Robotics
Spring 2023

Abhishek Gupta

TAs: Yi Li, Srivatsa GS

Recap: Course Overview

Filtering/Smoothing

Localization

Mapping

SLAM

Search

Motion Planning

TrajOpt

Stability/Certification

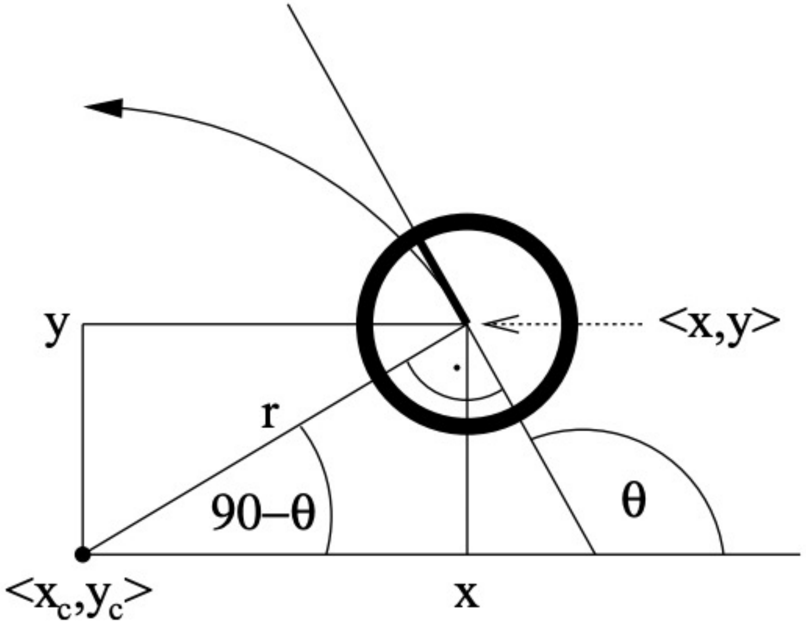
MDPs and RL

Imitation Learning

Solving POMDPs

Recap: Velocity Based Sampling

Generate noise free motion and then add noise to it



Given ν, ω first compute the radius of motion to get x, y and then compute the heading change

$$x_c = x_t - r\sin\theta \qquad r = \frac{\nu}{\omega}$$
$$y_c = y_t + r\cos\theta$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{\omega}\sin(\theta+\omega\Delta t) \\ y_c - \frac{v}{\omega}\cos(\theta+\omega\Delta t) \\ \theta+\omega\Delta t \end{pmatrix}$$
$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta+\omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta+\omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

Add noise to the velocities

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1|v|+\alpha_2|\omega|} \\ \varepsilon_{\alpha_3|v|+\alpha_4|\omega|} \end{pmatrix}$$

Replace ν, ω by $\hat{v}, \widehat{w}$

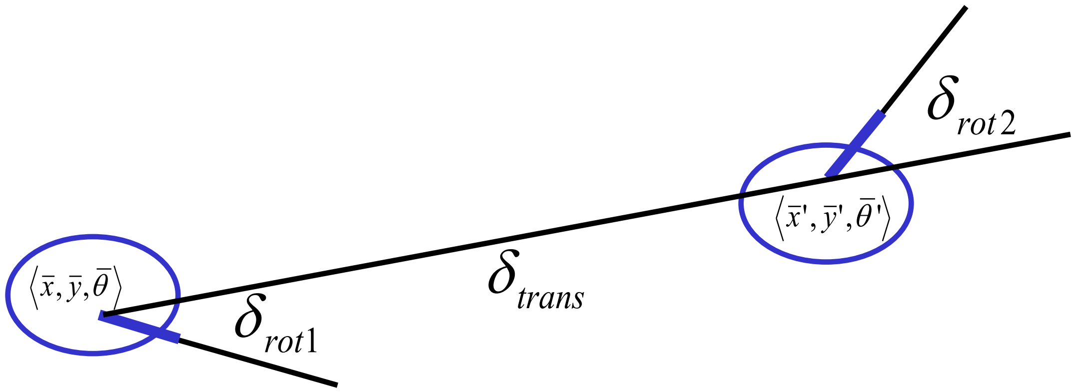
Useful for particle filters

Recap: Odometry Based Model Sampling

Goal: sample x_{t+1} from x_t with action $u = (\bar{x}, \bar{x'})$

1. Reparametrize u from $(\bar{x}, \bar{x'})$ to $(\delta_{\mathrm{rot1}}, \delta_{\mathrm{rot2}}, \delta_{\mathrm{trans}})$
2. Add noise to $(\delta_{\mathrm{rot1}}, \delta_{\mathrm{rot2}}, \delta_{\mathrm{trans}})$ to get $(\hat{\delta}_{\mathrm{rot1}}, \hat{\delta}_{\mathrm{rot2}}, \hat{\delta}_{\mathrm{trans}})$
3. Compute next state x_{t+1} from $(\hat{\delta}_{\mathrm{rot1}}, \hat{\delta}_{\mathrm{rot2}}, \hat{\delta}_{\mathrm{trans}})$

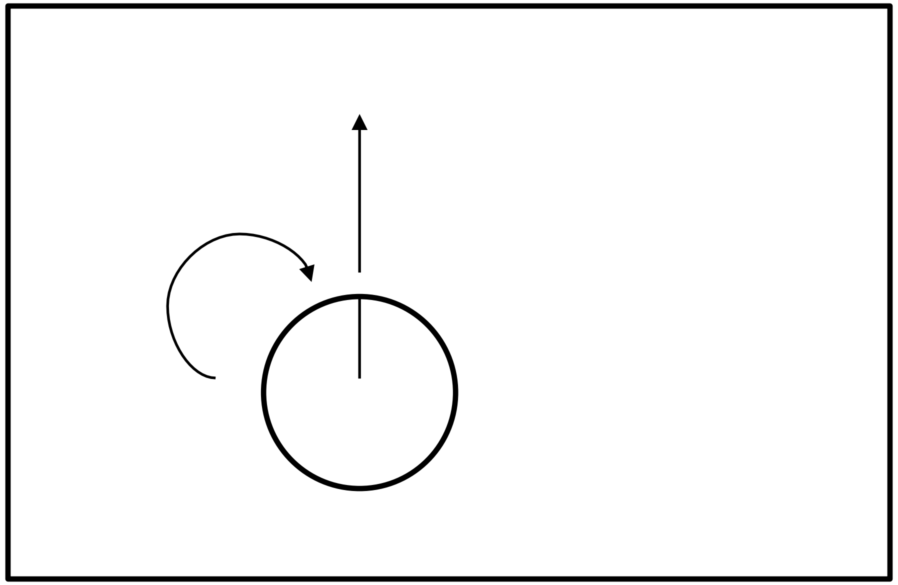
Key idea: odometry gives you change in angles, this is noisy and gives next state

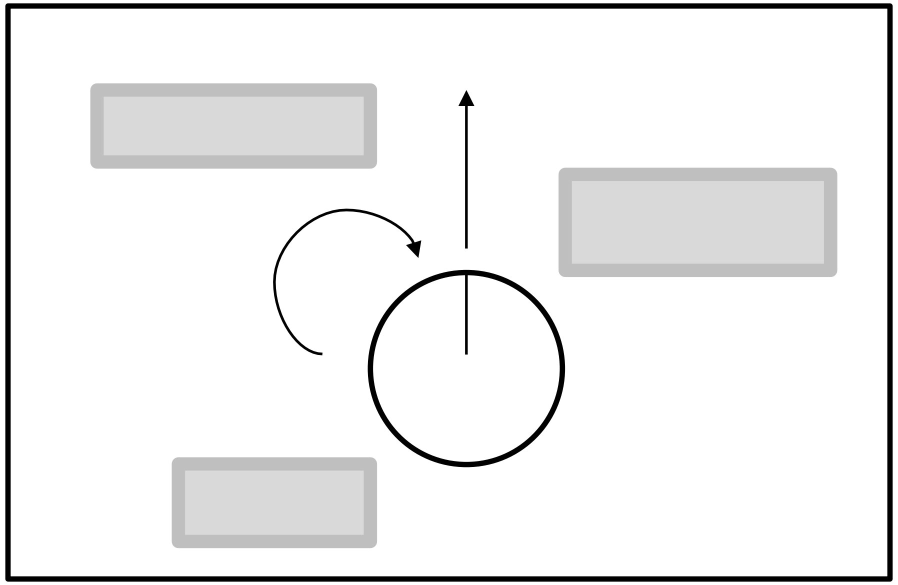


Integrating Maps into Motion Models

From free space motion models to maps

Free space motion models do not account for obstacles in a **known** map

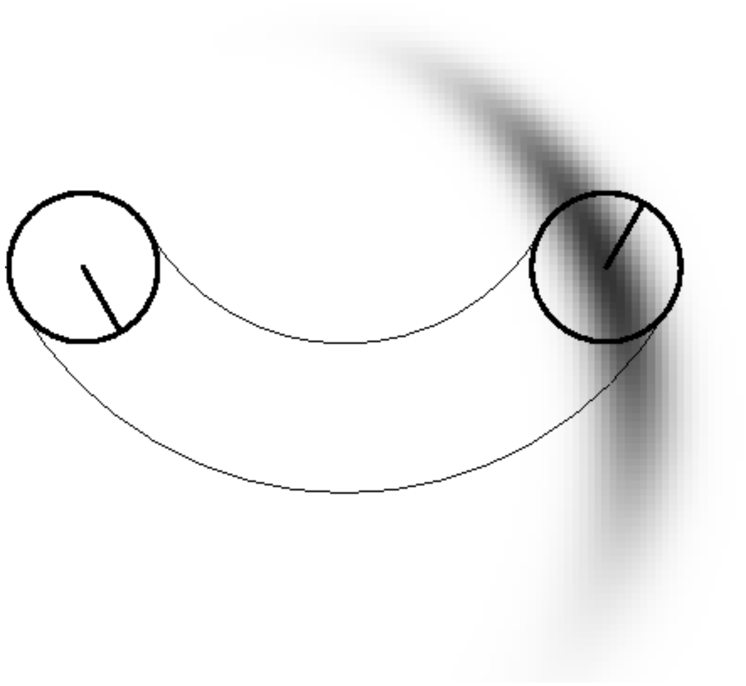




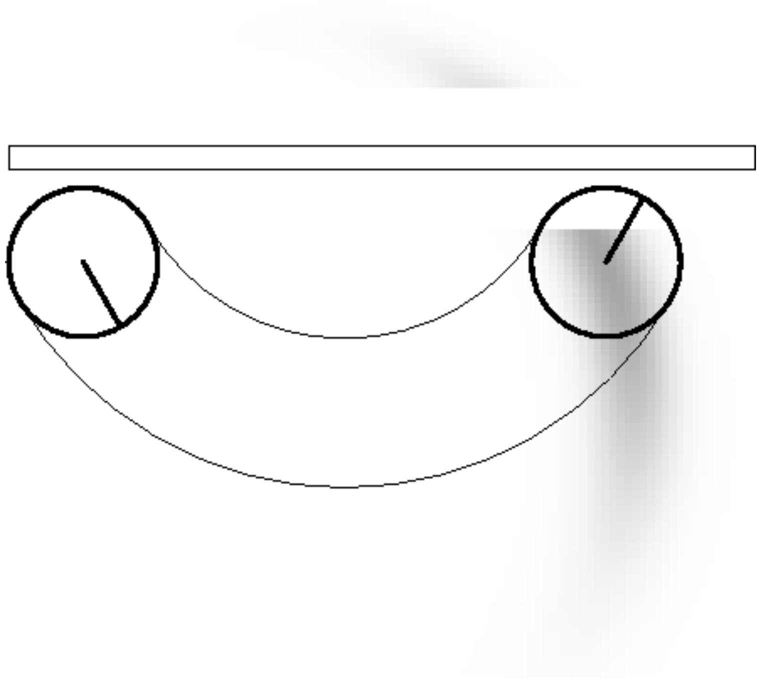
$$p(x'|u,x,m) \approx p(x'|m)p(x'|x,u)$$

