Development of intelligent systems (RInS)

Mobile robotics

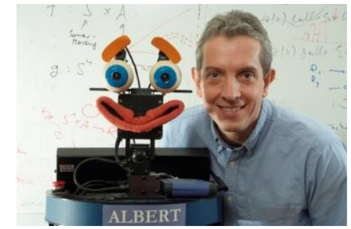
Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

Slides: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Kai Arras, UNI Freiburg, *Introduction to Mobile Robotics*

Academic year: 2021/22

Introduction

- Slides credit to:



Autonomous Intelligent Systems
Wolfram Burgard

Autonomous Intelligent Systems
Cyrill Stachniss

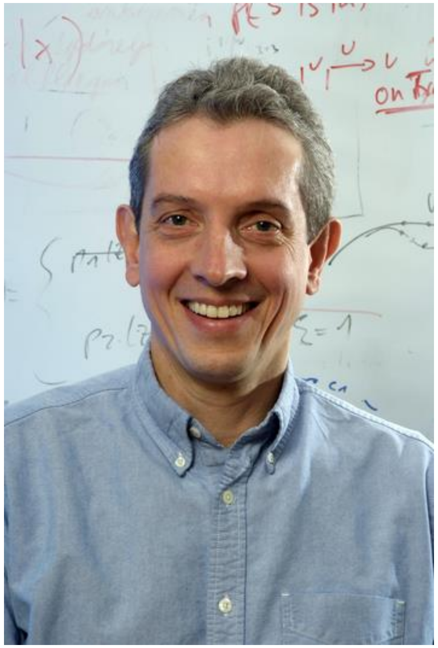
Humanoid Robots
Maren Bennewitz

Social Robotics
Kai Arras

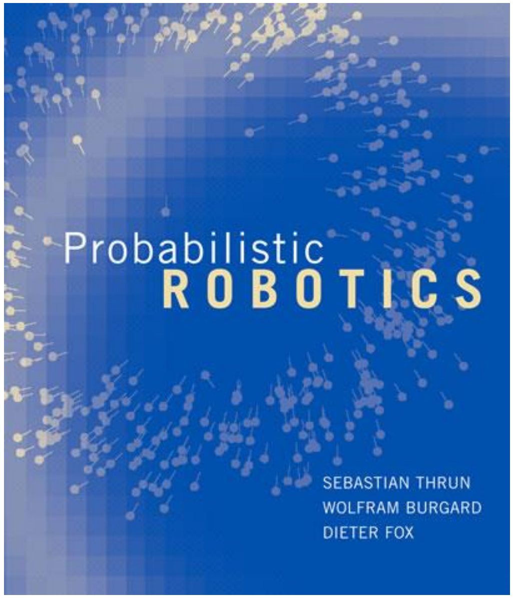
Albert-Ludwigs-Universität, Freiburg, Germany

Introduction

- Wolfram Burgard,
  Albert-Ludwigs-Universität Freiburg



- Sebastian Thrun, Wolfram Burgard and Dieter Fox,Probabilistic Robotics, The MIT Press, 2005



Introduction

- Course *Introduction to Mobile Robotics* – Autonomous Mobile Systems at the Albert-Ludwigs-Universität Freiburg

Introduction to Mobile Robotics (engl.) - Autonomous Mobile Systems

This course will introduce basic concepts and techniques used within the field of mobile robotics. We analyze the fundamental challenges for autonomous intelligent systems and present the state of the art solutions. Among other topics, we will discuss:

- Sensors,
- Kinematics,
- Path planning,
- Vehicle localization,
- Map building,
- SLAM,
- Exploration of unknown terrain

Introduction

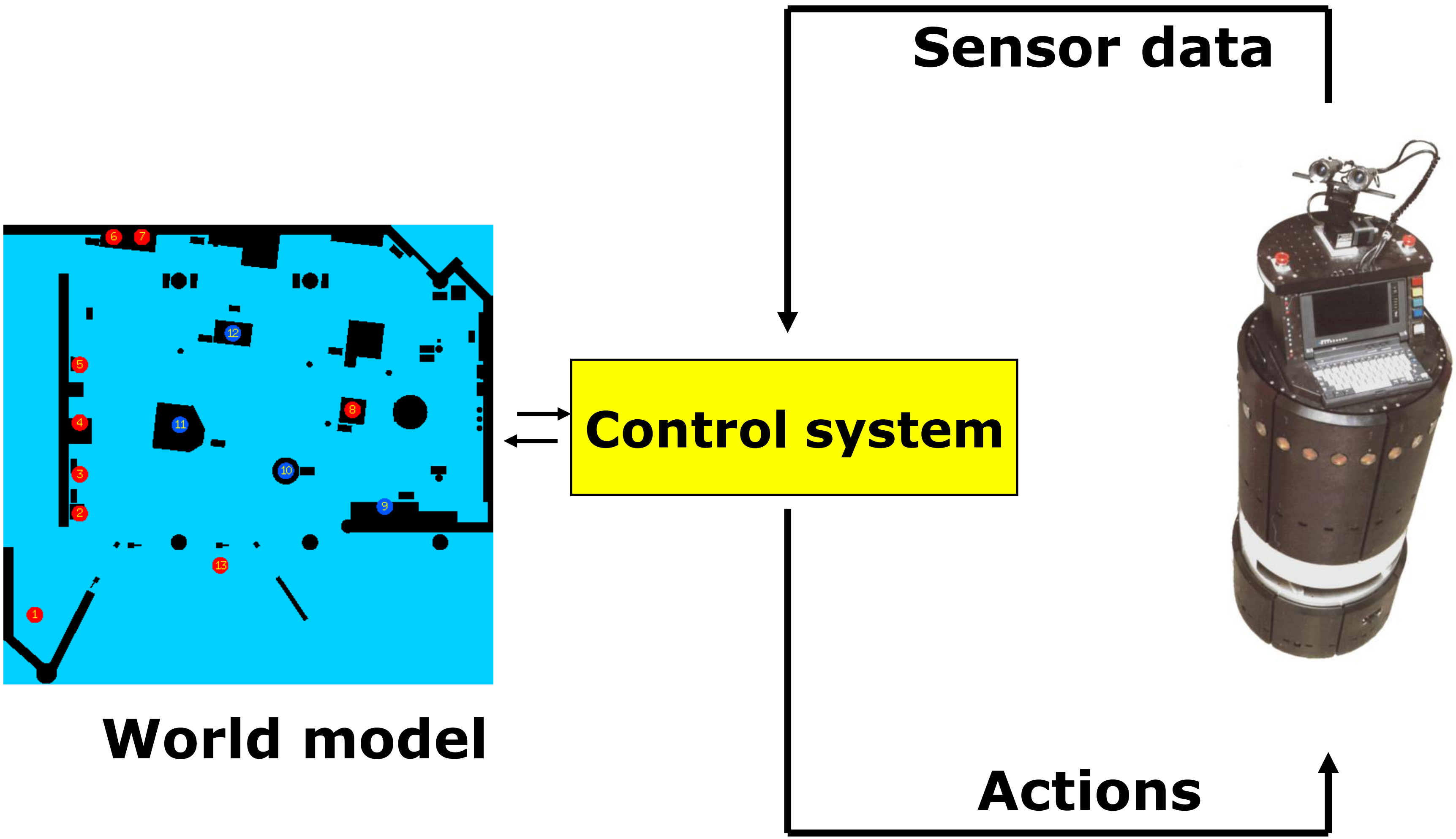
- Course *Introduction to Mobile Robotics* – Autonomous Mobile Systems at the Albert-Ludwigs-Universität Freiburg
- This course:

- Introduction PDF
- Robot Control Paradigms PDF
- Linear Algebra PDF
- Wheeled Locomotion PDF
- Proximity Sensors PDF
- Probabilistic Robotics PDF
- Motion Models PDF
- Sensor Models PDF
- Error Propagation PDF
- LSQ Estimation, Geometric Feature Extraction PDF
- Kalman Filter PDF
- Discrete Filter PDF
- Particle Filter PDF
- Mapping with Known Poses PDF
- EKF Localization PDF
- SLAM PDF
- Landmark-based FastSLAM PDF
- Grid-based FastSLAM PDF
- ICP: Iterative Closest Point Algorithm PDF
- Multi-Robot Exploration PDF
- Information Gain-Based Exploration PDF
- 3D Mapping PDF
- Robot Motion Planning PDF (updated 26.07.2011)

