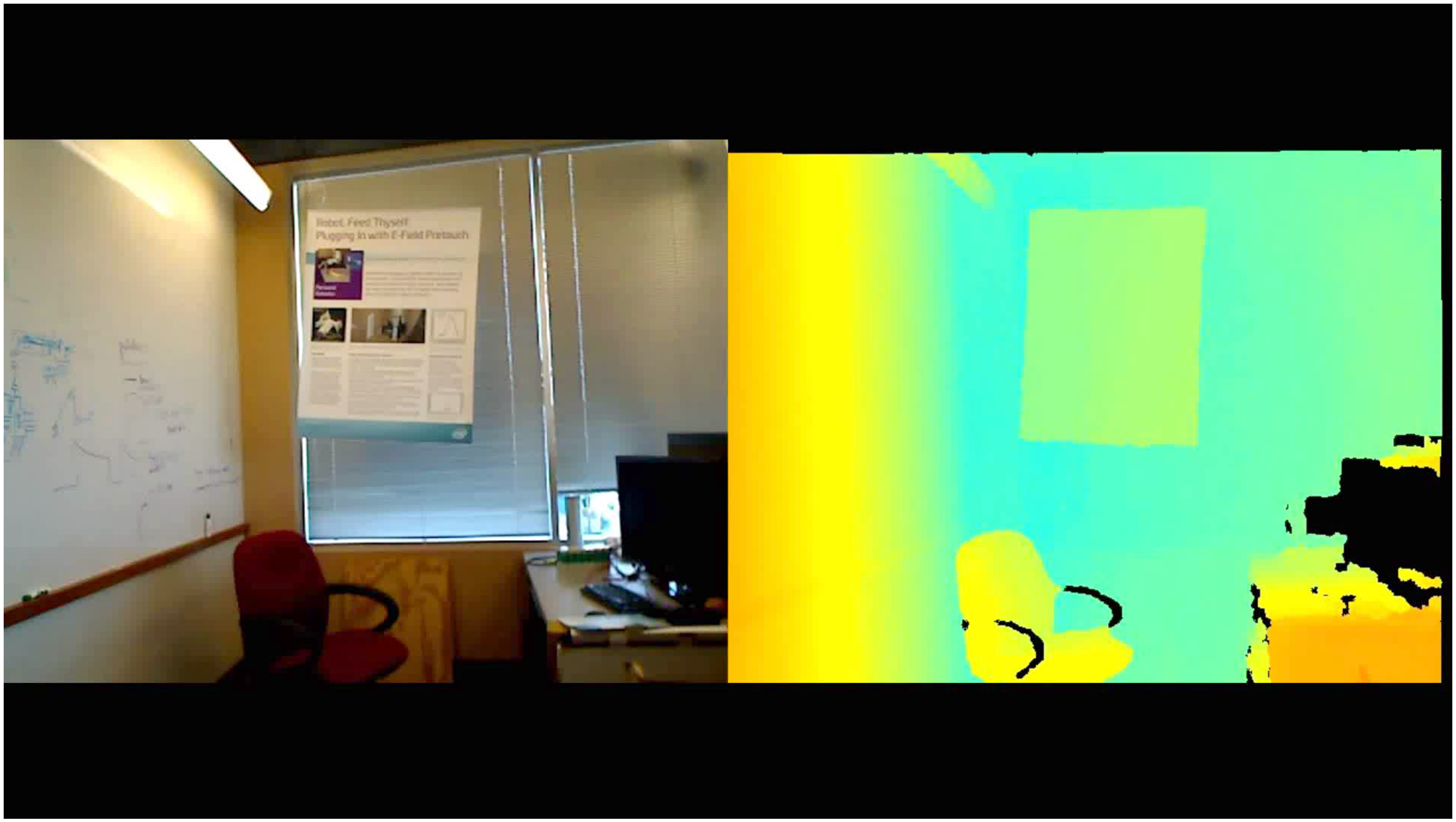
- Two main techniques:
  - Structured light with stereo
  - Time of flight