Zero-out positions that are not possible in the map

Motion Model with Map



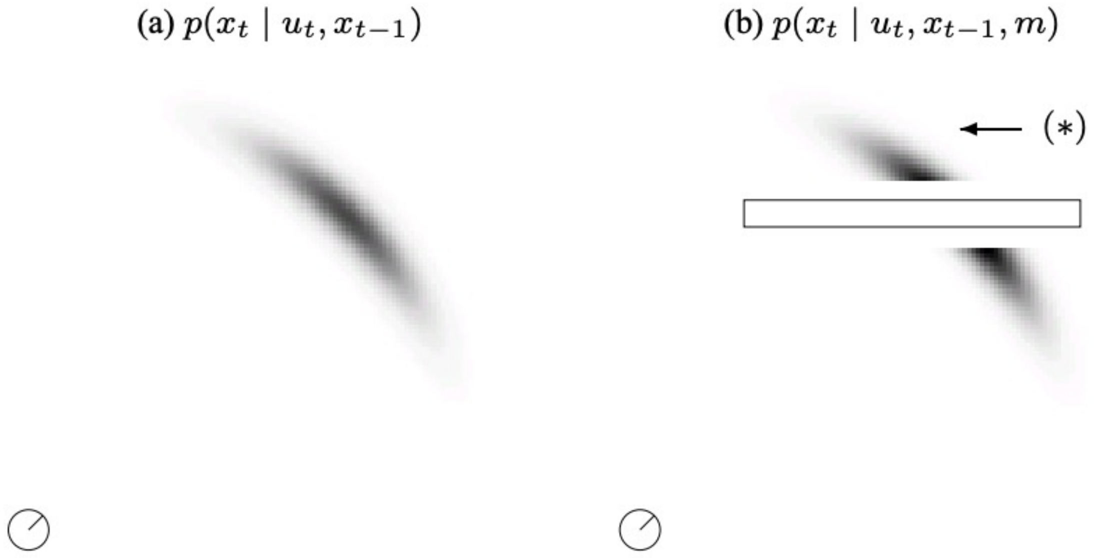
$$p(x'|u,x)$$



$$p(x'|u,x,m) \approx p(x'|m)p(x'|x,u)$$

Failure Case

Free space



Obstacle

Don’t account for motion through walls → deal with by increasing frequency

Lecture Outline

Sensor Models

↓

Parameter Estimation

↓

Occupancy Mapping

Sensor Models for Bayesian Filtering

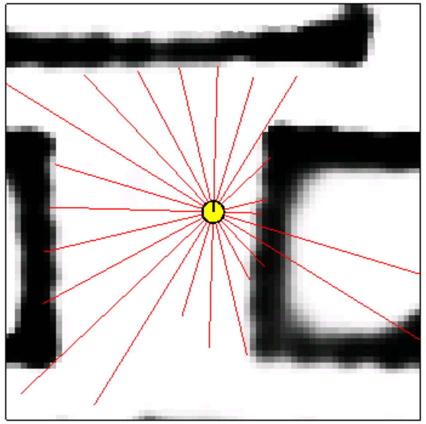
$$Bel(x_t) = P(x_t | u_{0:t-1}, z_{0:t})$$
$$= \eta \, p(z_t | x_t) \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

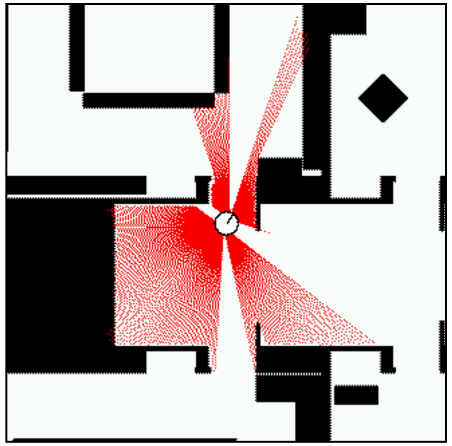
Let's try and specify what this is

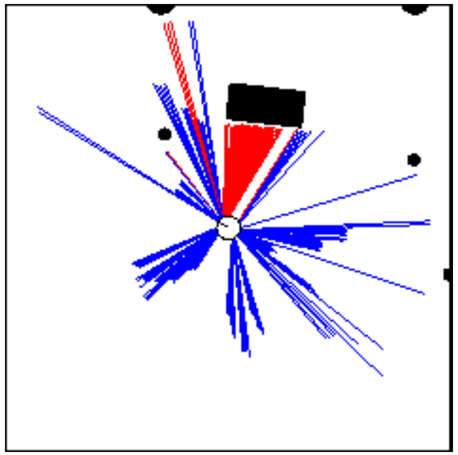
Sensors for Mobile Robots

- Contact sensors: Bumpers, touch sensors
- Internal sensors
  - Accelerometers (spring-mounted masses)
  - Gyroscopes (spinning mass, laser light)
  - Compasses, inclinometers (earth magnetic field, gravity)
  - Encoders, torque
- Proximity sensors
  - Sonar (time of flight)
  - Radar (phase and frequency)
  - Laser range-finders (triangulation, tof, phase)
  - Infrared (intensity)
- Visual sensors: Cameras, depth cameras
- Satellite-style sensors: GPS, MoCap

Proximity Sensors







- The central task is to determine ***P(z|x)***, i.e. the probability of a measurement ***z*** given that the robot is at position ***x***.
- **Question**: Where do the probabilities come from?
- **Approach**: Let's try to explain a measurement.

Beam-based Sensor Model

Beam-based Sensor Model

- Scan ***z*** consists of ***K*** measurements.

$$z = \{z_1, z_2, \ldots, z_K\}$$

Beam-based Sensor Model

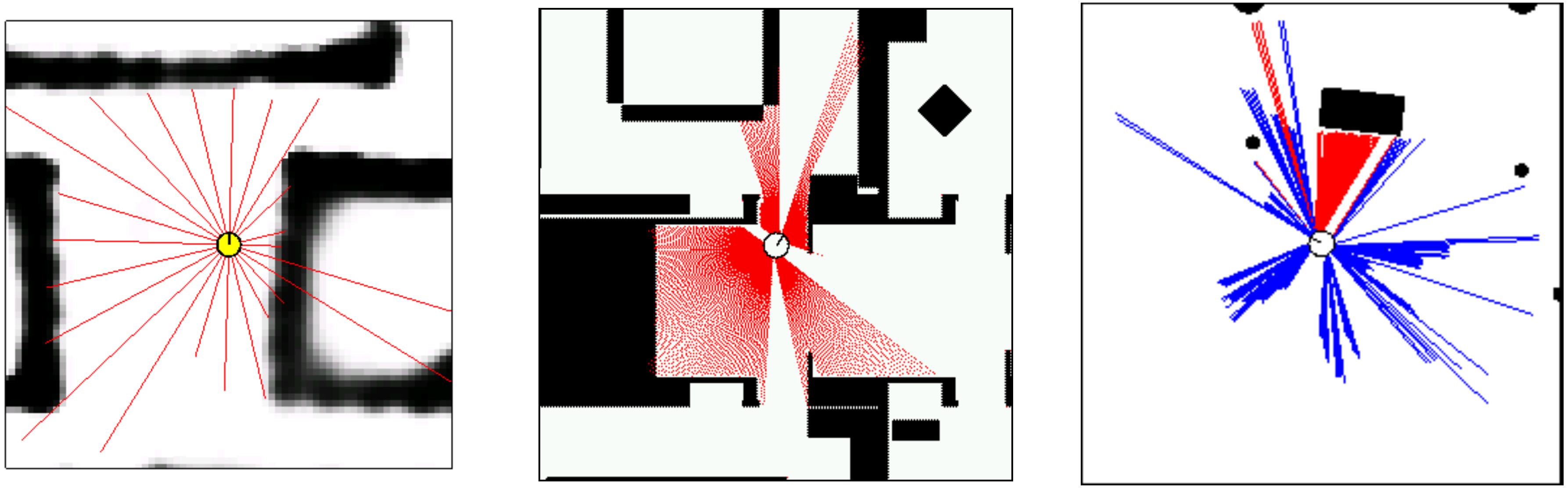
- Scan ***z*** consists of ***K*** measurements.

$$z = \{z_1, z_2, \dots, z_K\}$$

- Individual measurements are independent given the robot position and a map.

$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$

Beam-based Sensor Model



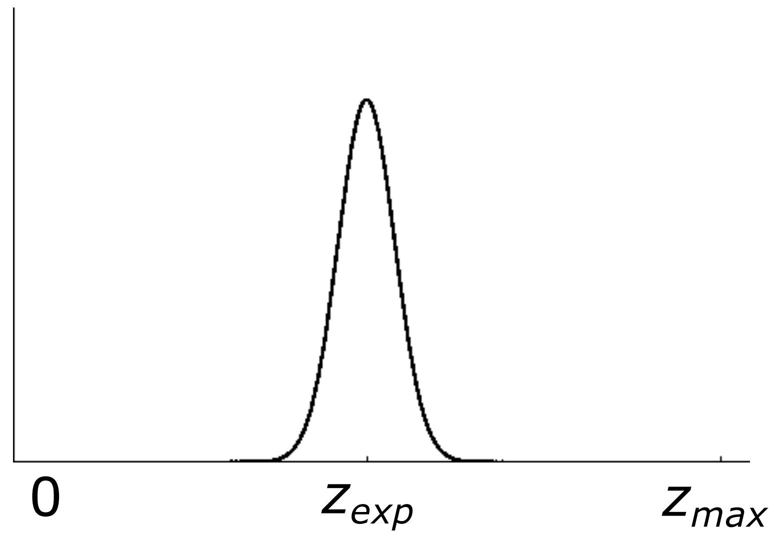
$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$

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.

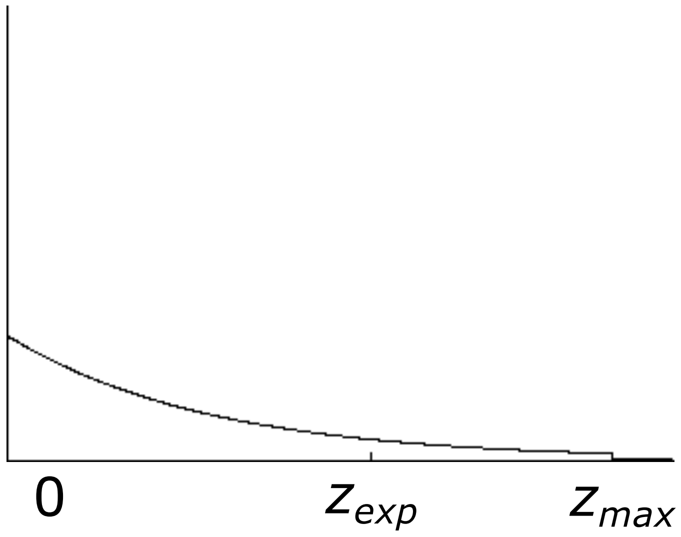
Beam-based Proximity Model

Measurement noise



$$P_{hit}(z \mid x,m) = \eta \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(z-z_{\exp})^2}{\sigma^2}}$$

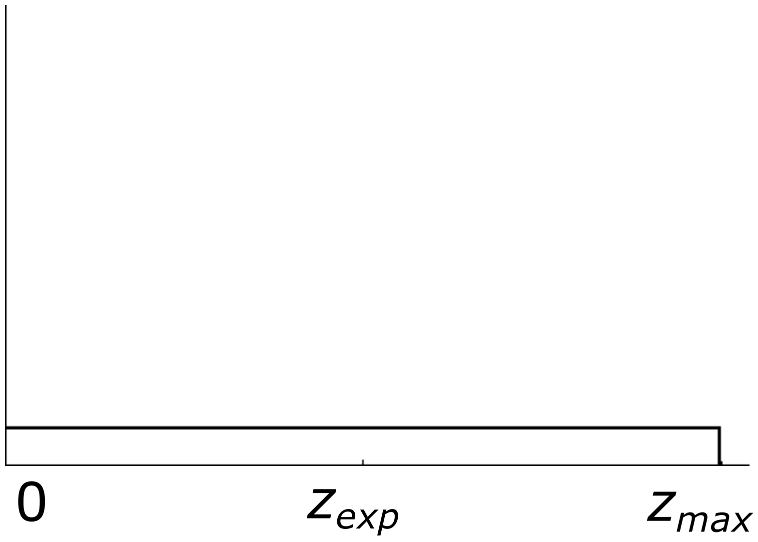
Unexpected obstacles



$$P_{\text{unexp}}(z \mid x,m) = \eta\ \lambda\ \mathrm{e}^{-\lambda z}$$

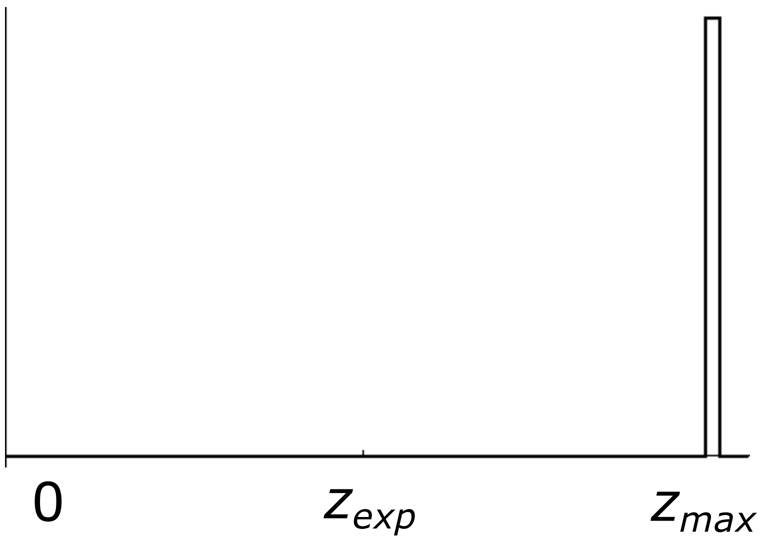
Beam-based Proximity Model

Random measurement



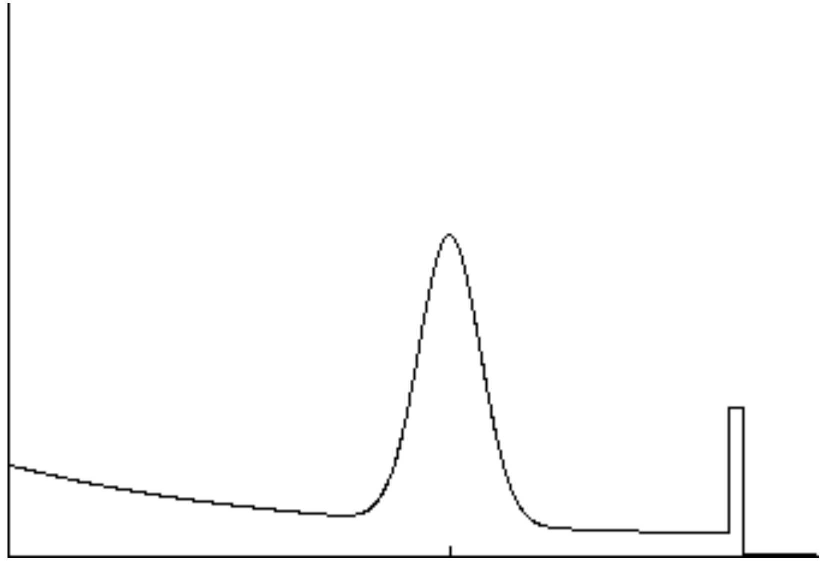
$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

Max range



$$P_{\max}(z \mid x, m) = \eta \frac{1}{z_{small}}$$

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?