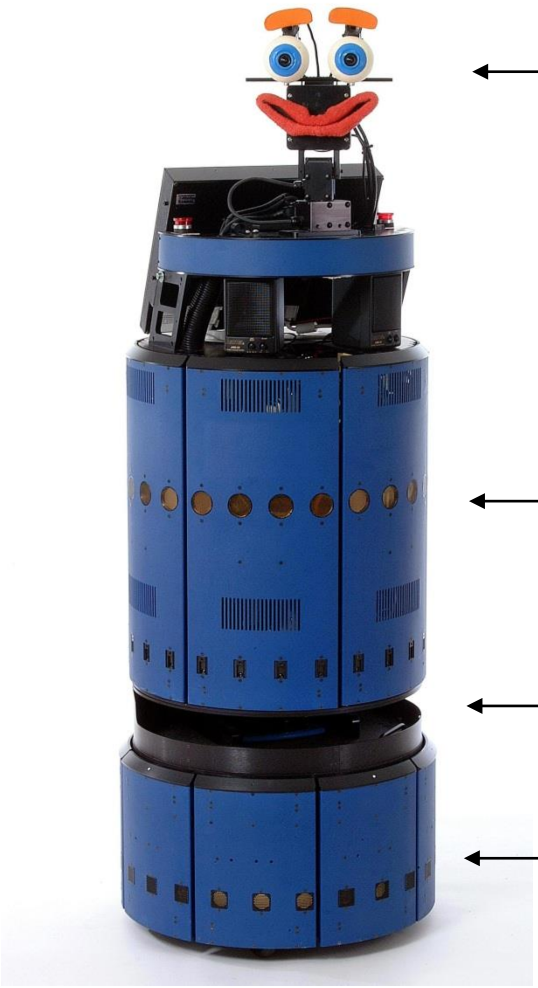
Goal of this course

- Provide an overview of problems / approaches in mobile robotics
- Probabilistic reasoning: Dealing with noisy data
- Hands-on experience

AI View on Mobile Robotics



Components of Typical Robots



cameras

sonars

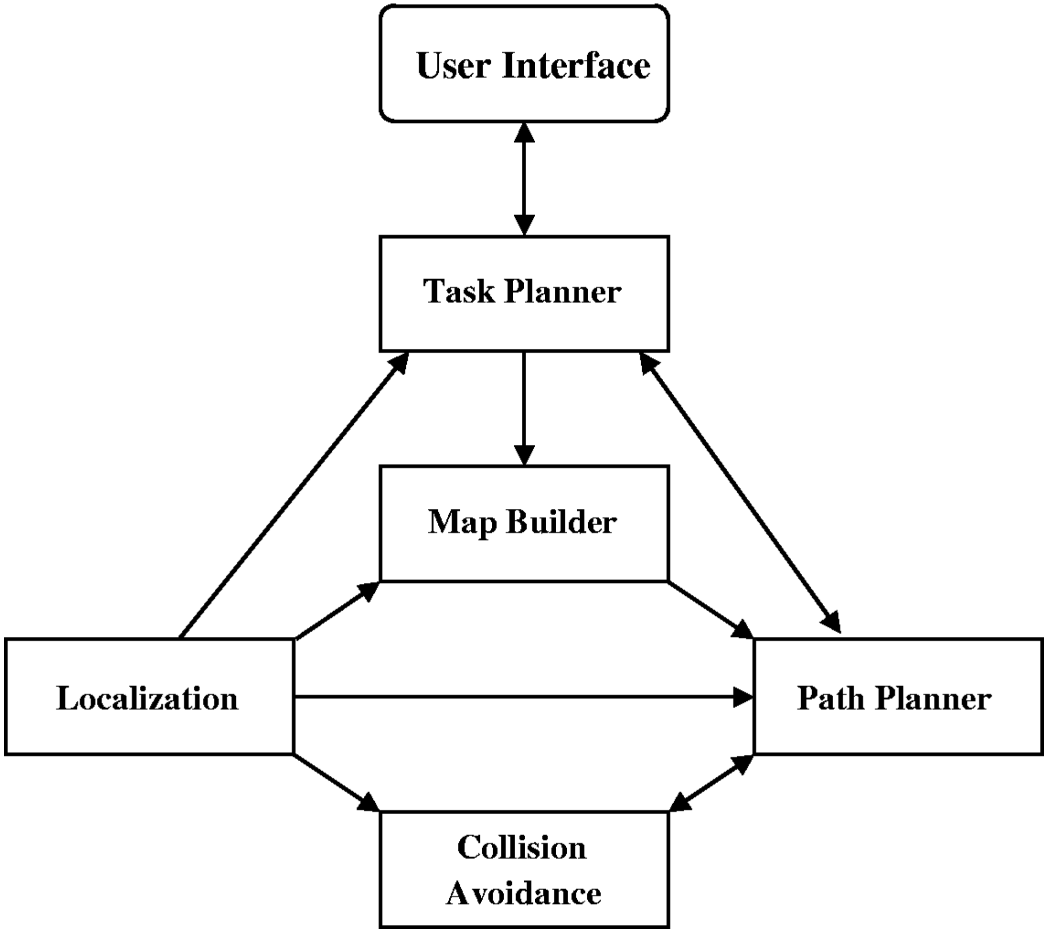
laser

sensors

base

actuators

Architecture of a Typical Control System



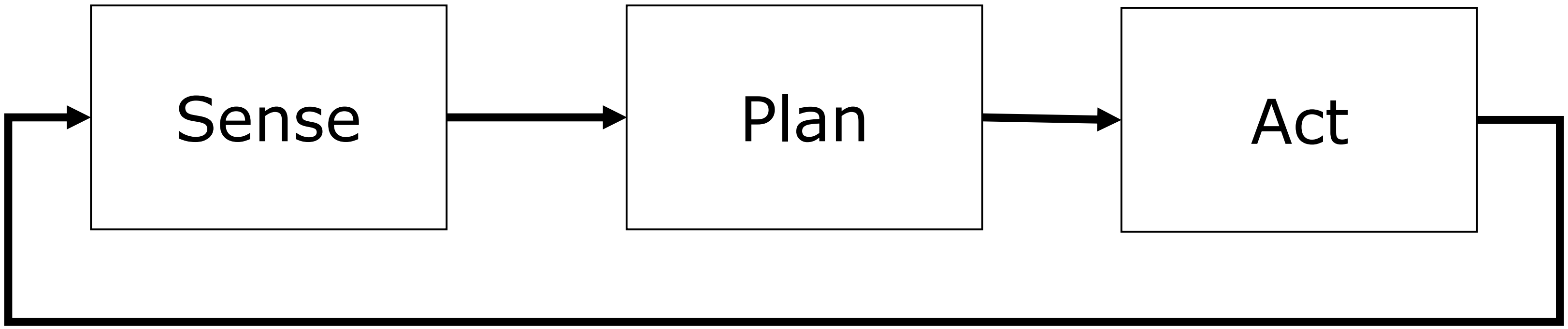
Introduction to Mobile Robotics

Robot Control Paradigms

Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

Classical / Hierarchical Paradigm



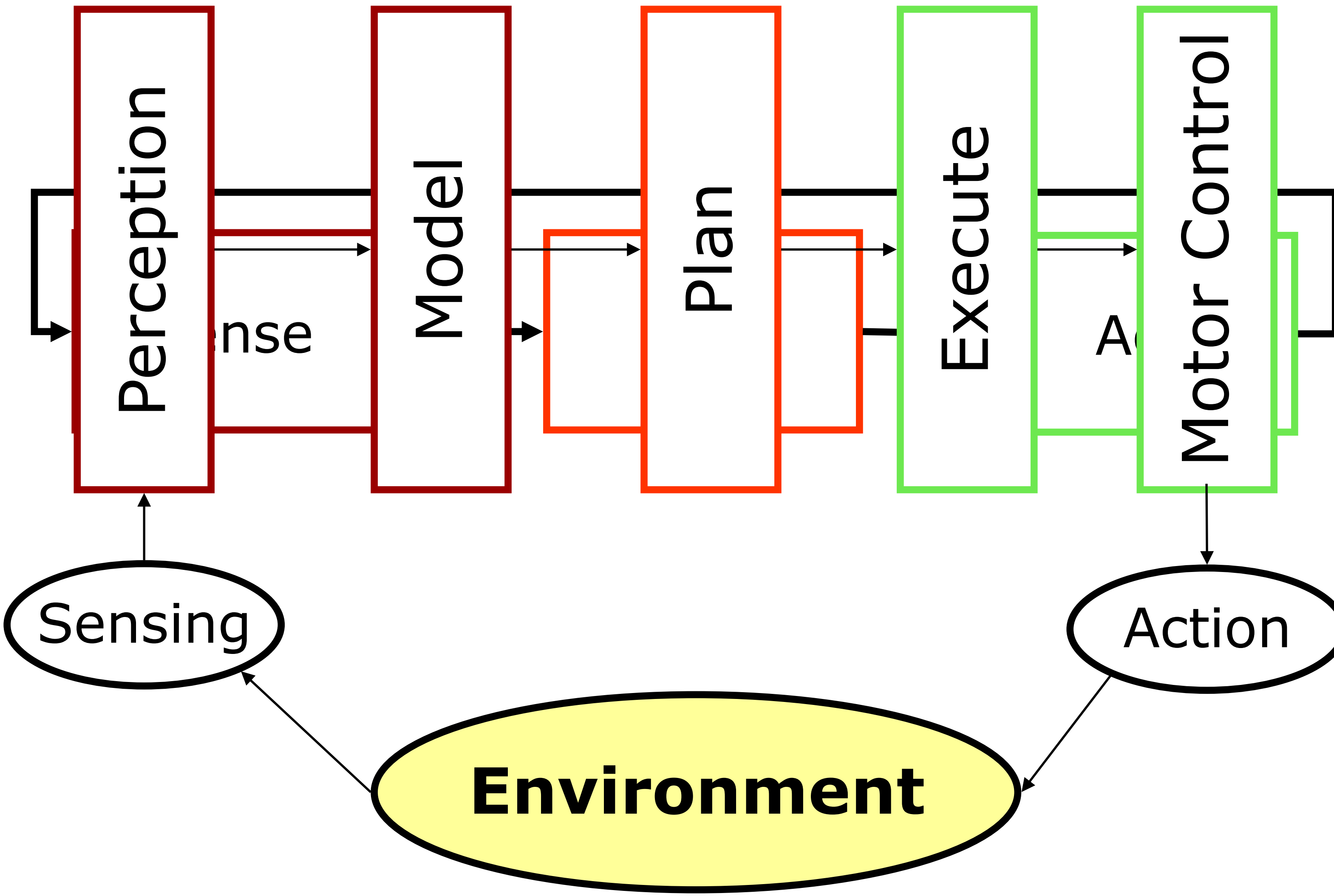
- 70's
- Focus on automated reasoning and knowledge representation
- STRIPS (Stanford Research Institute Problem Solver): Perfect world model, closed world assumption
- Find boxes and move them to designated position