- Huge impact on gesture recognition, object recognition, mapping, navigation

Gaming

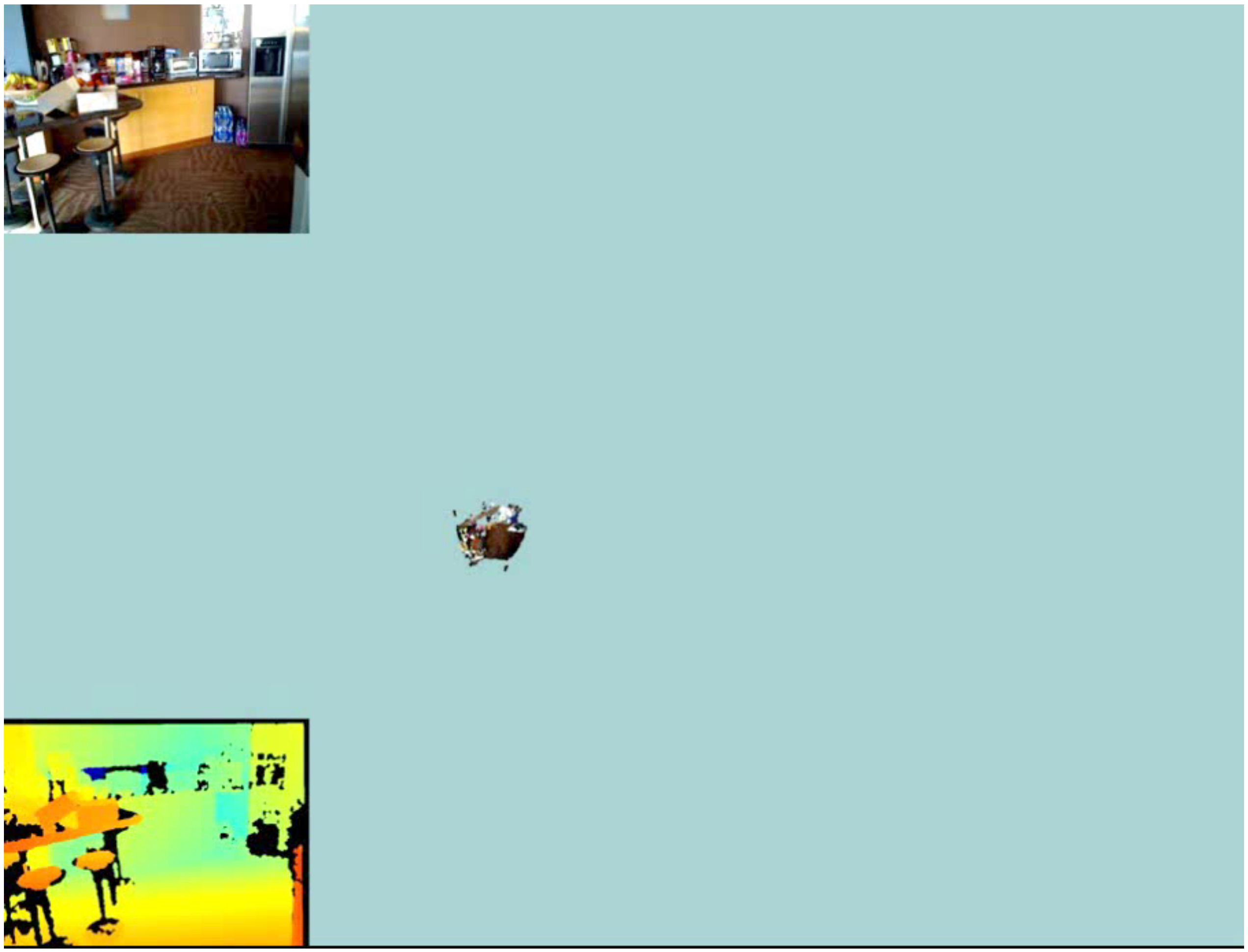
Microsoft Natal promo video

RGB-D: Raw Data

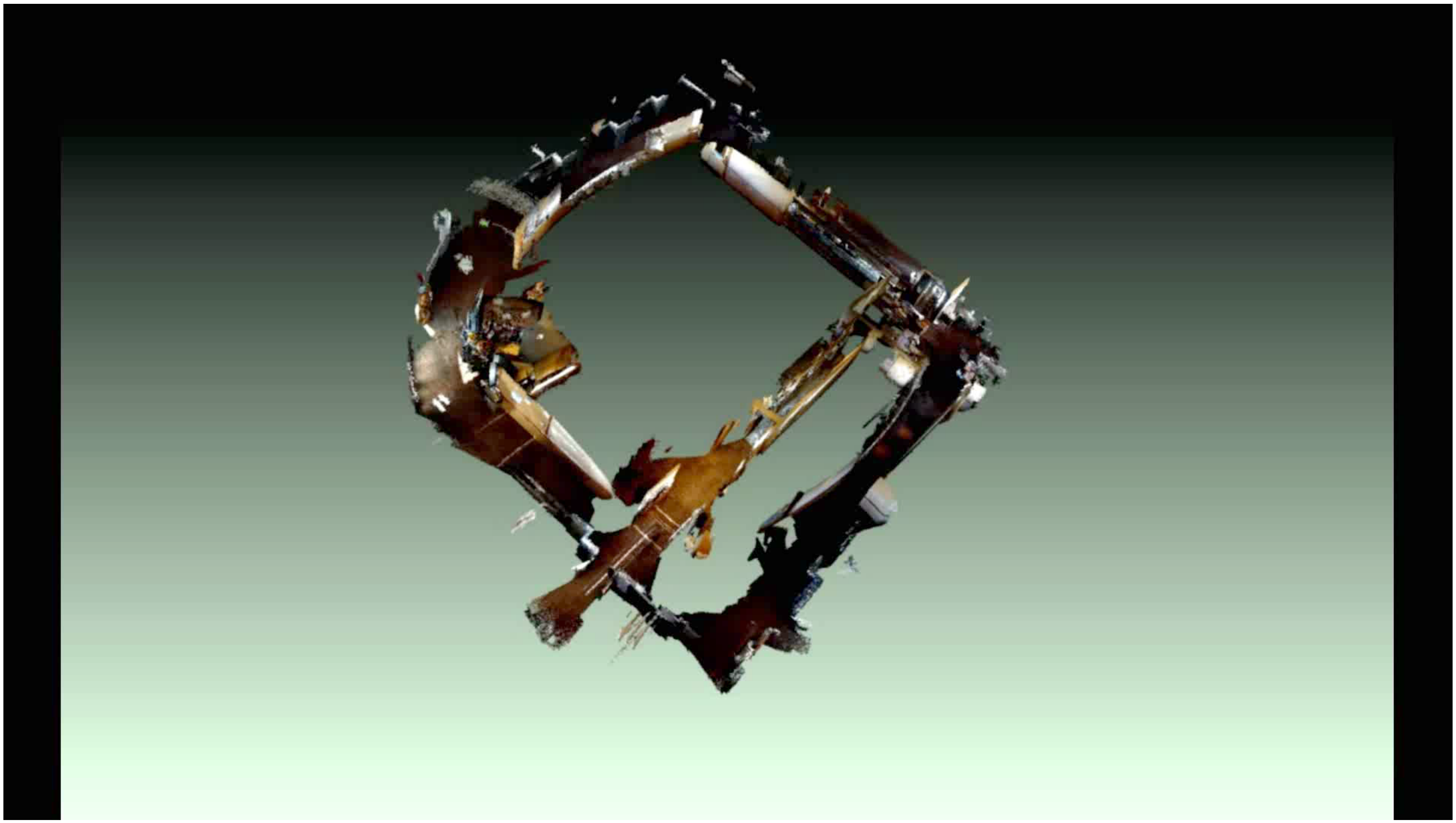


Provides depth typically between 50cm and 5m

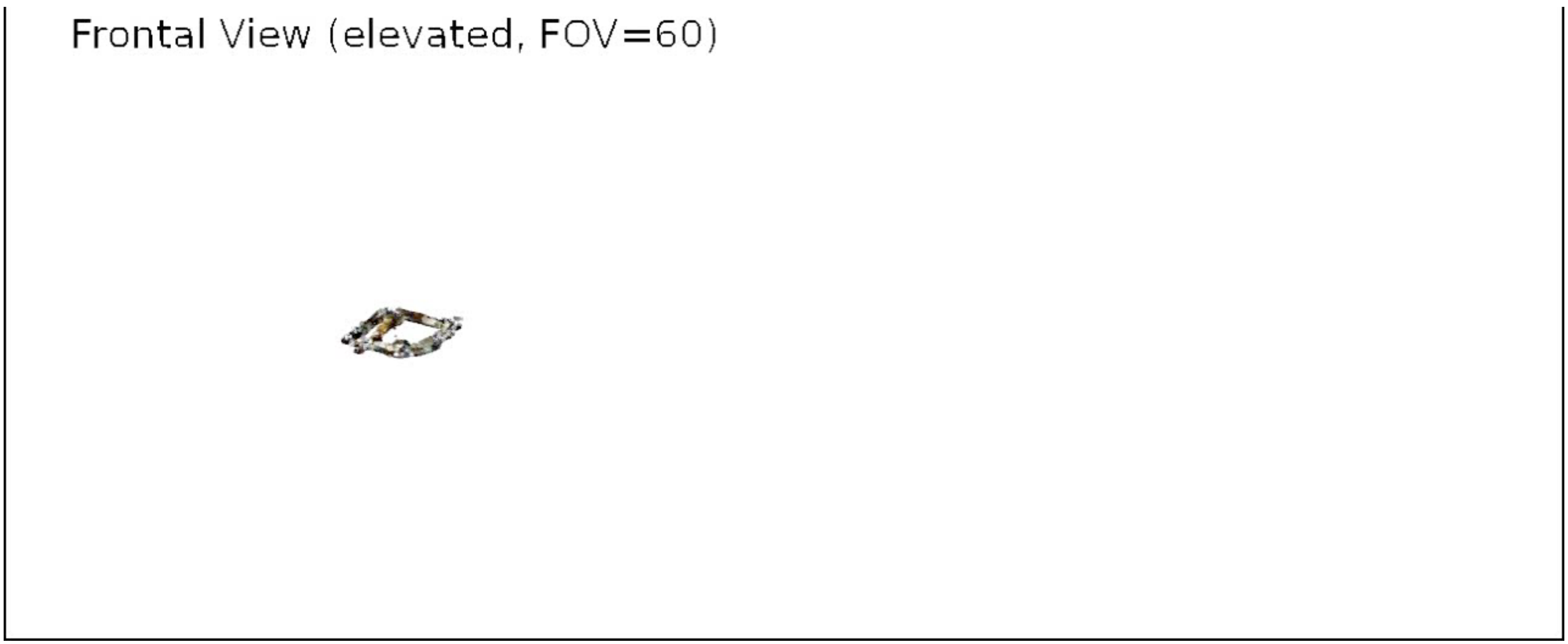
3D Mapping



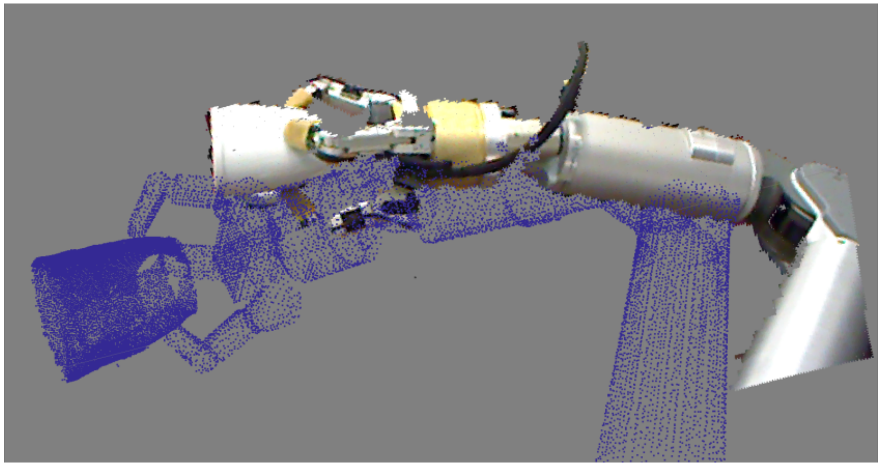
Map



Flythrough