→ More on this next

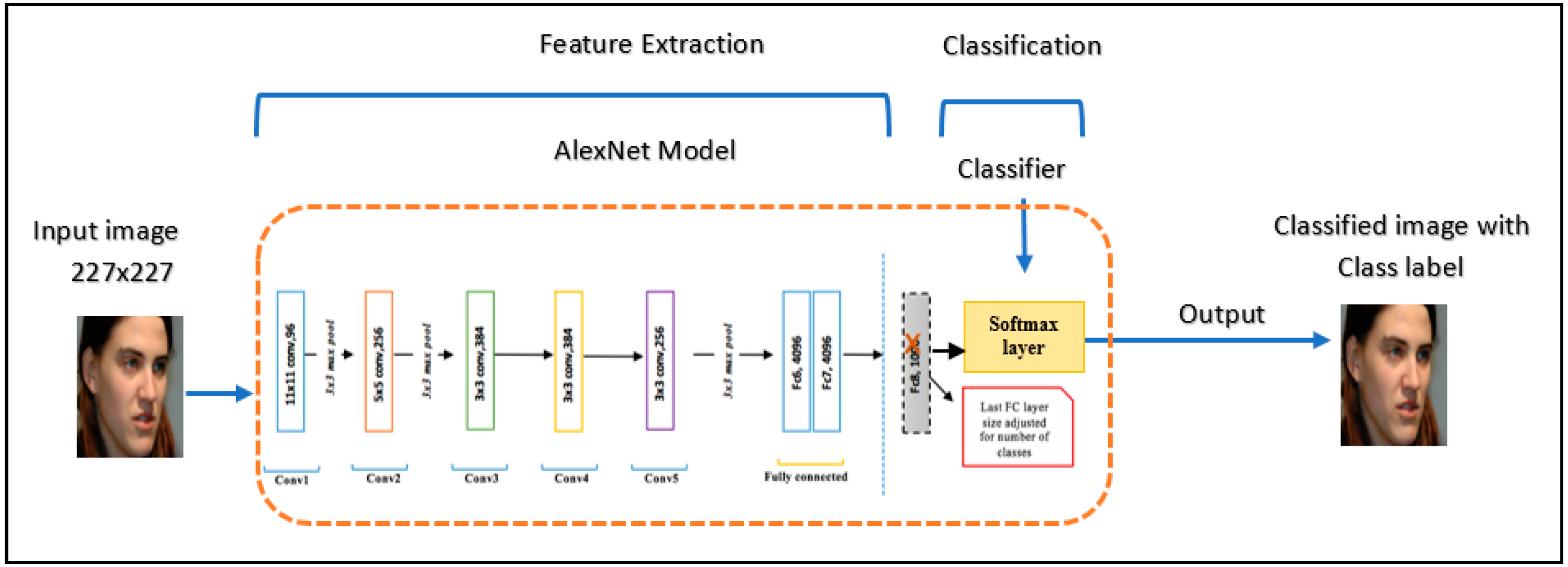
Summary Beam-based Model

- Assumes independence between beams.
  - Overconfident!
- Models physical causes for measurements.
  - Mixture of densities for these causes.
- Implementation
  - Learn parameters based on real data.
  - Different models can be learned for different angles at which the sensor beam hits the obstacle.
  - Determine expected distances by ray-tracing.
  - Expected distances can be pre-processed.

Landmark-based Sensor Model

When are raw measurement based models not enough?

- Scales unfavorably with dimensionality of the measurement

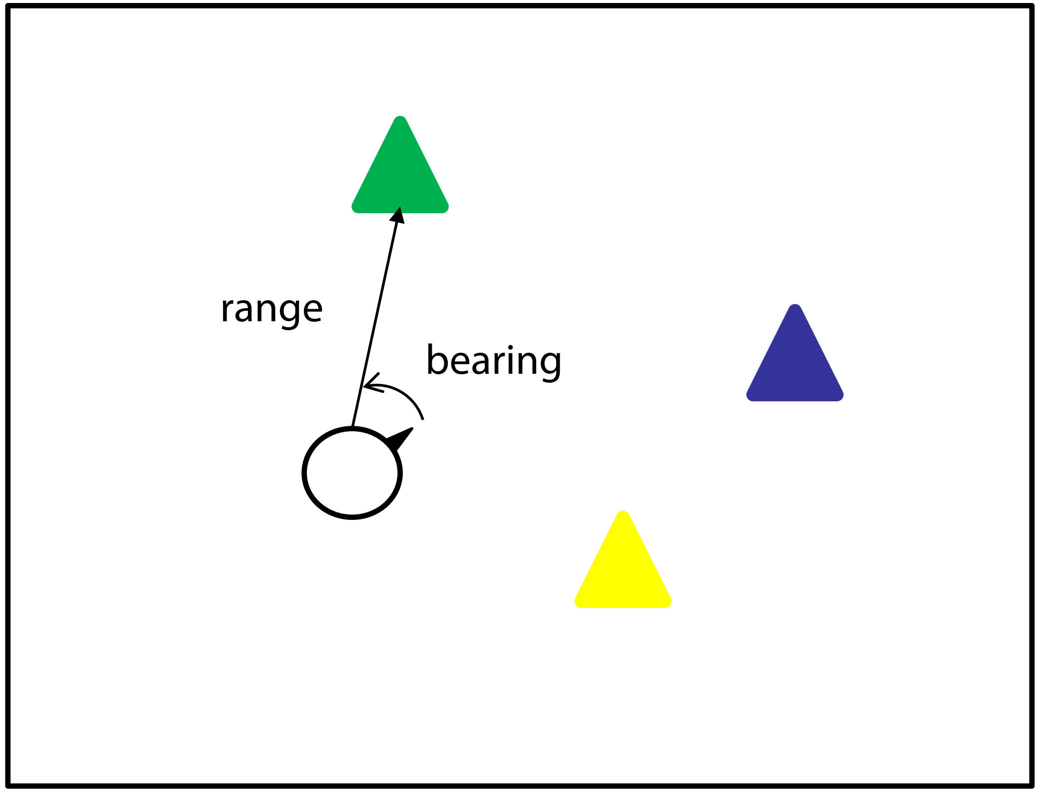


- Common strategy in machine learning - extract low dimensional features

Landmarks

- Active beacons (***e.g***. radio, GPS)
- Passive (***e.g***. visual, retro-reflective)
- Sensor provides
  - distance, or
  - bearing, or
  - distance and bearing.
  - signature

Distance and Bearing



Probabilistic Model

1. Algorithm **landmark_detection_model**(z,x,m):
   $z = \langle i, d, \alpha \rangle, x = \langle x, y, \theta \rangle$

Compute expected range/bearing

2. $\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$
3. $\hat{\alpha} = \mathrm{atan2}(m_y(i) - y, m_x(i) - x) - \theta$

Compute likelihood

4. $p_{\mathrm{det}} = \mathrm{prob}(\hat{d} - d, \varepsilon_d) \cdot \mathrm{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$

5. Return $z_{\mathrm{det}} p_{\mathrm{det}} + z_{\mathrm{fp}} P_{\mathrm{uniform}}(z \mid x, m)$

HW 1EKF Correction Step Pseudocode: Landmark Style

1. def EKF_correction($\mu_{t+1|0:t}$, $\Sigma_{t+1|0:t}$, u_t, z_{t+1}):
2. Linearize measurement:

$$H = \begin{bmatrix} \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} & \frac{\partial \phi}{\partial \theta} \end{bmatrix}$$

3. Correction:

$$K_{t+1} = \Sigma_{t+1|0:t} H^T (H \Sigma_{t+1} H^T + R)^{-1}$$
$$\mu_{t+1|0:t+1} = \mu_{t+1|0:t} + K_{t+1}(z_{t+1} - \boxed{h(\mu_{t+1|0:t})})$$
$$\Sigma_{t+1|0:t+1} = (I - K_{t+1} H)\Sigma_{t+1|0:t}$$

4. Return $\mu_{t+1|0:t+1}$, $\Sigma_{t+1|0:t+1}$

State – (x, y, θ)
Measurement – ϕ

(assumes d is perfectly known)

$$\phi = \boxed{\mathrm{atan2}(l_y - y, l_x - x) - \theta} + \delta$$

$$\delta \sim \mathcal{N}(0, \underbrace{\sigma_\delta^2}_{\text{R}})$$

Summary of Parametric Motion and Sensor Models

- Explicitly modeling uncertainty in motion and sensing is key to robustness.
- In many cases, good models can be found by the following approach:
  1. Determine parametric model of noise free motion or measurement.
  2. Analyze sources of noise.
  3. Add adequate noise to parameters (eventually mix in densities for noise).
  4. Learn (and verify) parameters by fitting model to data.
  5. Likelihood of measurement is given by "probabilistically comparing" the actual with the expected measurement.
- It is extremely important to be aware of the underlying assumptions!

Lecture Outline

Sensor Models

↓

Parameter Estimation

↓

Occupancy Mapping

But where do the actual noise values and A, B come from?

- Case 1: Fully observed training
  - We have an oracle technique to observe x, z, u (motion capture) at training time.
  - Maximize likelihood
- Case 2: Partially observed training
  - We can only observe z, u
  - Expectation-Maximization

Case 1: Observed z, x, u – Maximize Likelihood

- Maximize log likelihood of the data (z, x, u) under the motion and sensor models

Linear Gaussian

$$\boxed{\begin{aligned} x_{t+1} &= Ax_t + Bu_t + \epsilon_t \\ \epsilon_t &\sim \mathcal{N}(0, Q) \\ z_{t+1} &= Cx_{t+1} + \delta_t \\ \delta_t &\sim \mathcal{N}(0, R) \end{aligned}}$$

$\longrightarrow$

$$\max_{A,B,Q} \mathbb{E}_{(x,u,x')} \left[\hat{p}(x'|x,u)\right]$$

$$\hat{p}(.|x,u) = \mathcal{N}(Ax + Bu, Q)$$

$$\max_{C,R} \mathbb{E}_{(z,x)} \left[\hat{p}(z|x)\right]$$

$$\hat{p}(.|x) = \mathcal{N}(Cx, R)$$

Non-linear

Solve with LS or SGD

$$\boxed{\begin{aligned} x_{t+1} &= g(x_t, u_t) + \epsilon_t \\ \epsilon_t &\sim \mathcal{N}(0, Q) \\ z_t &= h(x_t) + \delta_t \\ \delta_t &\sim \mathcal{N}(0, R) \end{aligned}}$$

$\longrightarrow$

$$\max_{\theta} \mathbb{E}_{(x,u,x')} \left[\hat{p}(x'|x,u)\right]$$

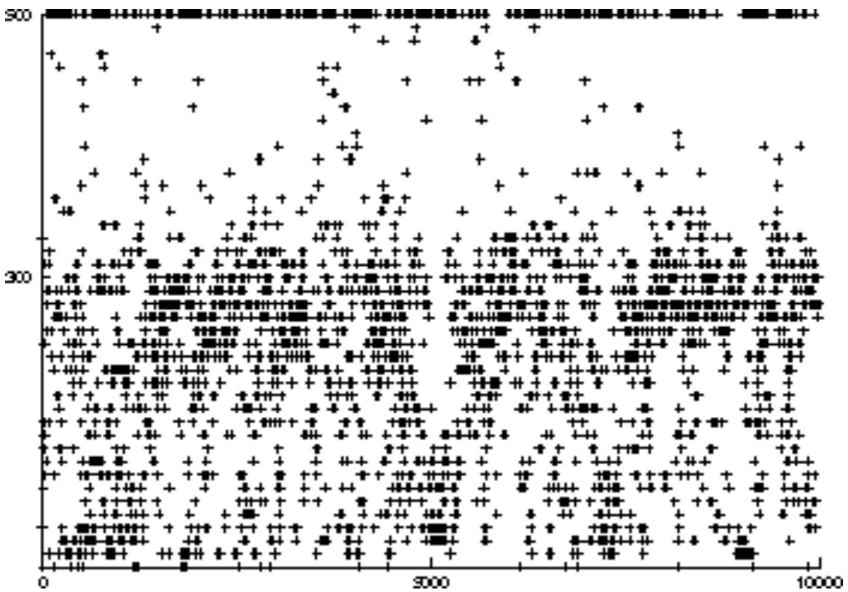
$$\hat{p}(.|x,u) = \mathcal{N}(g_\theta(x,u), Q_\theta)$$

$$\max_{\phi} \mathbb{E}_{(z,x)} \left[\hat{p}(z|x)\right]$$

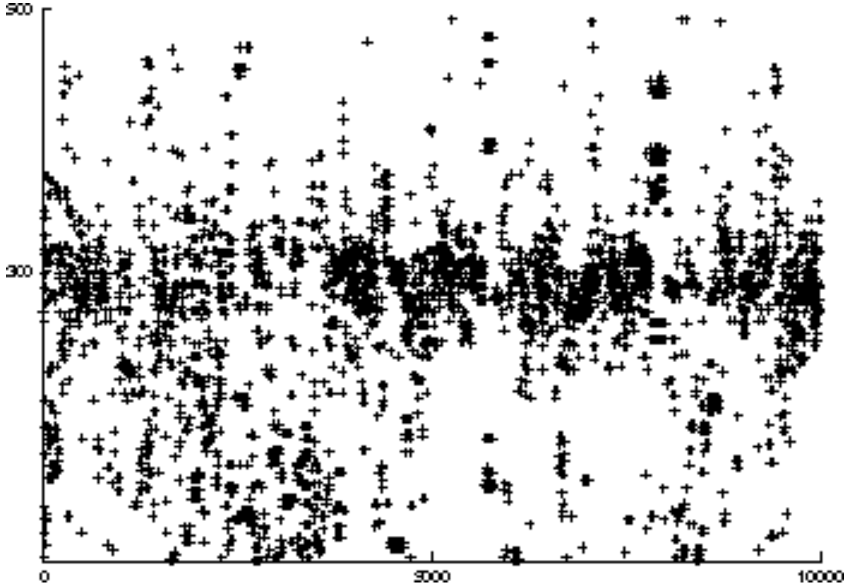
$$\hat{p}(.|x) = \mathcal{N}(h_\phi(x), R_\phi)$$

Raw Sensor Data

Measured distances for expected distance of 300 cm.