Classical Paradigm
Stanford Cart

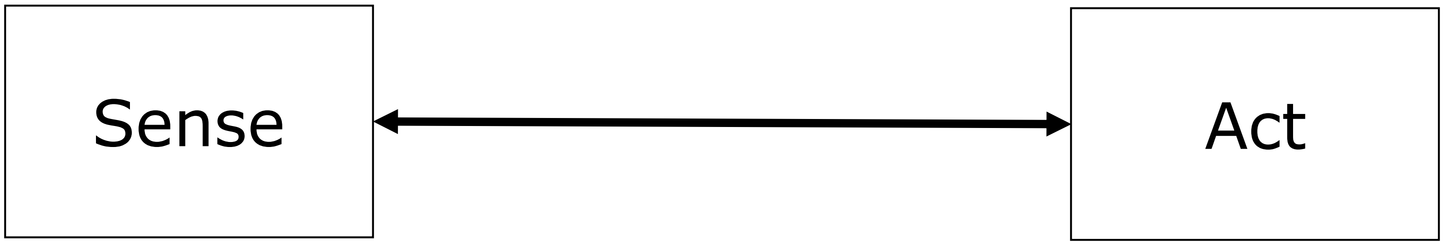
1. Take nine images of the environment, identify interesting points in one image, and use other images to obtain depth estimates.
2. Integrate information into global world model.
3. Correlate images with previous image set to estimate robot motion.
4. On basis of desired motion, estimated motion, and current estimate of environment, determine direction in which to move.
5. Execute the motion.

Stanford Cart

Classical Paradigm as Horizontal/Functional Decomposition

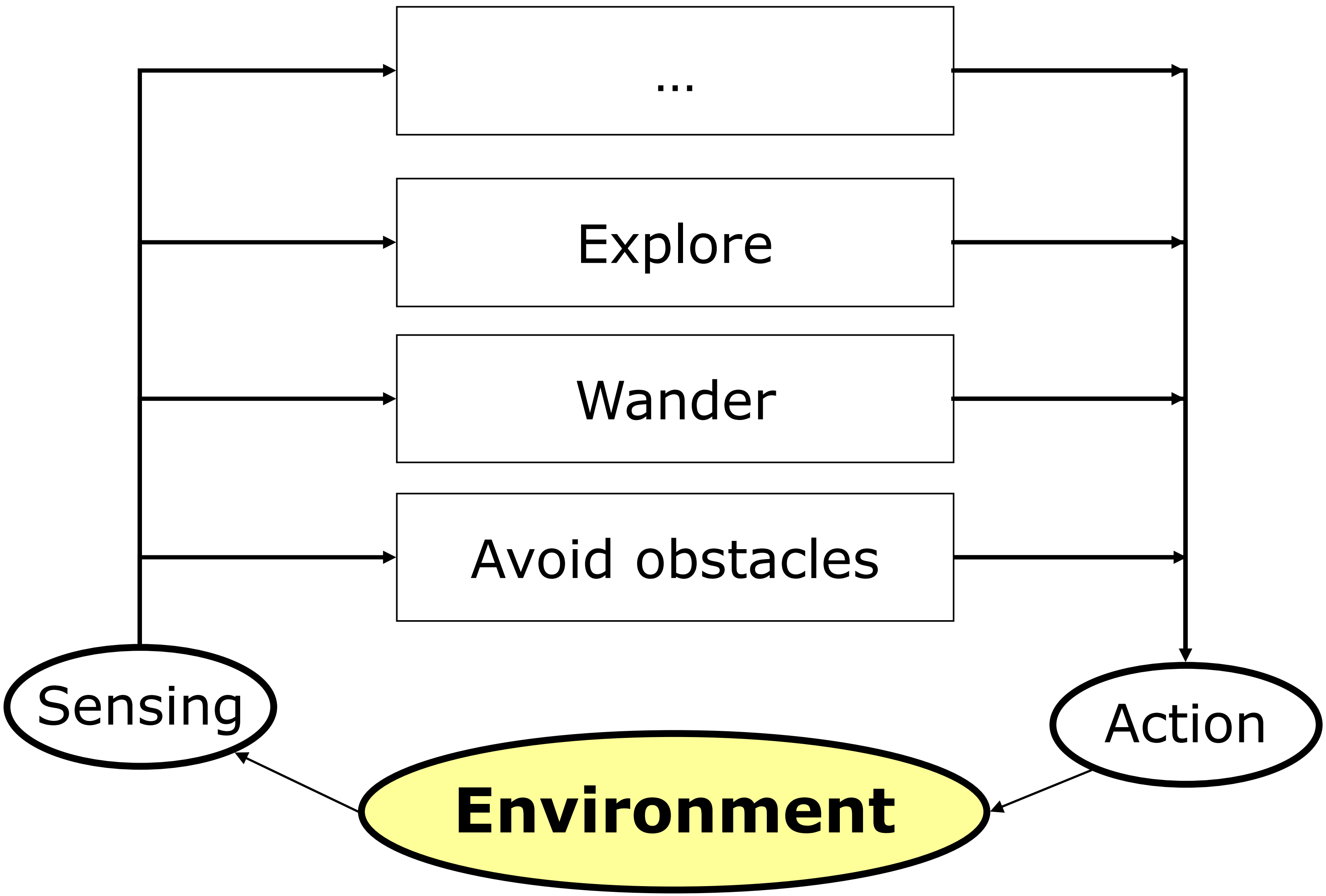


Reactive / Behavior-based Paradigm



- No models: The world is its own, best model
- Easy successes, but also limitations
- Investigate biological systems

Reactive Paradigm as Vertical Decomposition



Characteristics of Reactive Paradigm

- Situated agent, robot is integral part of the world.
- No memory, controlled by what is happening in the world.
- Tight coupling between perception and action via behaviors.
- Only local, behavior-specific sensing is permitted (ego-centric representation).

Behaviors

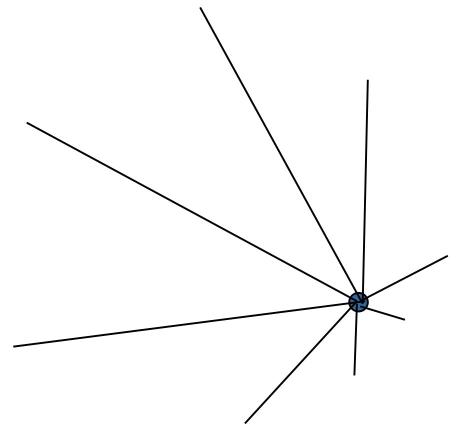
- … are a direct mapping of sensory inputs to a pattern of motor actions that are then used to achieve a task.
- … serve as the basic building block for robotics actions, and the overall behavior of the robot is emergent.
- … support good software design principles due to modularity.

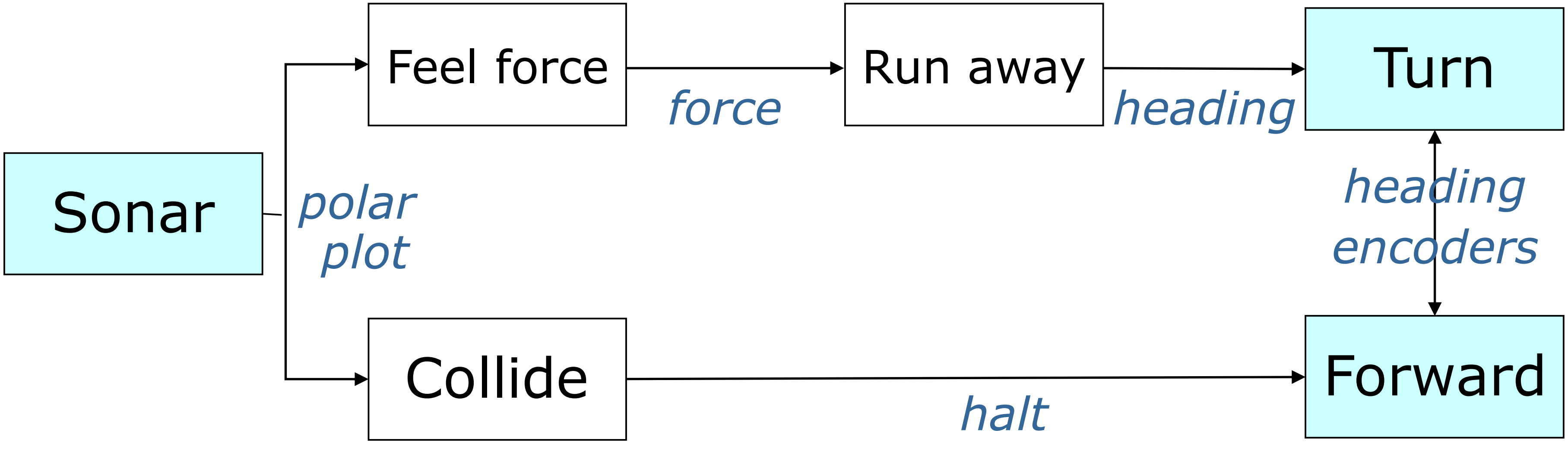
Subsumption Architecture

- Introduced by Rodney Brooks ’86.
- Behaviors are networks of sensing and acting modules (augmented finite state machines AFSM).
- Modules are grouped into layers of competence.
- Layers can subsume lower layers.
- No internal state!