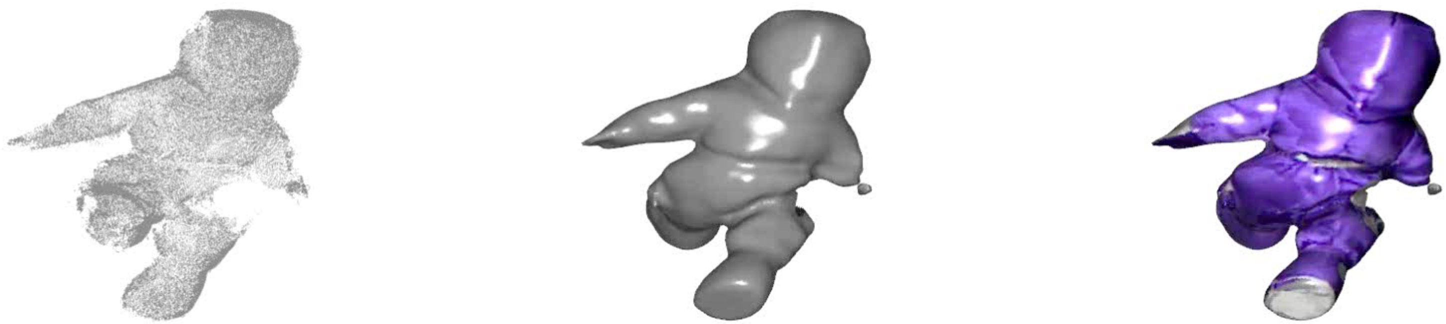
Autonomous Object Modeling



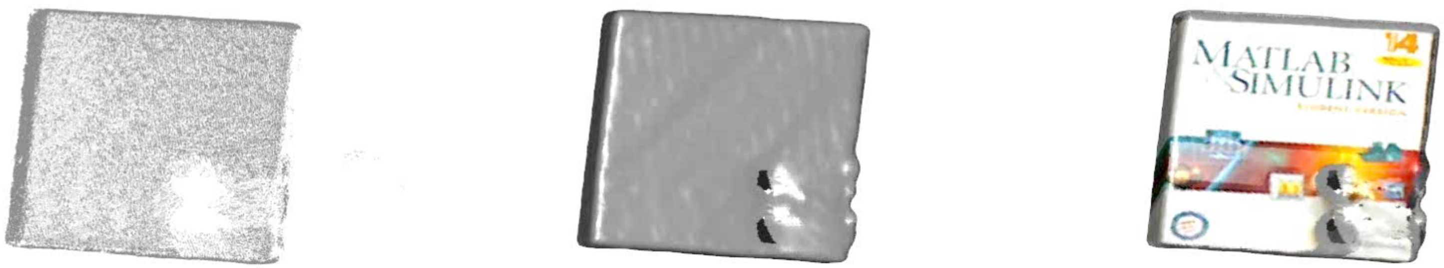
- Enable robots to autonomously learn new objects
- Robot picks up objects and builds models of them
- Camera feedback allows inaccurate manipulator

Autonomous Object Modeling

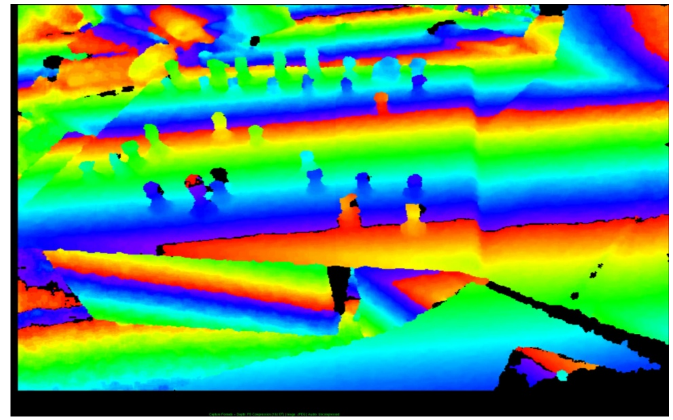
Object Models



Object Models



Smart Gaming Manipulators



Can we build smart manipulators that are cheap enoug

Conclusions

- **Multi purpose robots in unstructured environments**
  - Robust navigation and mapping
  - Maturing manipulation and object recognition
- **Sensing and manipulation hardware still too expensive**
  - Statistical algorithms produce robust and adaptive systems
  - New breed of RGB-D cameras can dramatically decrease cost of robust navigation and interaction platforms
- Focus shifts from mechanics to silicon

Summary

- Whenever computers are connected to the real world, there is no such thing as
  - A perfect sensor
  - A deterministic environment
  - A deterministic robot
  - An accurate model
- Probabilistic approaches and machine learning are key to dealing with the real world