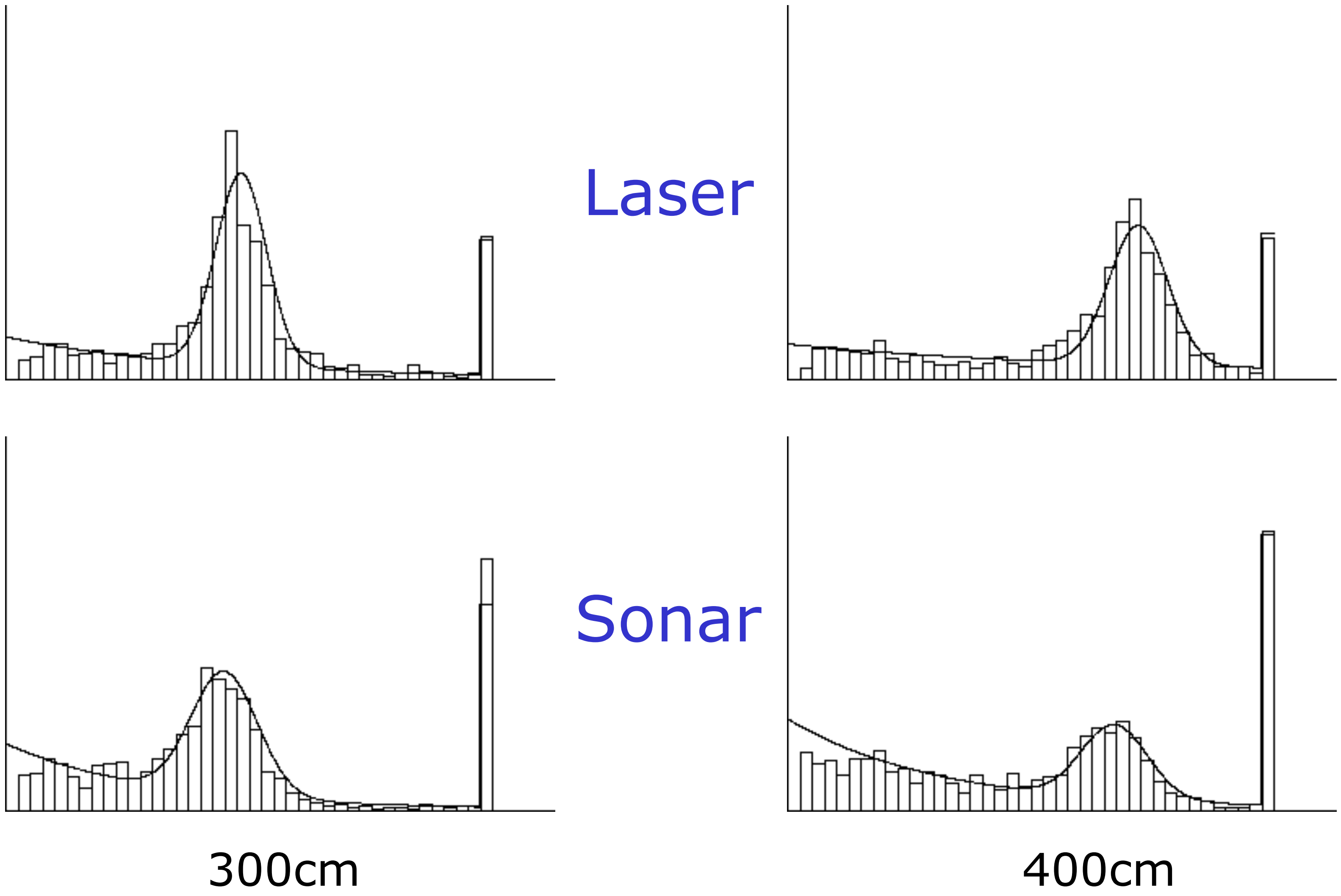


Sonar

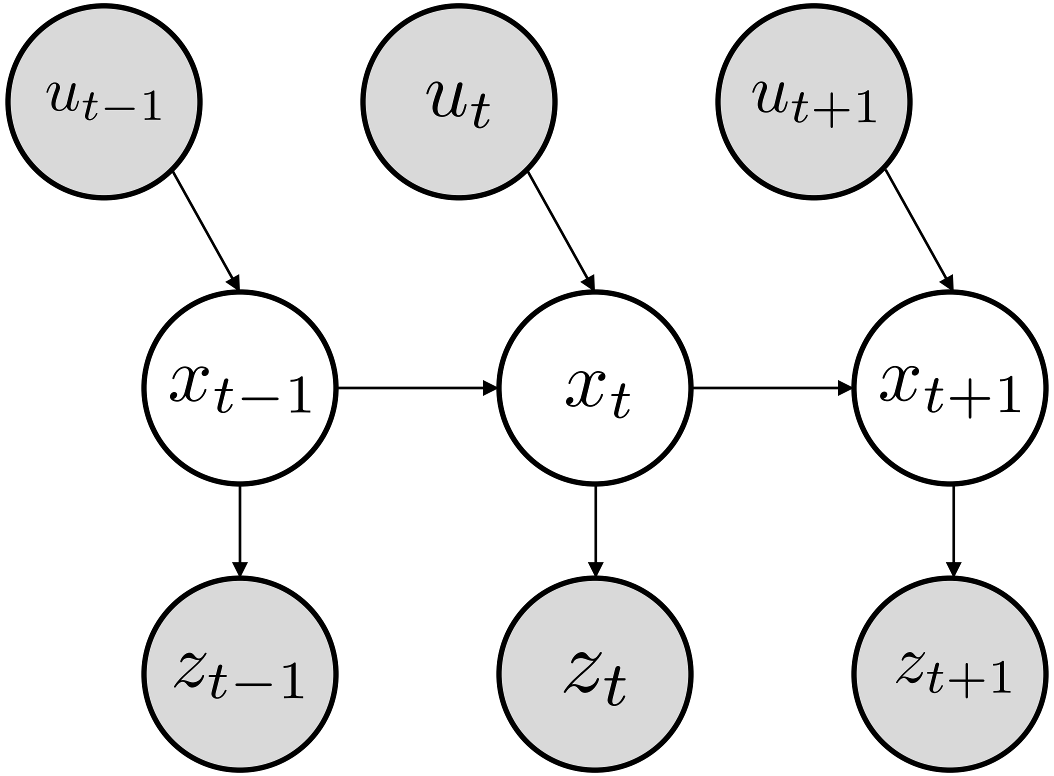


Laser

Approximation Results



Why is estimating parameters generally not so easy?

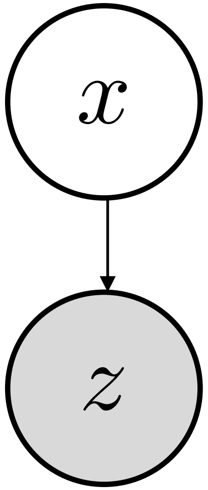


X is actually not observed typically, only z, u

Latent variable inference problem

Parameter Estimation in Latent Variable Models

Hard problem to solve exactly

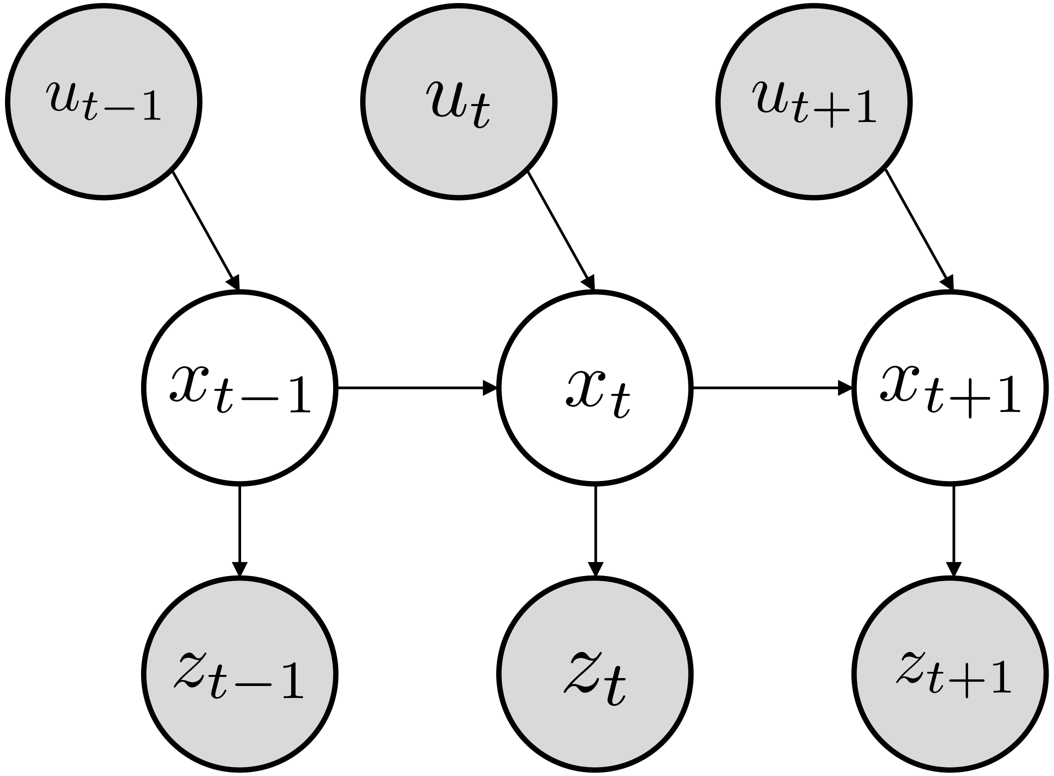


$$\max \log p(z)$$
$$= \log \int p(z|x)p(x)dx$$

Intractable problem

Solve via iterative optimization – Expectation Maximization algorithm (EM)
→ Much more general than filtering/localization

EM Algorithm for Latent Variable Parameter Estimation



$$\max \log p(z)$$

Approximate with a 2 step process:

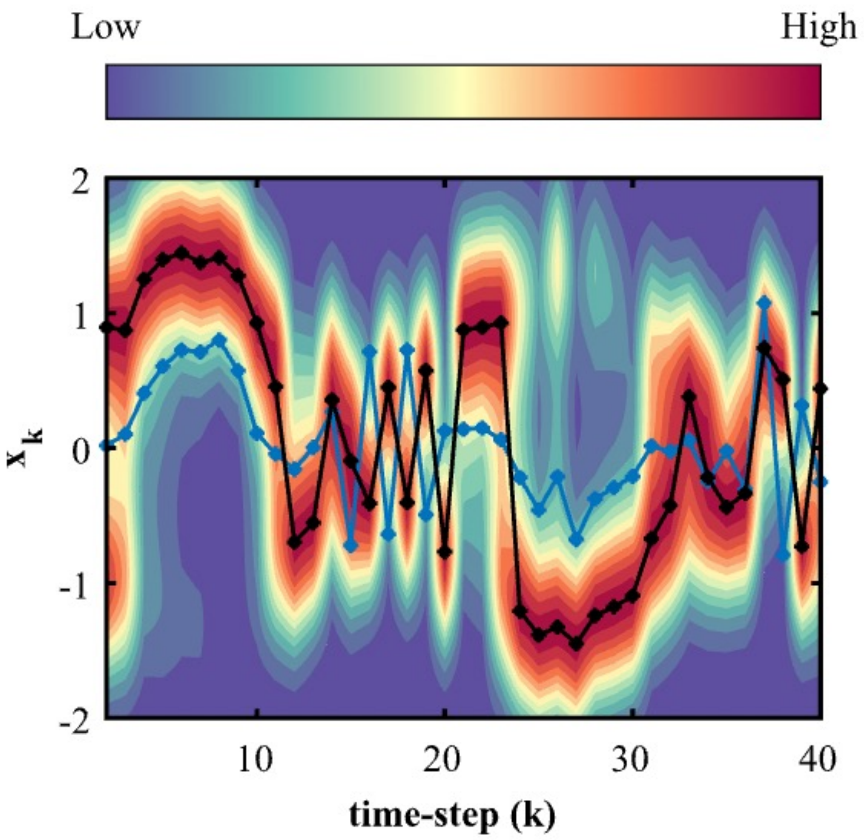
1. **E-step:** Fill in missing data x(i) according to what is most likely given the current model