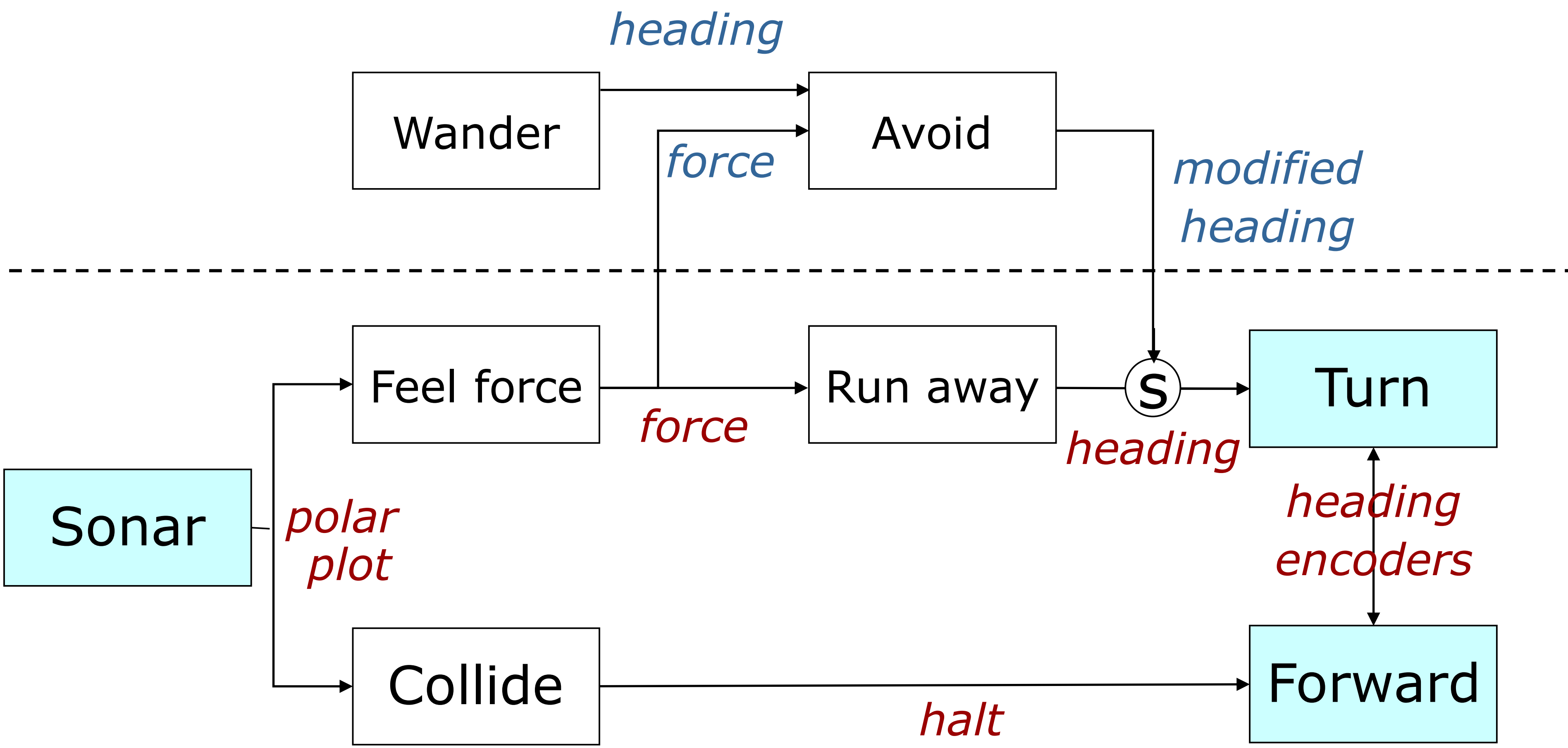
Level 0: Avoid

Polar plot of sonars

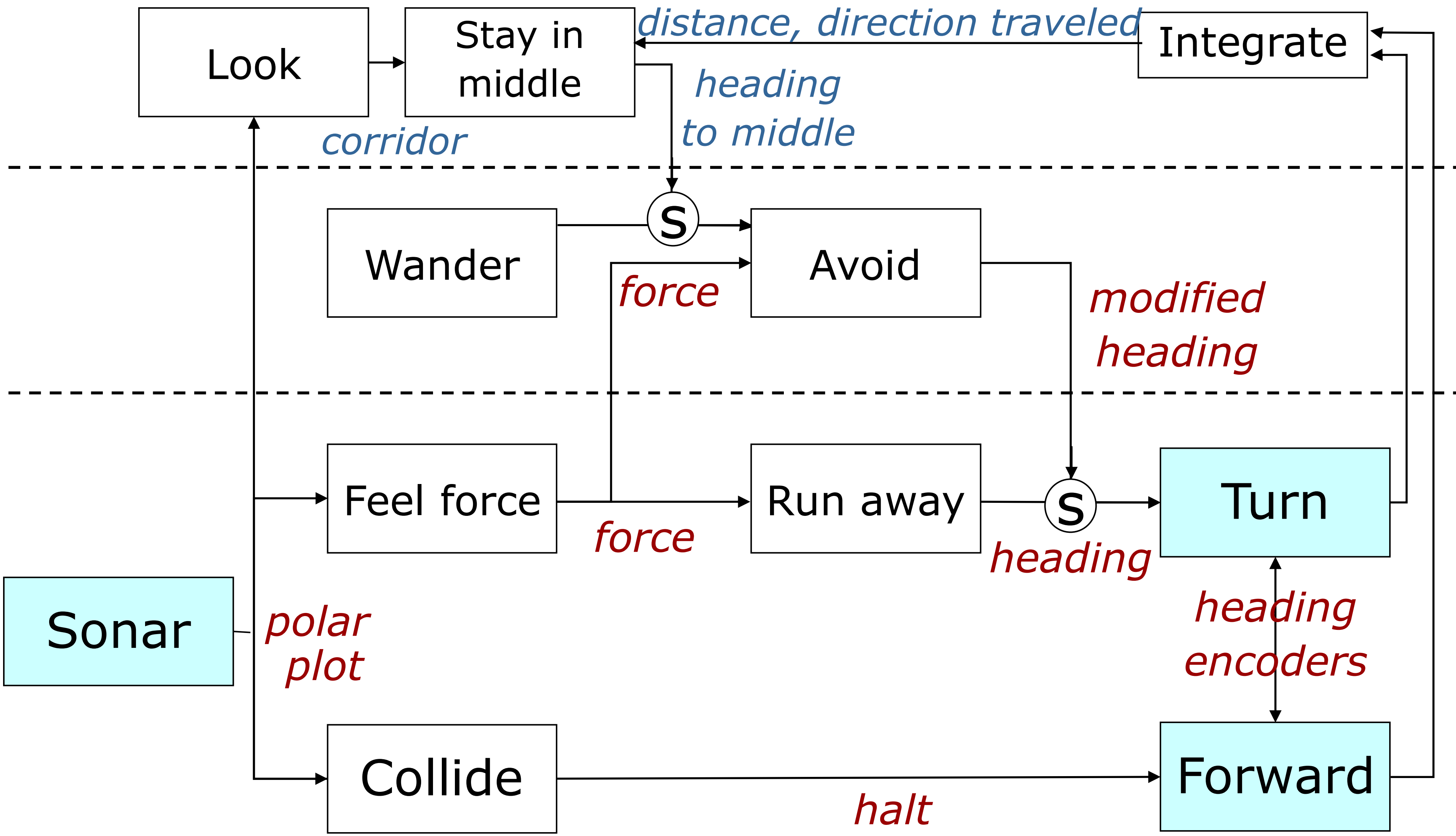




Level 1: Wander



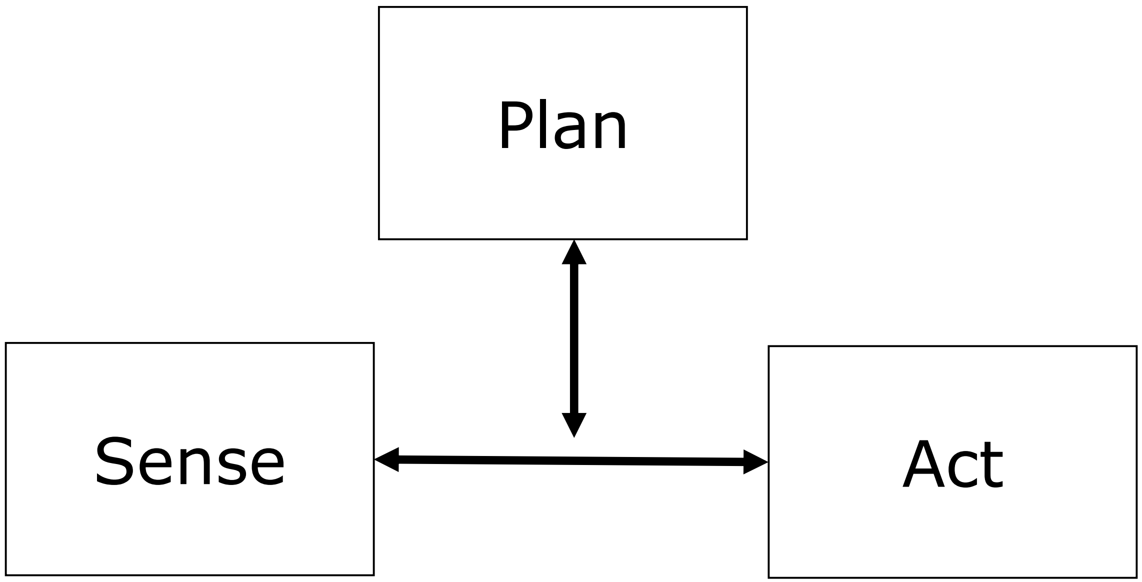
Level 2: Follow Corridor



Reactive Paradigm

- Representations?
- Good software engineering principles?
- Easy to program?
- Robustness?
- Scalability?

Hybrid Deliberative/reactive Paradigm



- Combines advantages of previous paradigms