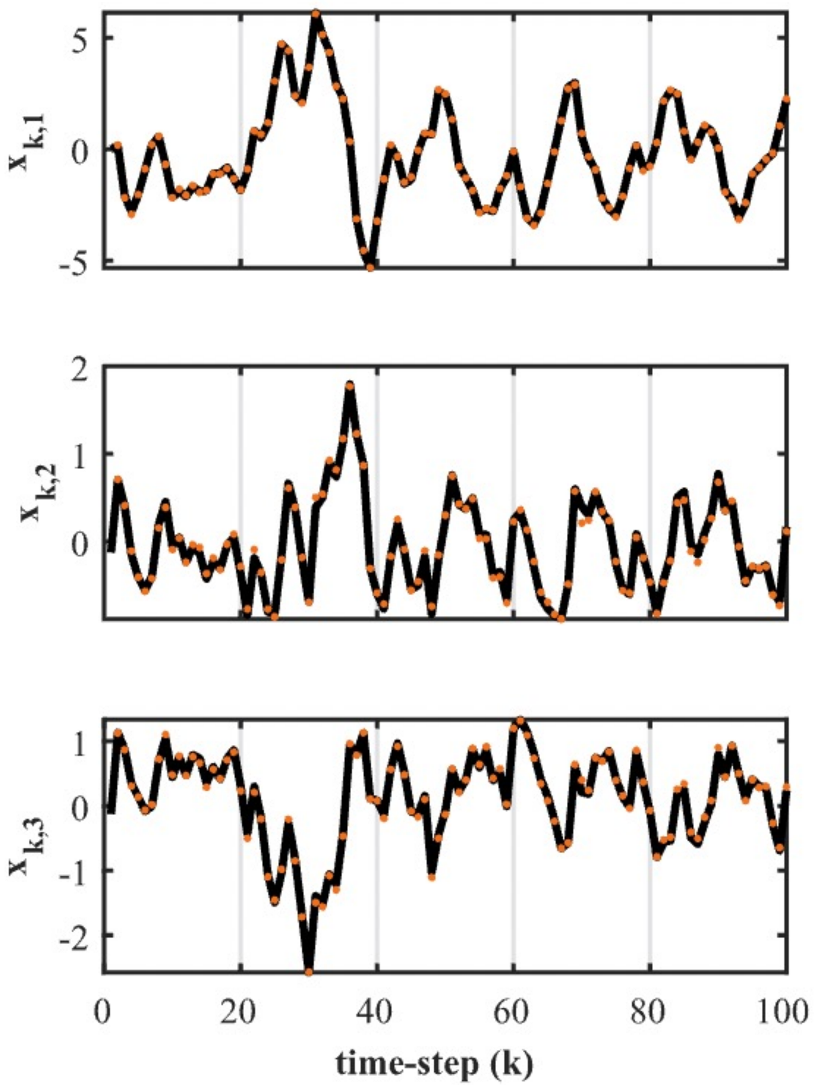
2. **M-step:** Run ML for completed data, which gives new model

Pretend we know the X

Do the best possible given inferred latents

EM Algorithms in Action for Estimating Motion/Sensor Models





Recap: Course Overview

Filtering/Smoothing
Localization
Mapping
SLAM

Search
Motion Planning
TrajOpt
Stability/Certification

MDPs and RL
Imitation Learning
Solving POMDPs

Lecture Outline

Sensor Models

↓

Parameter Estimation

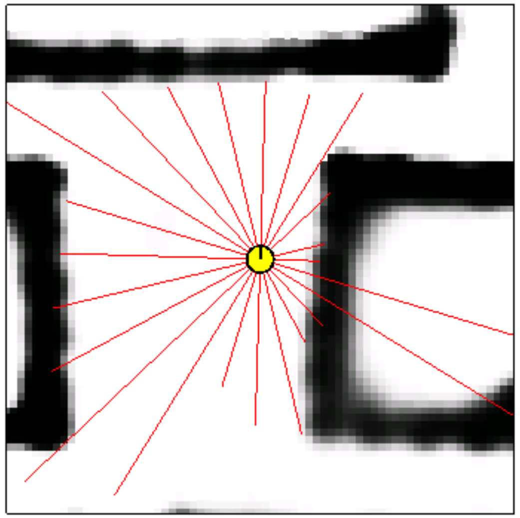
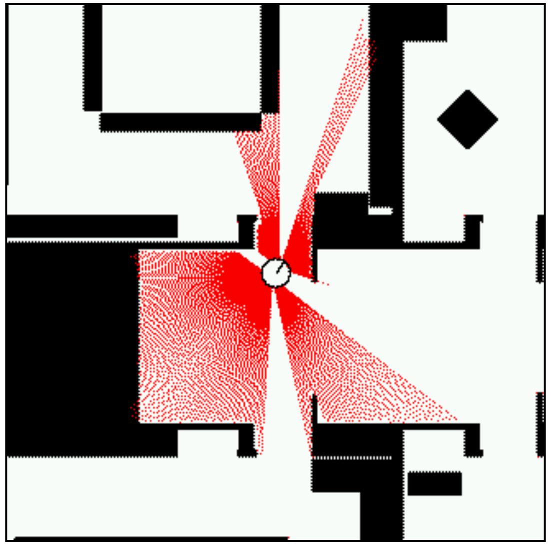
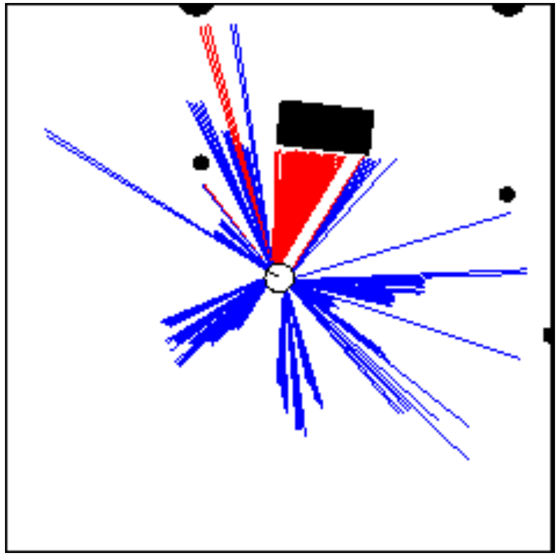
↓

Occupancy Mapping

What is mapping?

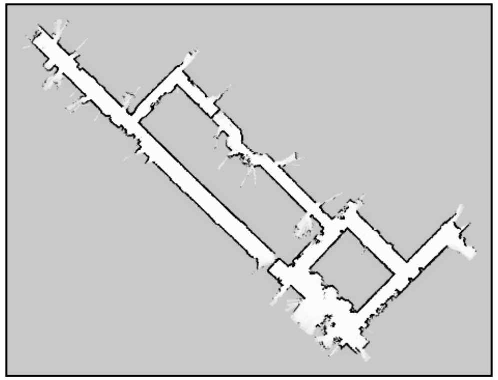
In all the localization examples thus far, the map m was assumed to be known

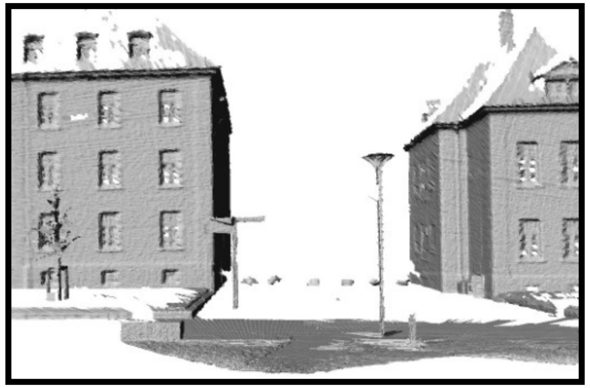
→ Not trivial in most environments

Types of Maps

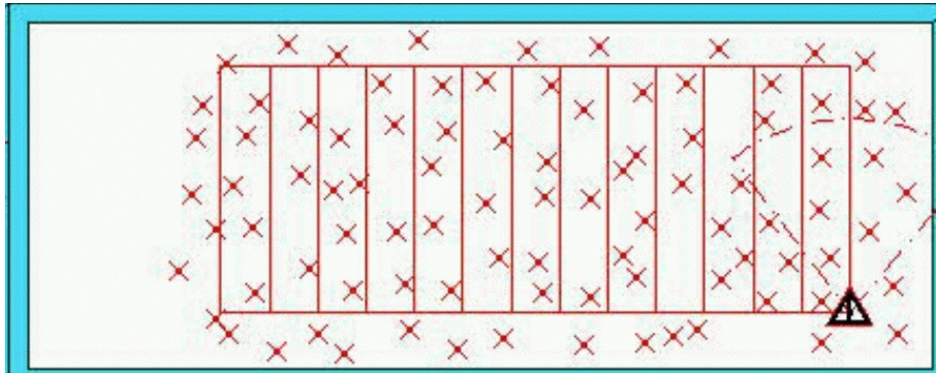
Grid maps or scans





Spatial 2-D or 3-D map:
→ Each grip cell has occupancy 0/1 and a signature

Sparse landmarks or RGB / Depth Maps



List of landmarks and their positions

Problems in Mapping

- Sensor interpretation
  - How do we extract relevant information from raw sensor data?
  - How do we represent and integrate this information over time?
- Robot locations have to be known
  - How can we estimate them during mapping?

Occupancy Grid Maps

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- Key assumptions
  - Occupancy of individual cells is independent

$$Bel(m_t) = P(m_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$
$$= \prod_{x,y} Bel(m_t^{[xy]})$$

  - Robot positions are known!

Updating Occupancy Grid Maps

- Idea: Update each individual cell using a binary Bayes filter.

$$Bel(m_t^{[xy]}) = \eta\ p(z_t \mid m_t^{[xy]}) \sum_{m_{t-1}^{[xy]}} p(m_t^{[xy]} \mid m_{t-1}^{[xy]}, u_{t-1}) Bel(m_{t-1}^{[xy]})$$

- Additional assumption: Map is static

$$Bel(m_t^{[xy]}) = \eta\ p(z_t \mid m_t^{[xy]}) Bel(m_{t-1}^{[xy]})$$

- What is a binary Bayes?
  - Random variable are binary, map is static
  - Tricks for numerical stability/efficiency

Binary Bayes Filter

Remember the Bayes Filter

$$Bel(x_t) = P(x_t | u_{0:t-1}, z_{0:t})$$
$$= \eta \, p(z_t | x_t) \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Let us just treat the variable x as binary! → occupied or not

Allows us to define a very simple, stable algorithm → filtering on m, not x

Express as Log-Odds

$$p(m_i | z_{1:t}, x_{1:t})$$
$$\Downarrow$$
$$\frac{p(m_i | z_{1:t}, x_{1:t})}{p(\neg m_i | z_{1:t}, x_{1:t})}$$

Recursion on Log-Odds

$$l_{t,i} = l_{t-1,i} + \log \frac{p(m_i | z_t, x_t)}{1 - p(m_i | z_t, x_t)} - \log \frac{p(m_i)}{1 - p(m_i)}$$

Binary Bayes Filter: Log Odds

Simple, numerically stable, efficient way to represent likelihoods

Odds:

$$\frac{p(x)}{p(\neg x)} = \frac{p(x)}{1 - p(x)}$$

Log Odds → makes products of odds additive.
Directly represent things in log space

$$\log \frac{p(x)}{1 - p(x)}$$

Easy conversion between log-odds and probs:

$$p(x) = 1 - \frac{1}{1 + \exp(l(x))}$$

Binary Bayes Filter: Recursive Update

Original Filtering problem: $p(m_i|z_{1:t}, x_{1:t})$

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, x_{1:t}, z_{1:t-1})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})}$$

Bayes Rule

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})}$$

Markov Property

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i|x_t)p(z_t|z_{1:t-1}, x_{1:t})}$$

Bayes Rule

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

Independence

$$p(\neg m_i|z_{1:t}, x_{1:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}$$

Same operations for negation

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_t, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})p(\neg m_i)}{p(\neg m_i|z_t, x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})p(m_i)}$$

Odds

Binary Bayes Filter: Recursive Update

Original Filtering problem: $p(m_i|z_{1:t}, x_{1:t})$

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_t, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})p(\neg m_i)}{p(\neg m_i|z_t, x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})p(m_i)}$$

$$\log \frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} = \log \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} + \log \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i|z_{1:t-1}, x_{1:t-1})} + \log \frac{p(\neg m_i)}{p(m_i)}$$

$$l_{t,i} = \log \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} + l_{t-1,i} + \log \frac{p(\neg m_i)}{p(m_i)}$$

Simple recursive update!

Inverse measurement model | Previous log odds | Prior

Can recover map likelihood per cell from here

→ Likelihood of an entire map is the product of individual grid likelihoods

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t})$$

Inverse Sensor Model for Occupancy Grid Maps

- Predict map likelihood from z, x - $p(m_i | z_t, x_t)$

Algorithm inverse_range_sensor_model(i, x_t, z_t)**:**

Let x_i, y_i be the center-of-mass of $\mathbf{m}_i$

$r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

$\phi = \text{atan2}(y_i - y, x_i - x) - \theta$

$k = \text{argmin}_j \, |\phi - \theta_{j,\text{sens}}|$

if $r > \min(z_{\max}, z_t^k + \alpha/2)$ or $|\phi - \theta_{k,\text{sens}}| > \beta/2$ then

    return l_0

if $z_t^k < z_{\max}$ and $|r - z_{\max}| < \alpha/2$

    return l_{occ}

if $r \leq z_t^k$

    return l_{free}

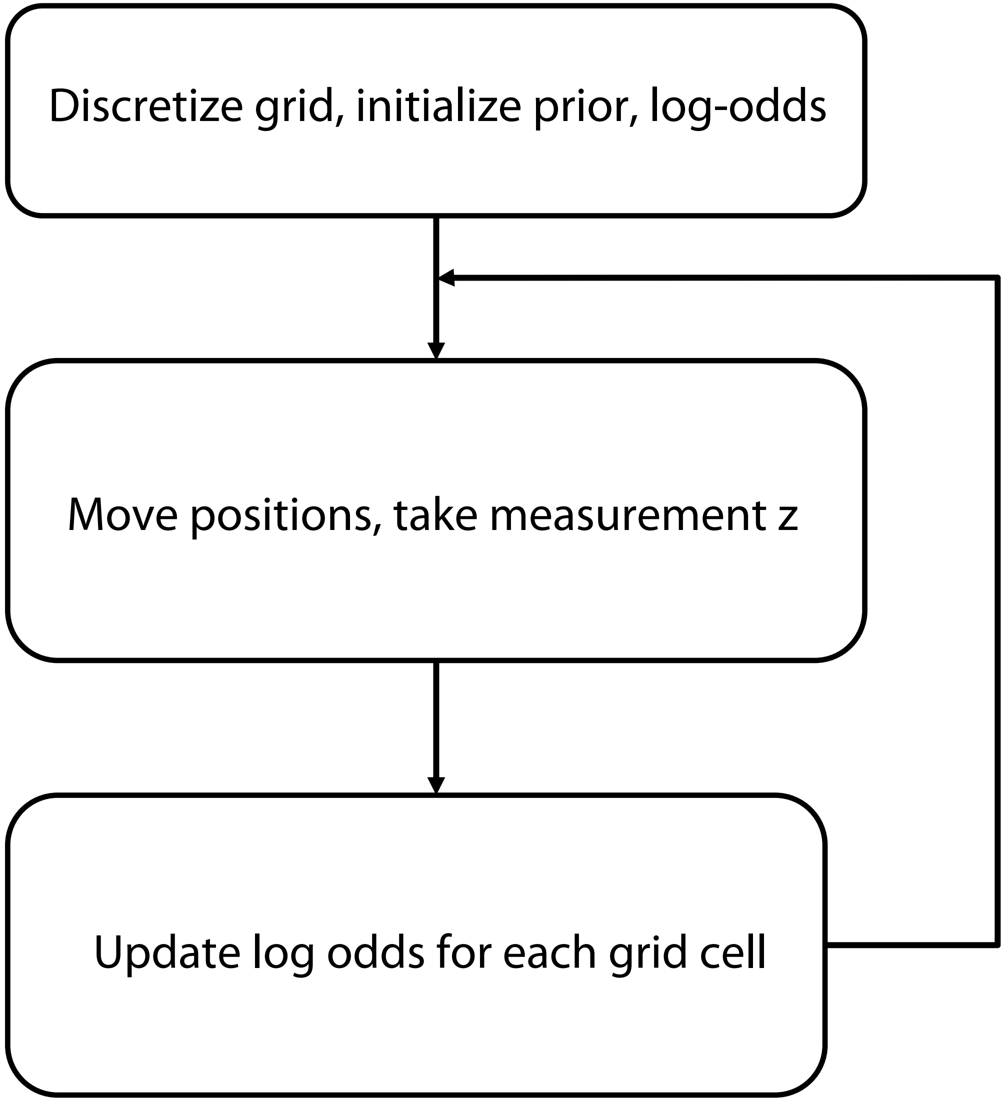
endif

Combined Occupancy Mapping Algorithm

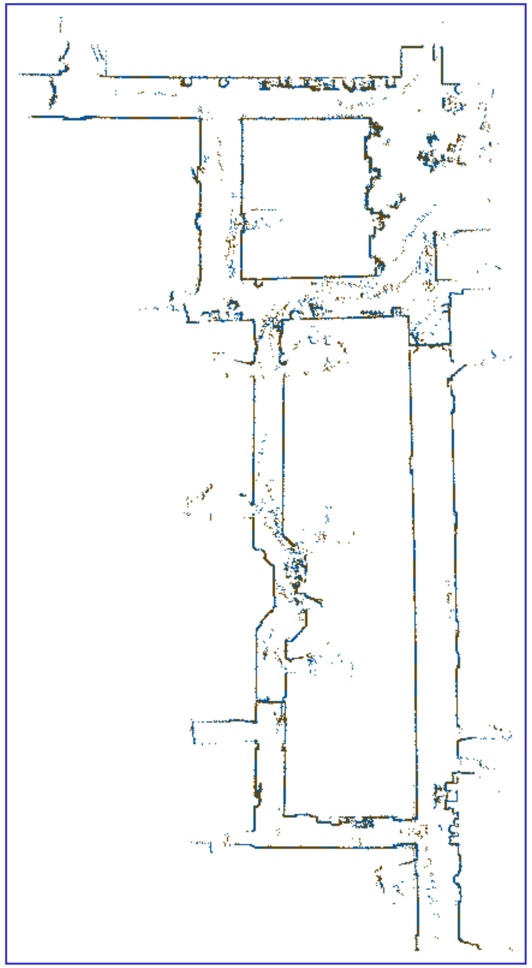
$$p(m_i)$$

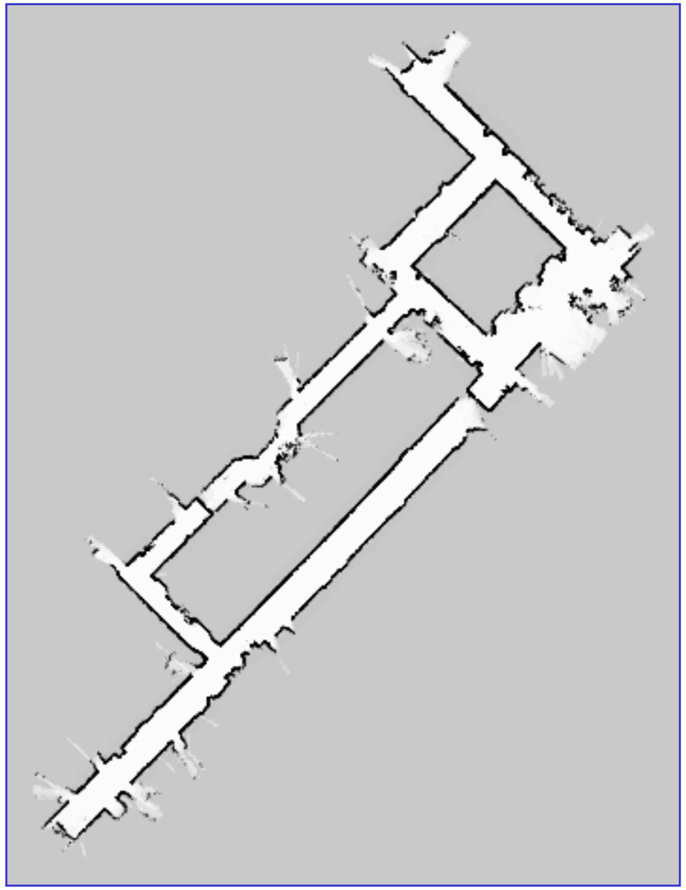
$$p(m_i|z_t, x_t)$$

$$l_{t,i} = \log \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} + l_{t-1,i} + \log \frac{p(\neg m_i)}{p(m_i)}$$



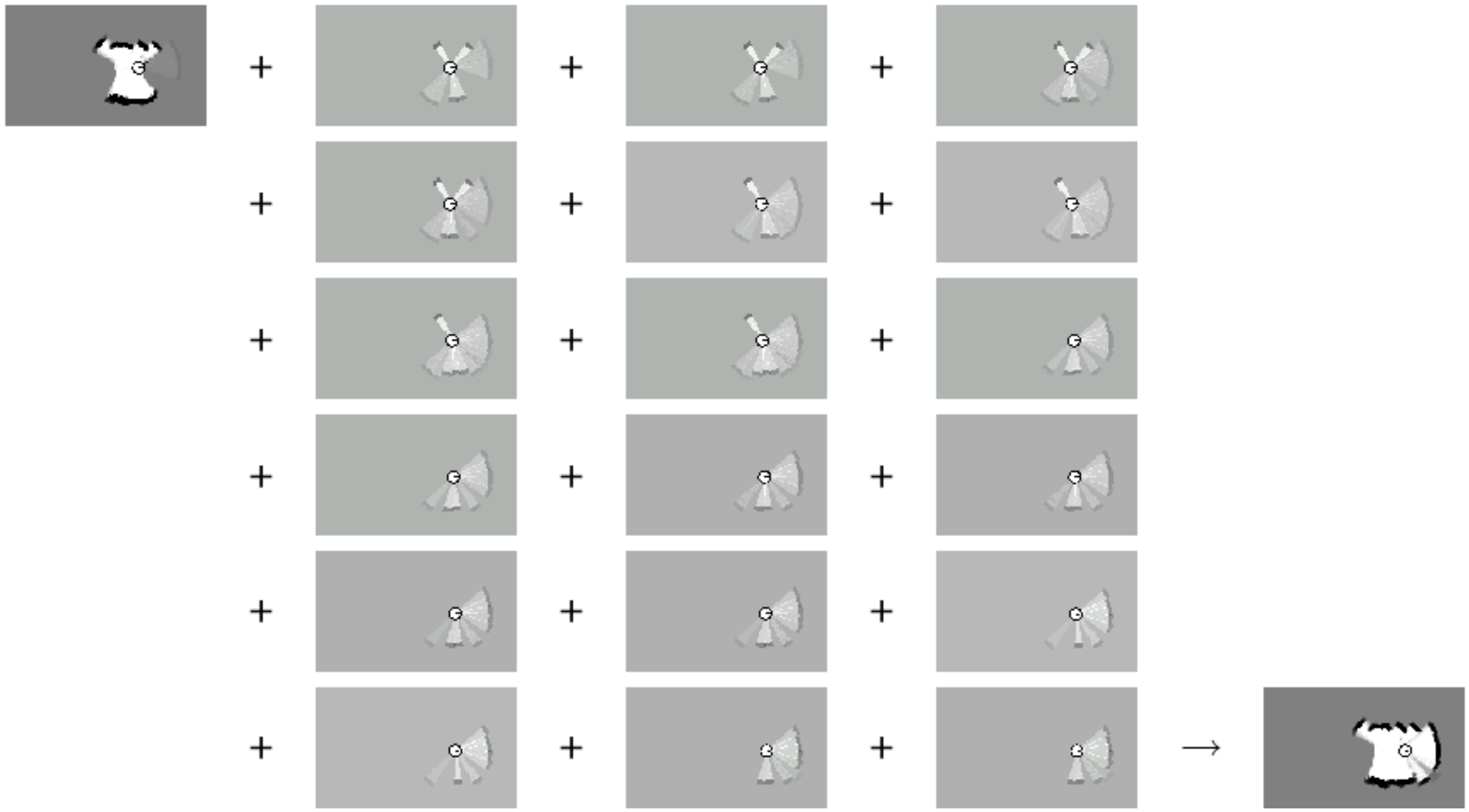
Occupancy Grids: From scans to maps



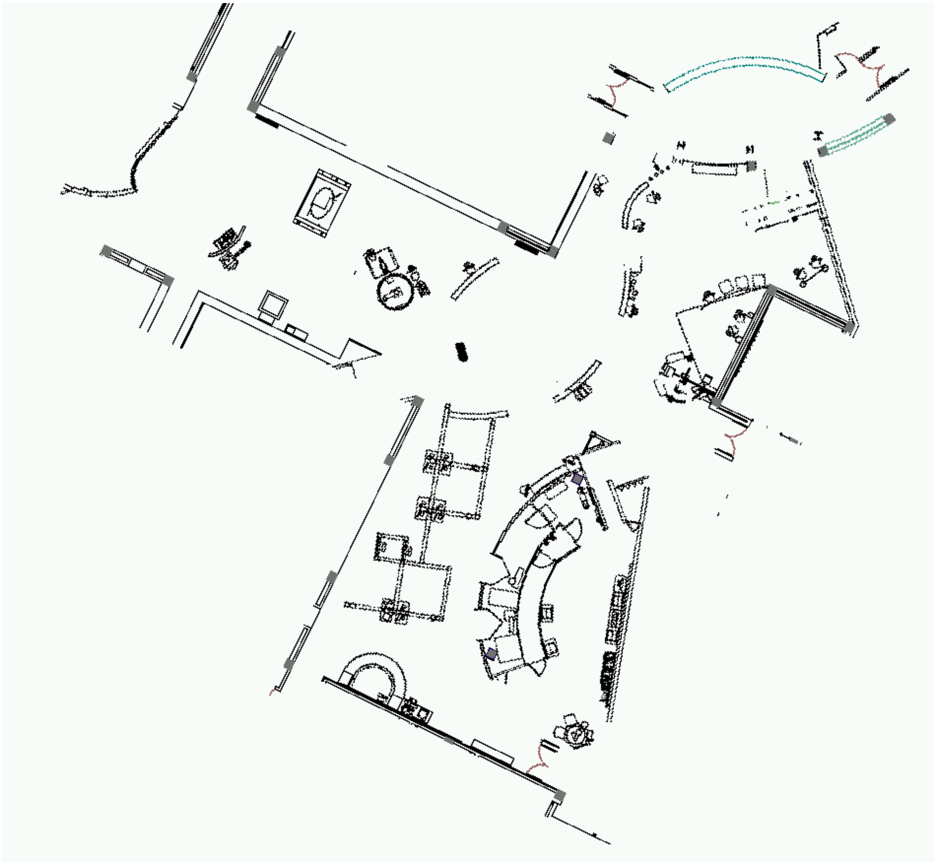


Much less random noise, more traversable!