  - World model used for planning
  - Closed loop, reactive control

Introduction to Mobile Robotics

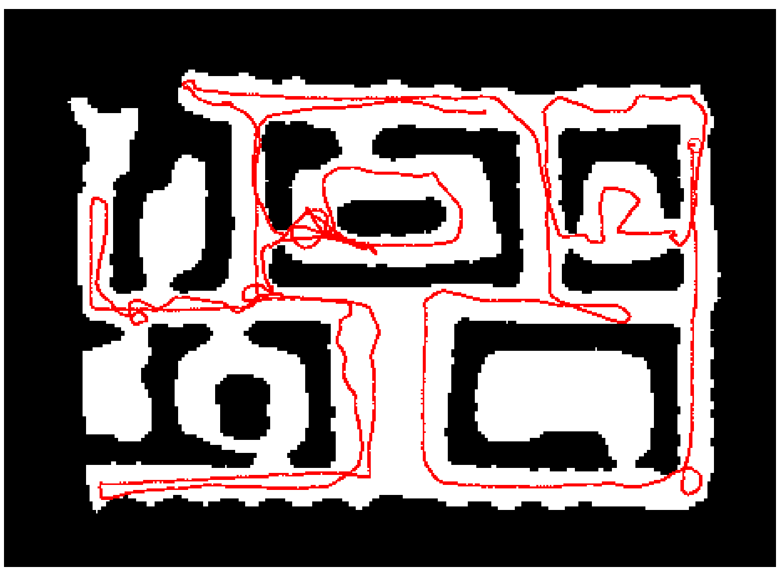
Probabilistic Motion Models

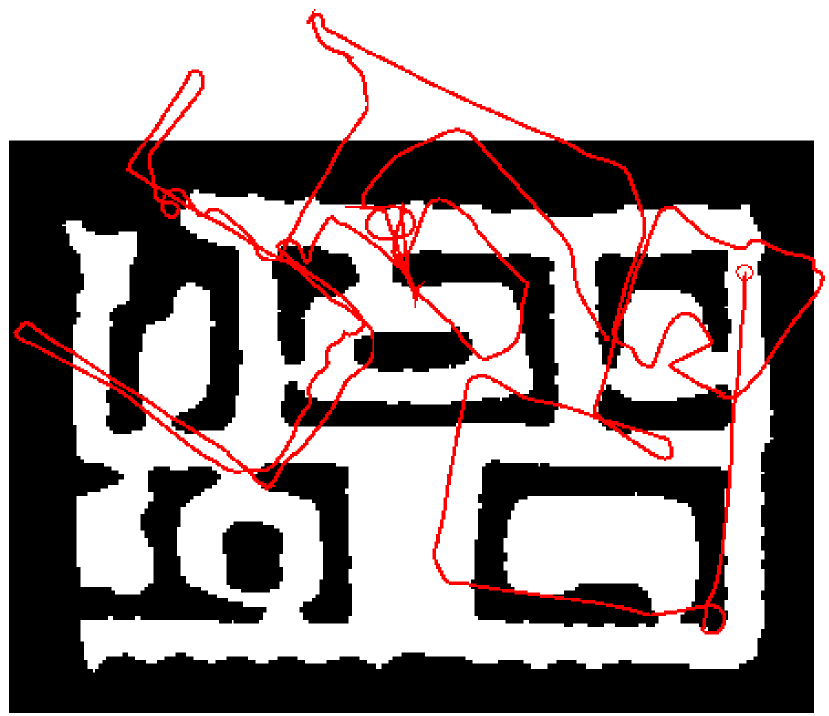
Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

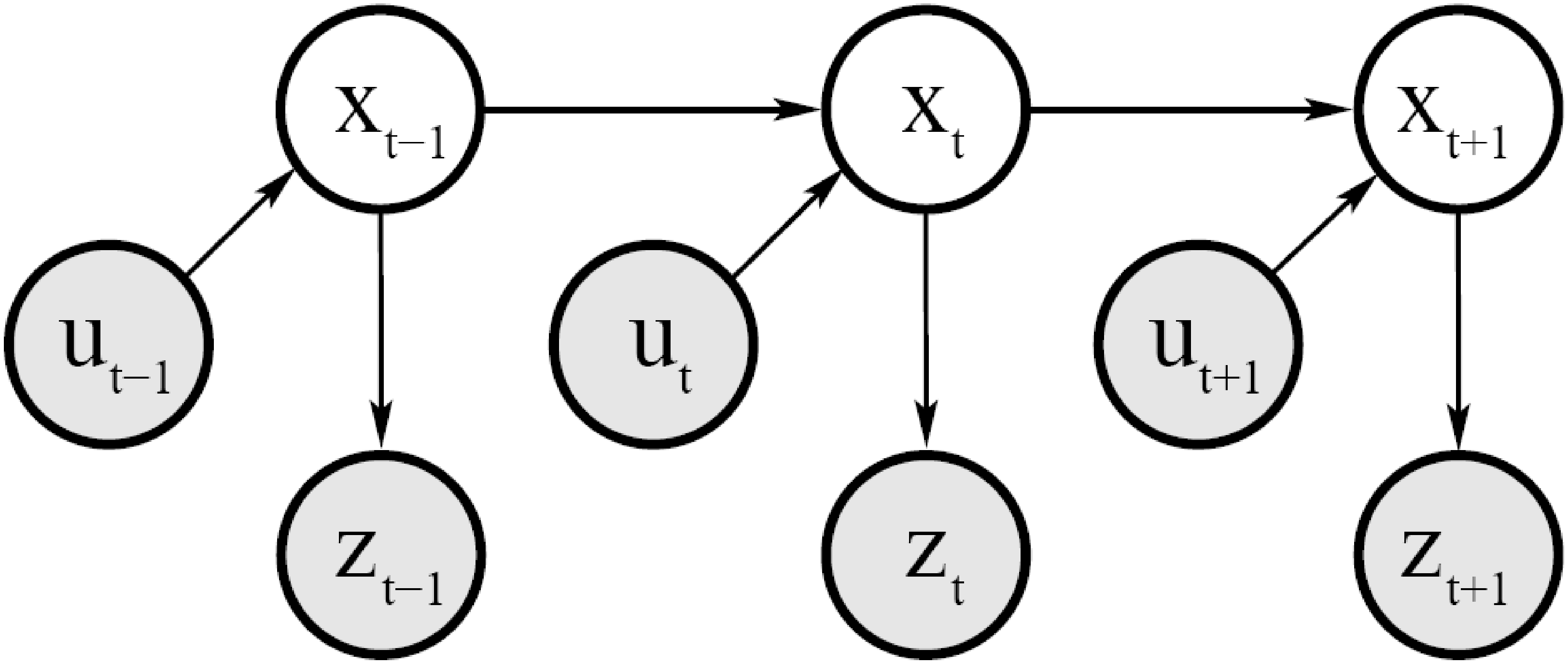
Robot Motion

- Robot motion is inherently uncertain.
- How can we model this uncertainty?