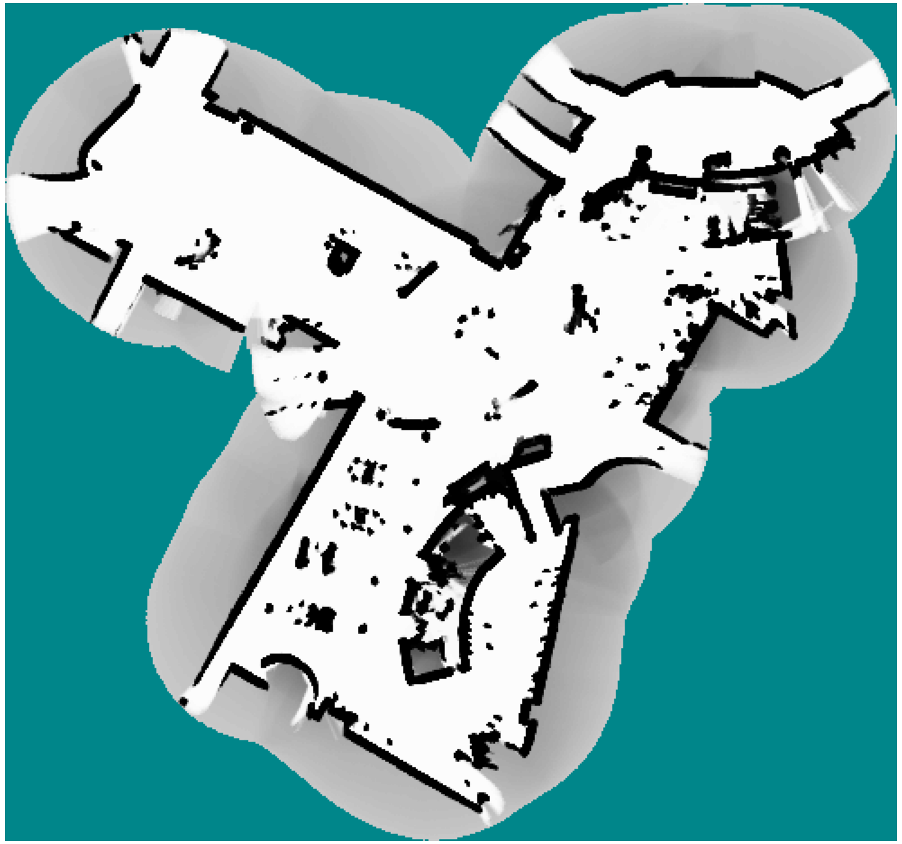
Incremental Updating of Occupancy Grids (Example)



Tech Museum, San Jose



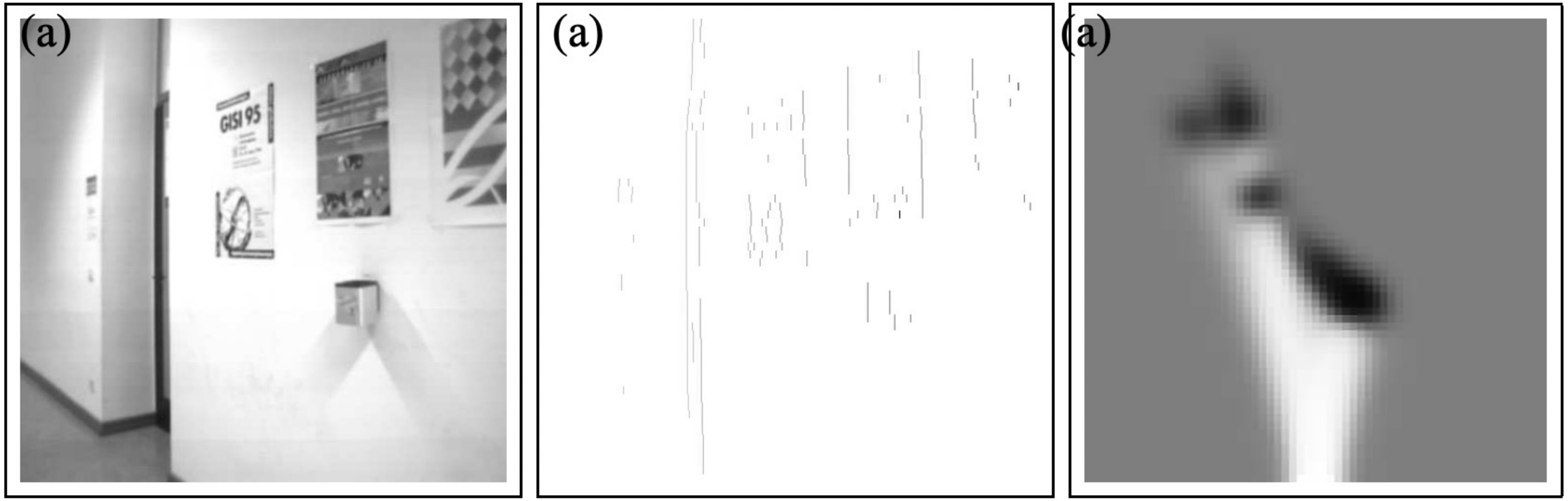
CAD map



occupancy grid map

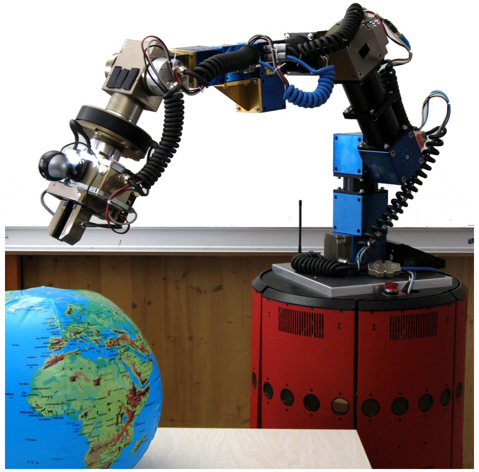
Multi-sensor Fusion

Occupancy maps produced by every sensor may be different



Construct most pessimistic estimate $\mathbf{m}_i = \max_k \mathbf{m}_i^k$

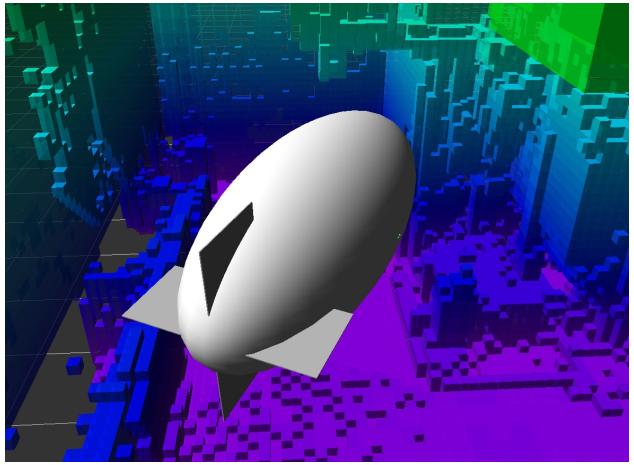
Robots in 3D Environments



Mobile manipulation

Outdoor navigation

Humanoid robots



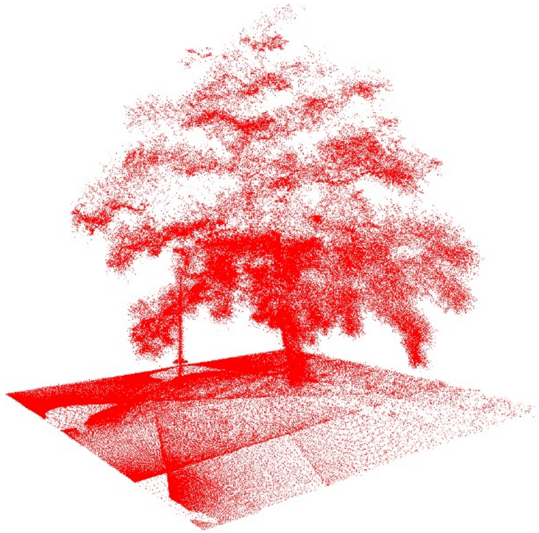
Flying robots

3D Map Requirements

- Full 3D Model
  - Volumetric representation
  - Free-space
  - Unknown areas (e.g. for exploration)
- Can be updated
  - Probabilistic model (sensor noise, changes in the environment)
  - Update of previously recorded maps
- Flexible
  - Map is dynamically expanded
  - Multi-resolution map queries
- Compact
  - Memory efficient
  - Map files for storage and exchange

Map Representations: Pointclouds

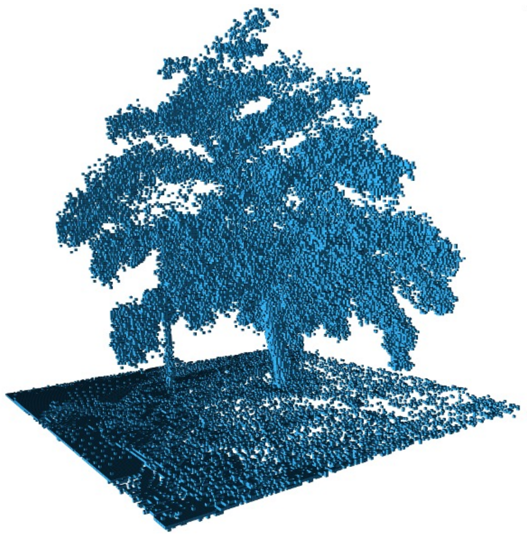
- Pro:
  - No discretization of data
  - Mapped area not limited



- Cons:
  - Unbounded memory usage
  - No direct representation of free or unknown space

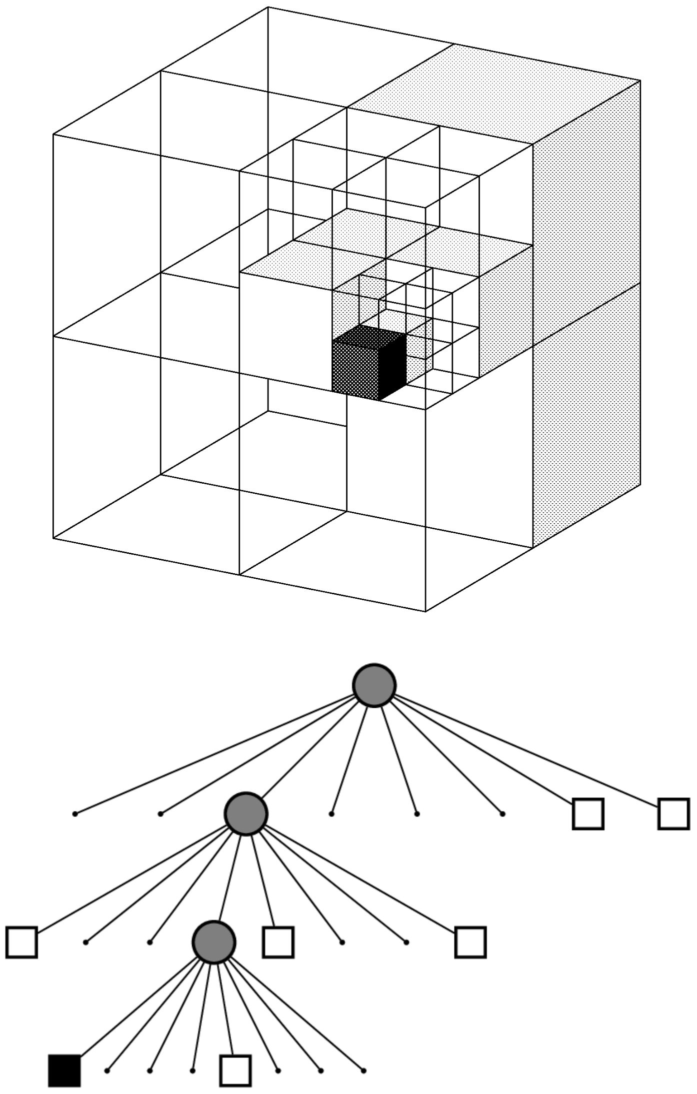
Map Representations: 3D voxel grids

- Pros:
  - Probabilistic update
  - Constant access time
- Cons:
  - Memory requirement
    - Extent of map has to be known
    - Complete map is allocated in memory



Map Representations: Octrees

- Tree-based data structure
- Recursive subdivision of space into octants
- Volumes allocated as needed
- Multi-resolution



OctoMap

A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems

K.M. Wurm, *A. Hornung*,

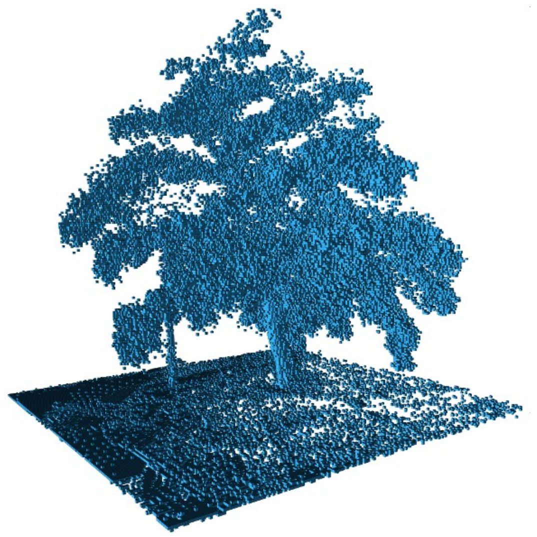
M. Bennewitz, C. Stachniss, W. Burgard

University of Freiburg, Germany

http://octomap.sf.net

Map Representations: Octrees

- Pro:
  - Full 3D model
  - Probabilistic
  - Flexible, multi-resolution
  - Memory efficient



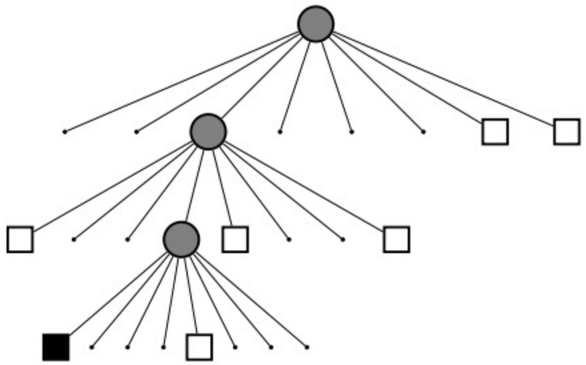
- Cons:
  - Implementation can be tricky (memory, update, map files, …)
  - Open source implementation as C++ library available at http://octomap.sf.net

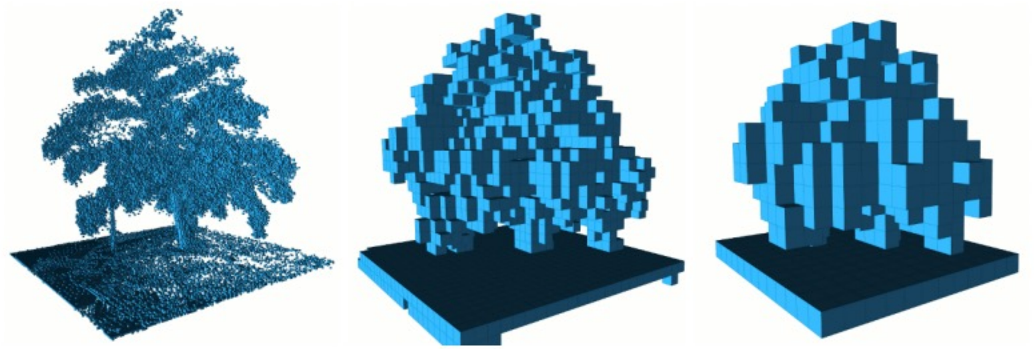
Probabilistic Map Update

OctoMap:
An Efficient Probabilistic 3D Mapping Framework Based on Octrees

Armin Hornung · Kai M. Wurm · Maren Bennewitz ·
Cyrill Stachniss · Wolfram Burgard

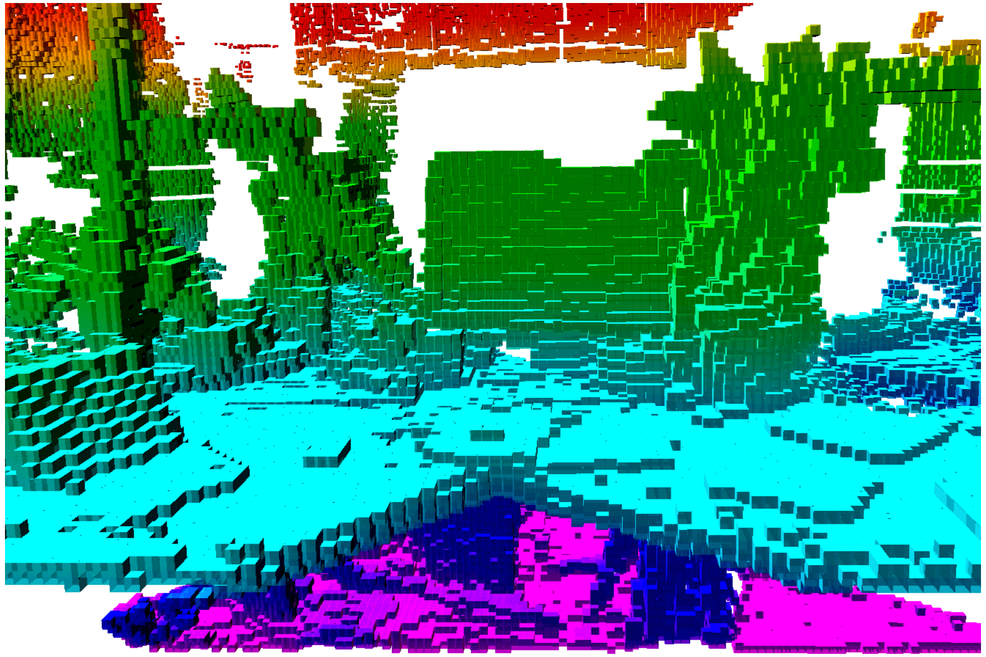
- Perform standard log-odds update on 3-D map
  - Clamping updates
  - Multi-resolution queries on non-leaf nodes





Examples

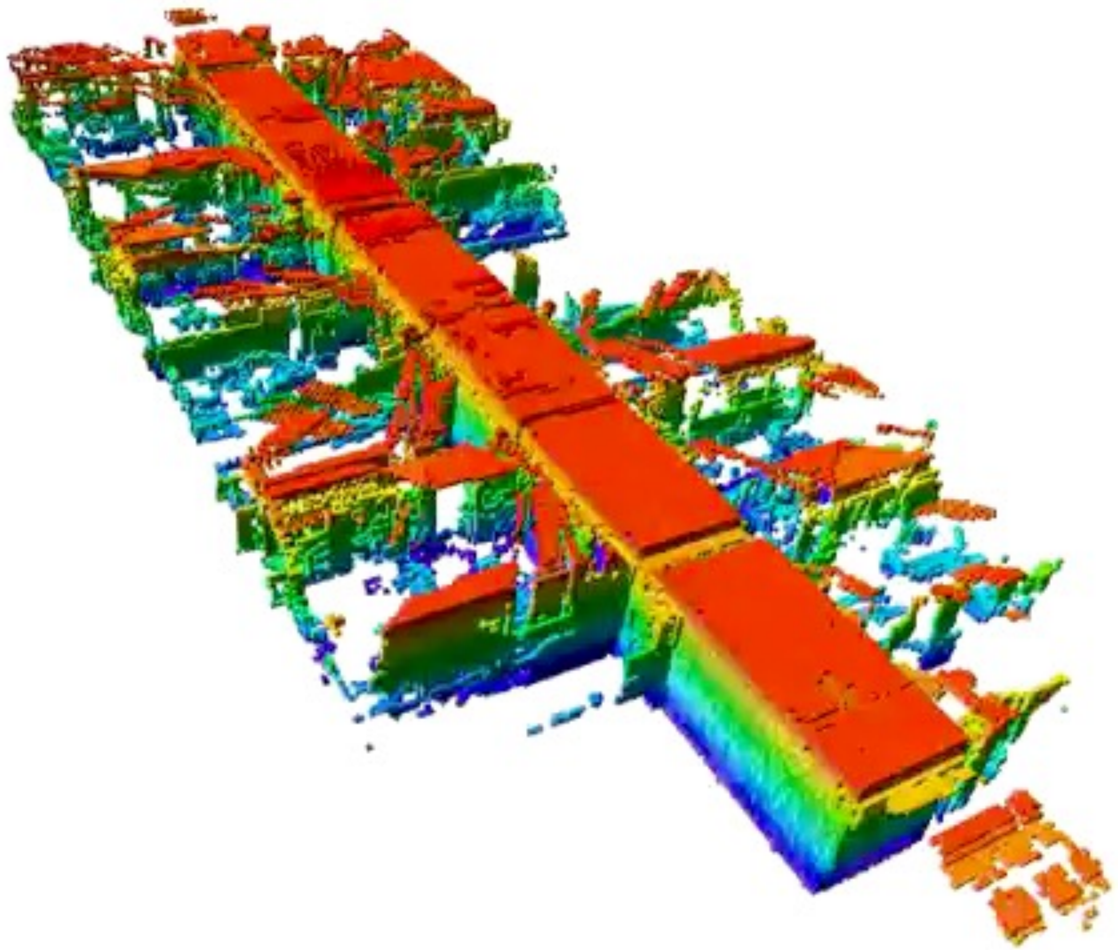
- Cluttered office environment



Map resolution: 2 cm

Examples: Office Building

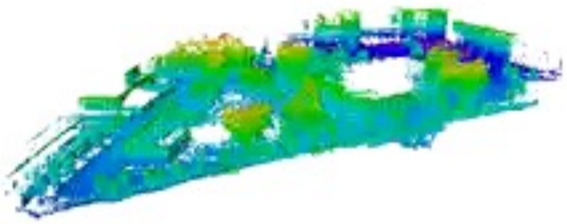
- Freiburg, building 079



Examples: Large Outdoor Areas

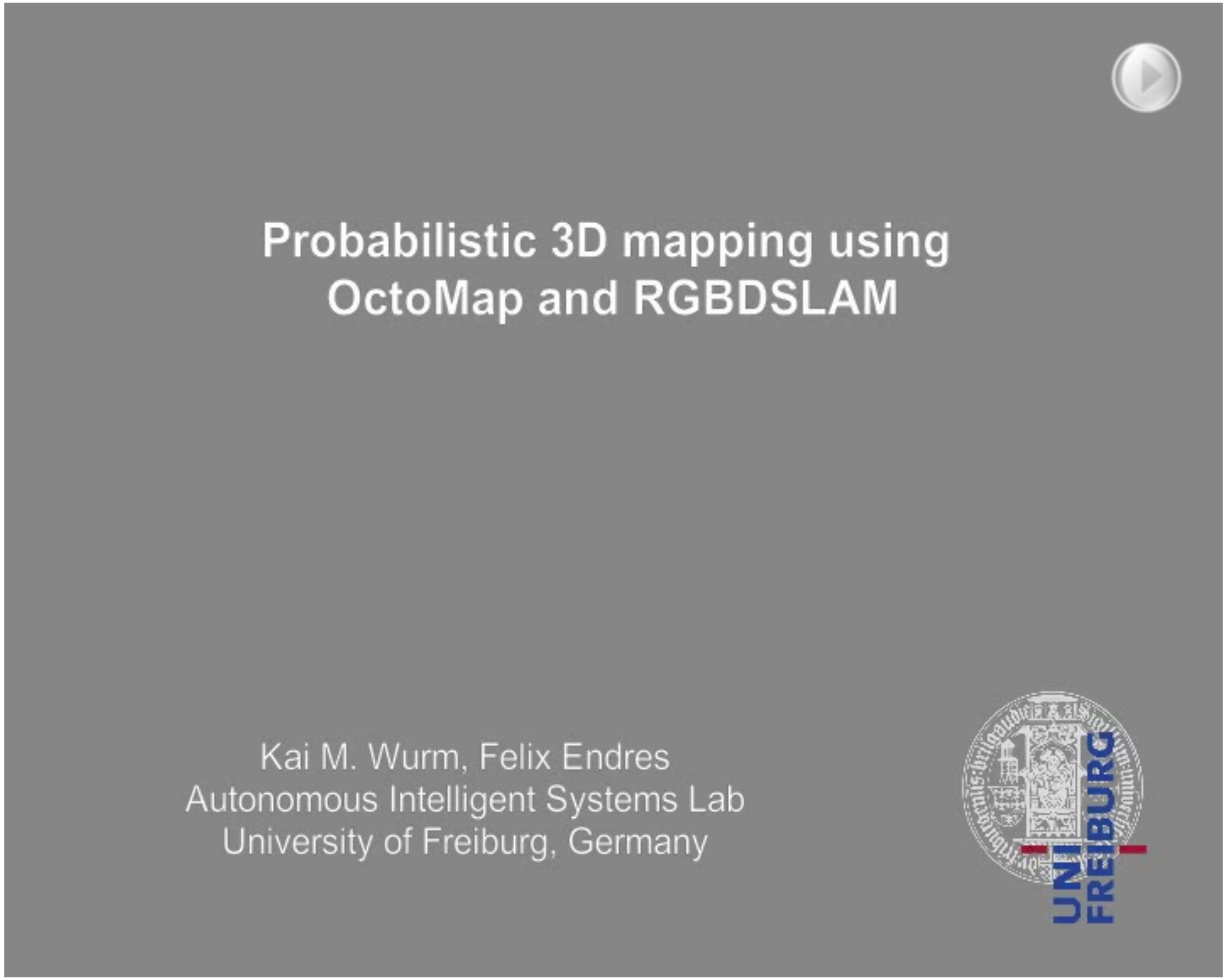
- Freiburg computer science campus

  (292 x 167 x 28 m^3, 20 cm resolution)



Examples: Tabletop

Adding Color



Lecture Outline

Sensor Models

↓

Parameter Estimation

↓

Occupancy Mapping