Dynamic Bayesian Network for Controls, States, and Sensations

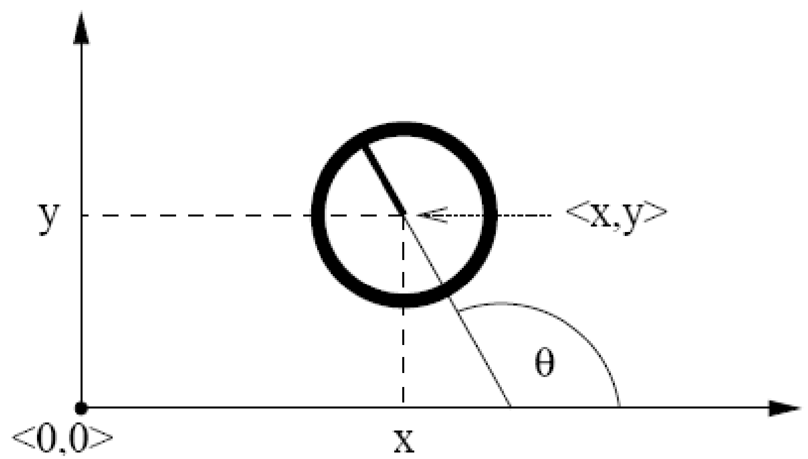


Probabilistic Motion Models

- To implement the Bayes Filter, we need the transition model $p(x \mid x', u)$.
- The term $p(x \mid x', u)$ specifies a posterior probability, that action u carries the robot from x' to x.
- In this section we will specify, how $p(x \mid x', u)$ can be modeled based on the motion equations.

Coordinate Systems

- In general the configuration of a robot can be described by six parameters.
- Three-dimensional Cartesian coordinates plus three Euler angles pitch, roll, and tilt.
- Throughout this section, we consider robots operating on a planar surface.
- The state space of such systems is three-dimensional (x,y,θ).

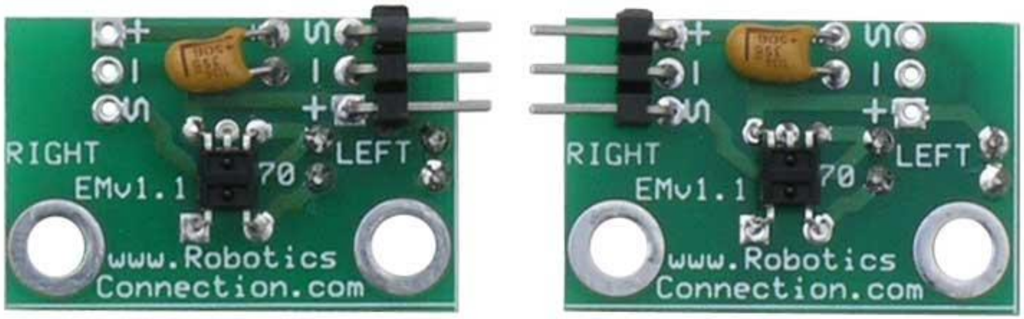


Typical Motion Models

- In practice, one often finds two types of motion models:
  - **Odometry-based**
  - **Velocity-based** (**dead reckoning**)
- Odometry-based models are used when systems are equipped with wheel encoders.
- Velocity-based models have to be applied when no wheel encoders are given.
- They calculate the new pose based on the velocities and the time elapsed.

Example Wheel Encoders

These modules require +5V and GND to power them, and provide a 0 to 5V output. They provide +5V output when they "see" white, and a 0V output when they "see" black.



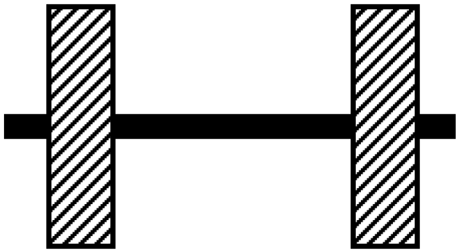
These disks are manufactured out of high quality laminated color plastic to offer a very crisp black to white transition. This enables a wheel encoder sensor to easily see the transitions.

Source: http://www.active-robots.com/

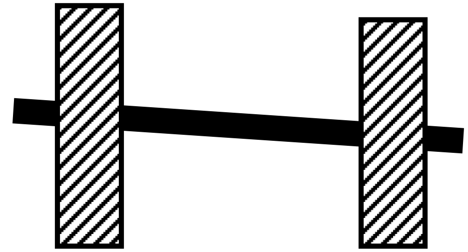
Dead Reckoning

- Derived from “deduced reckoning.”
- Mathematical procedure for determining the present location of a vehicle.
- Achieved by calculating the current pose of the vehicle based on its velocities and the time elapsed.

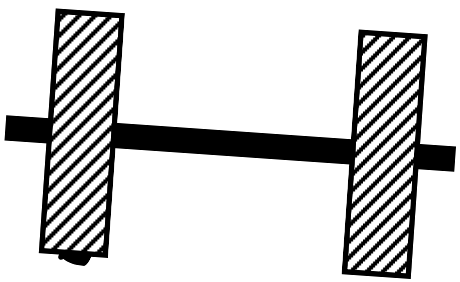
Reasons for Motion Errors



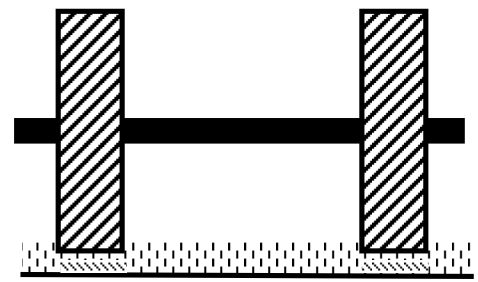
ideal case



different wheel diameters



bump



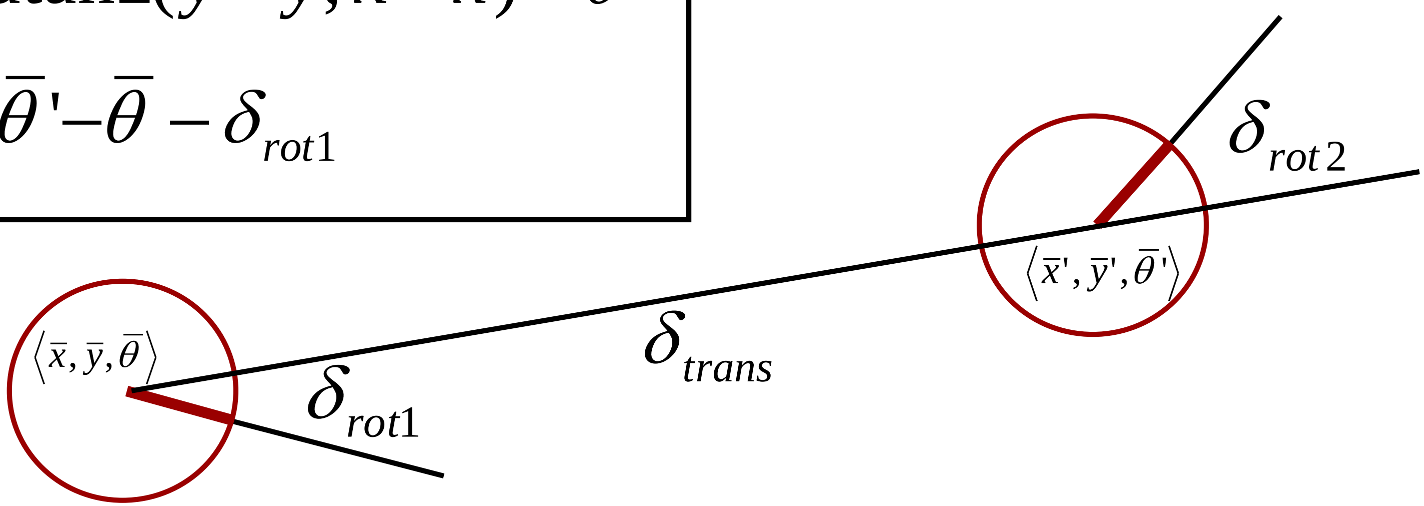
carpet

and many more …

Odometry Model

- Robot moves from $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$ to $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$.
- Odometry information $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$.

$$\boxed{\begin{aligned} \delta_{trans} &= \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \\ \delta_{rot1} &= \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\ \delta_{rot2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot1} \end{aligned}}$$



Noise Model for Odometry

- The measured motion is given by the true motion corrupted with noise.

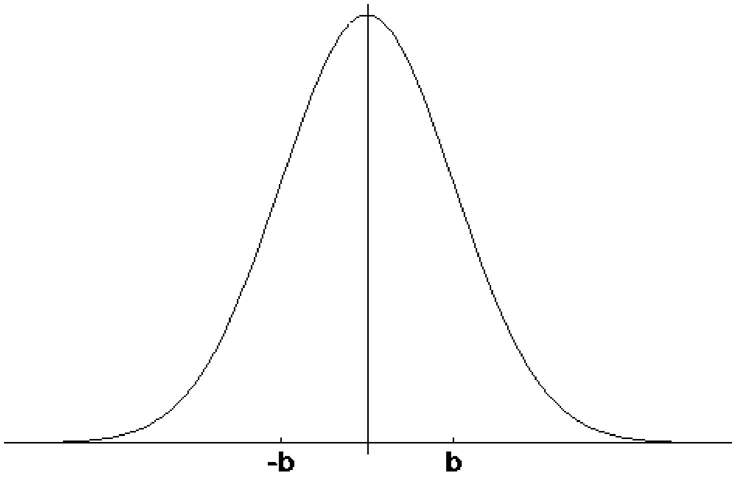
$$\hat{\delta}_{rot1} = \delta_{rot1} + \varepsilon_{\alpha_1 |\delta_{rot1}| + \alpha_2 |\delta_{trans}|}$$

$$\hat{\delta}_{trans} = \delta_{trans} + \varepsilon_{\alpha_3 |\delta_{trans}| + \alpha_4 |\delta_{rot1} + \delta_{rot2}|}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} + \varepsilon_{\alpha_1 |\delta_{rot2}| + \alpha_2 |\delta_{trans}|}$$

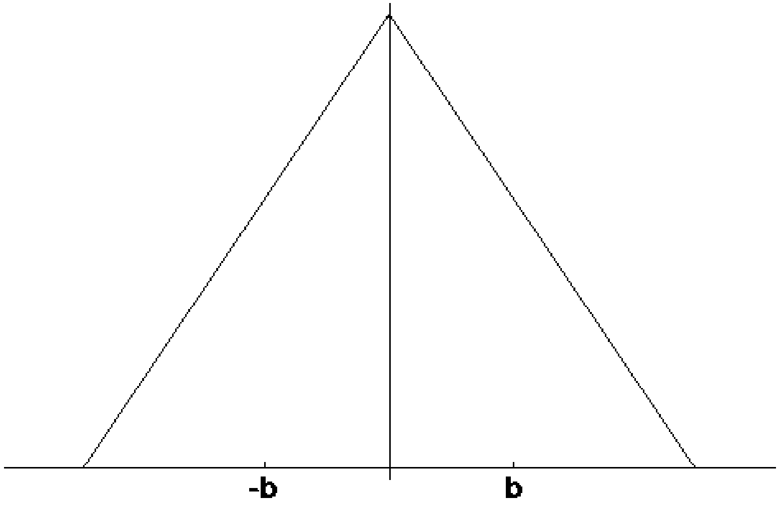
Typical Distributions for Probabilistic Motion Models

Normal distribution



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

Triangular distribution

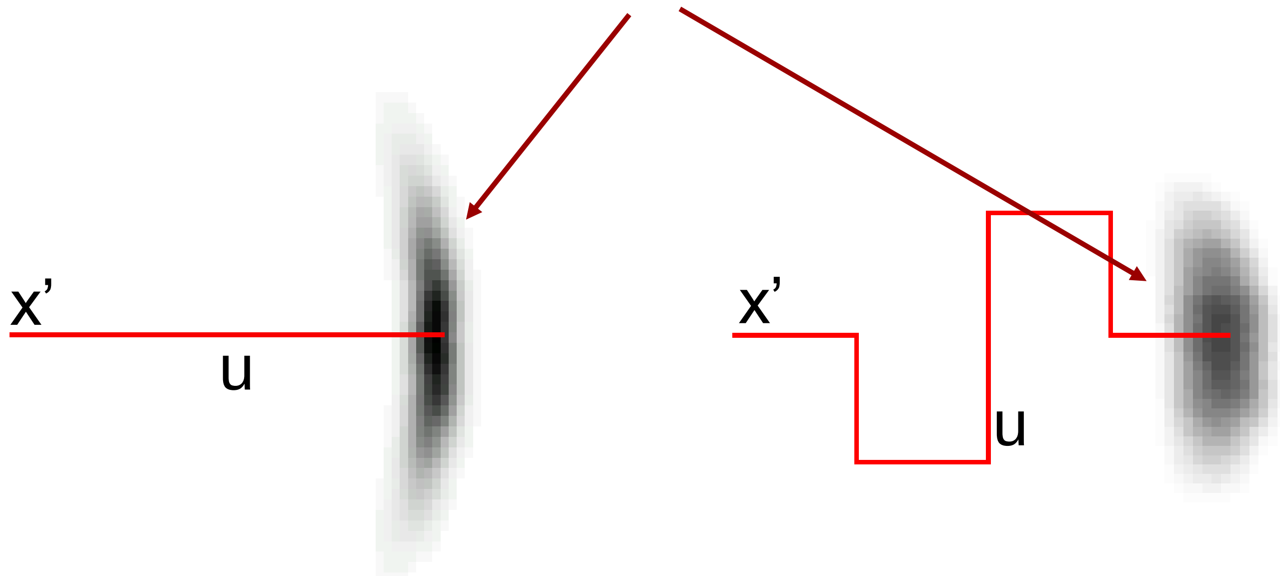


$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0 \text{ if } |\mathrm{x}| > \sqrt{6\sigma^2} \\ \dfrac{\sqrt{6\sigma^2} - |x|}{6\sigma^2} \end{cases}$$

Application

- Repeated application of the sensor model for short movements.
- Typical banana-shaped distributions obtained for 2d-projection of 3d posterior.

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



Sample Odometry Motion Model

1. Algorithm **sample_motion_model**(u, x):

   $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \theta \rangle$

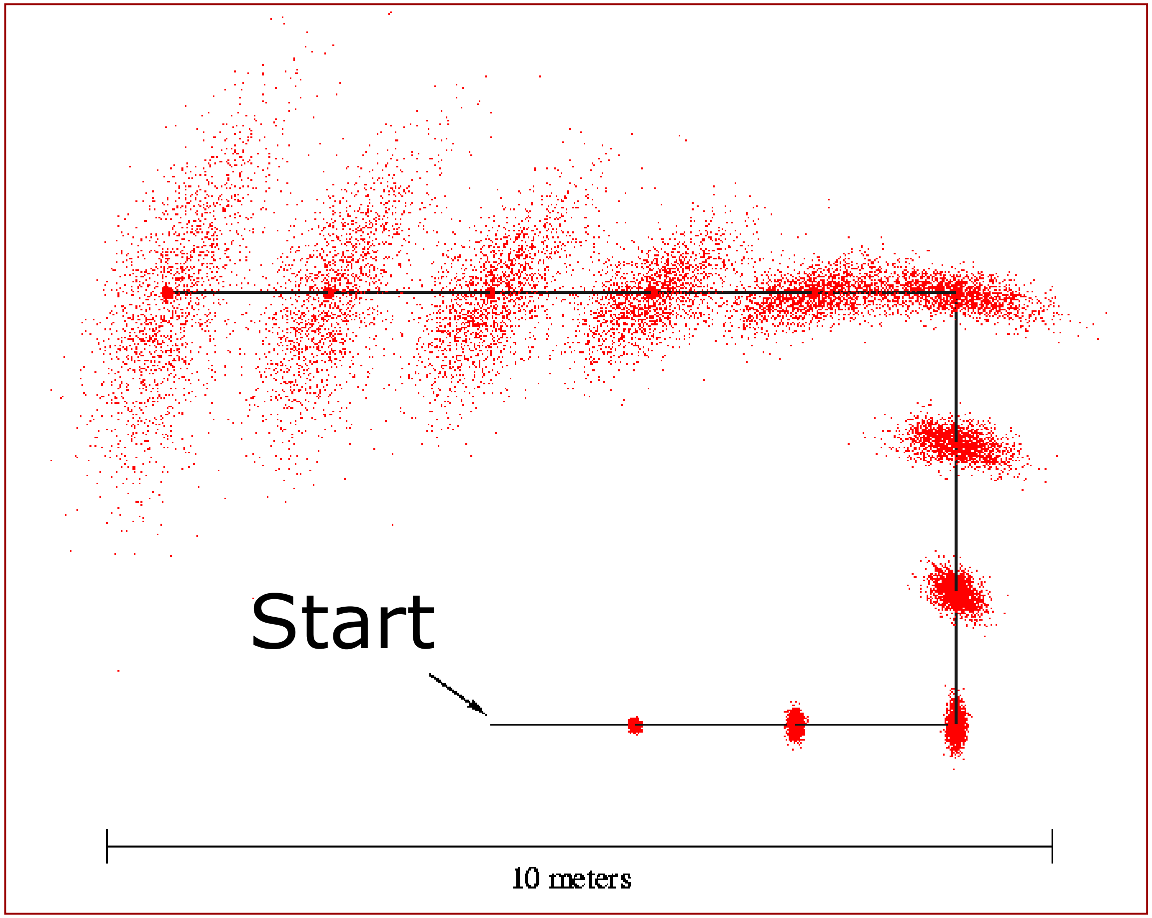
1. $\hat{\delta}_{rot1} = \delta_{rot1} + \text{sample}(\alpha_1 | \delta_{rot1} | + \alpha_2 \ \delta_{trans})$
2. $\hat{\delta}_{trans} = \delta_{trans} + \text{sample}(\alpha_3 \ \delta_{trans} + \alpha_4 \ (| \delta_{rot1} | + | \delta_{rot2} |))$
3. $\hat{\delta}_{rot2} = \delta_{rot2} + \text{sample}(\alpha_1 | \delta_{rot2} | + \alpha_2 \ \delta_{trans})$

4. $x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
5. $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6. $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$

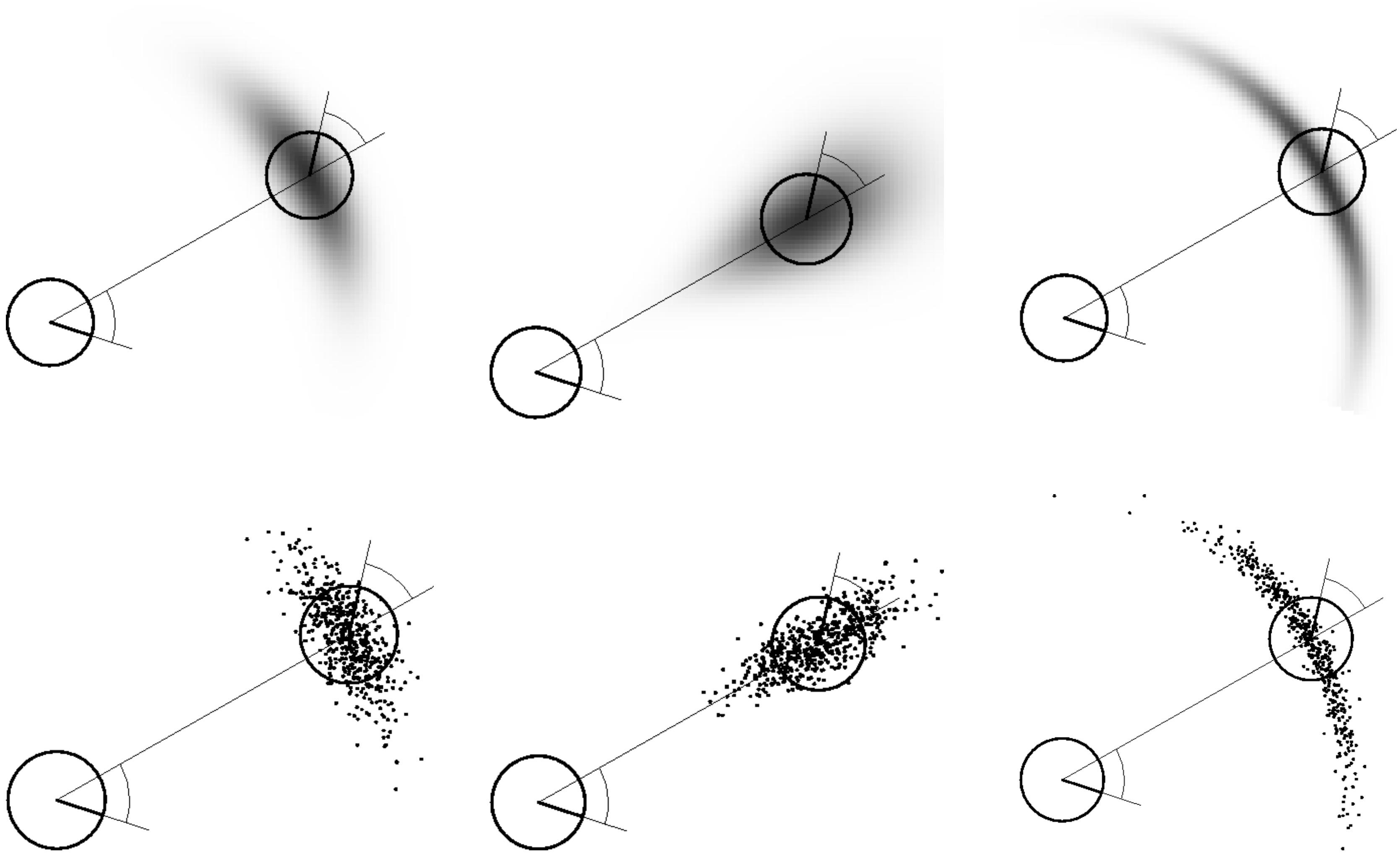
7. Return $\langle x', y', \theta' \rangle$

sample_normal_distribution

Sampling from Our Motion Model



Examples (Odometry-Based)



Introduction to Mobile Robotics

Probabilistic Sensor Models

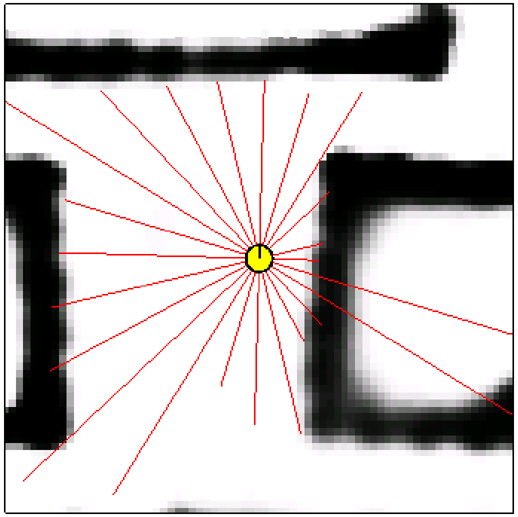
Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Giorgio Grisetti, Kai Arras

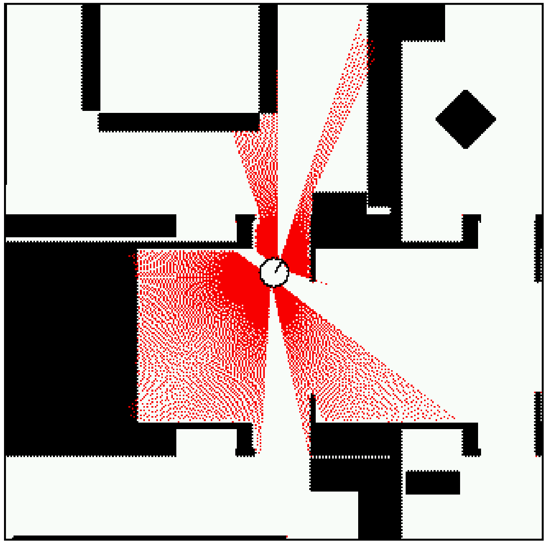
UNI FREIBURG

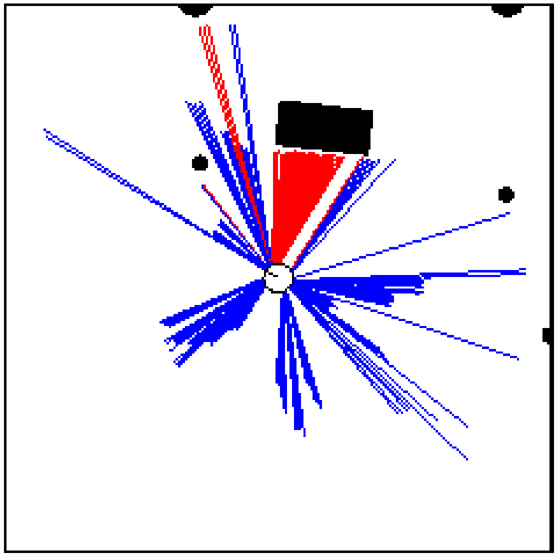
Sensors for Mobile Robots

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

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

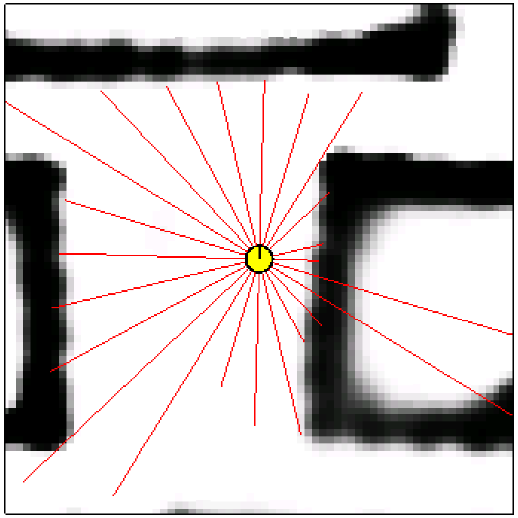
- Scan *z* consists of *K* measurements.

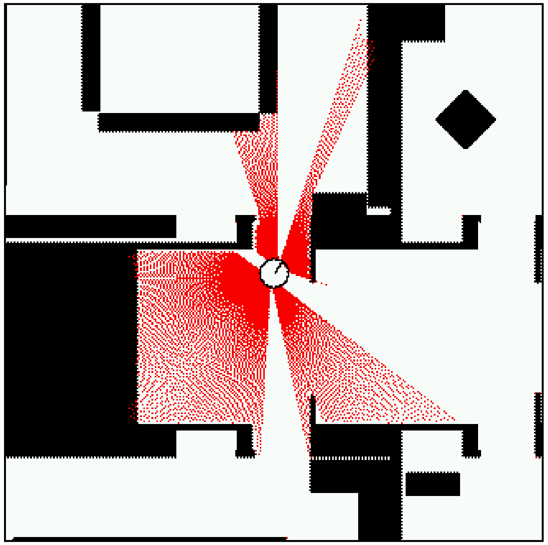
$$z = \{z_1, z_2, ..., z_K\}$$

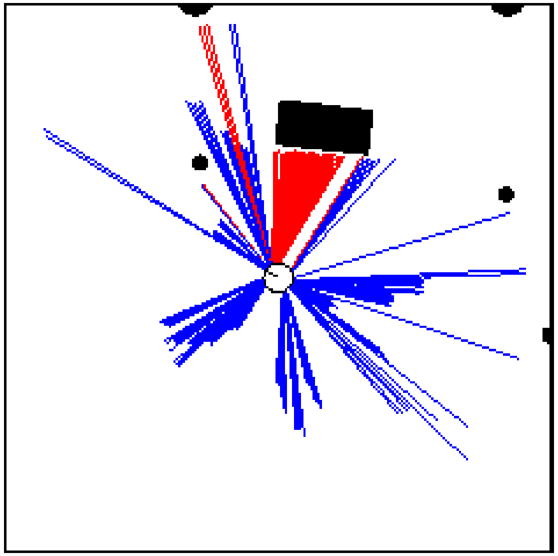
- Individual measurements are independent given the robot position.

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

Beam-based Sensor Model



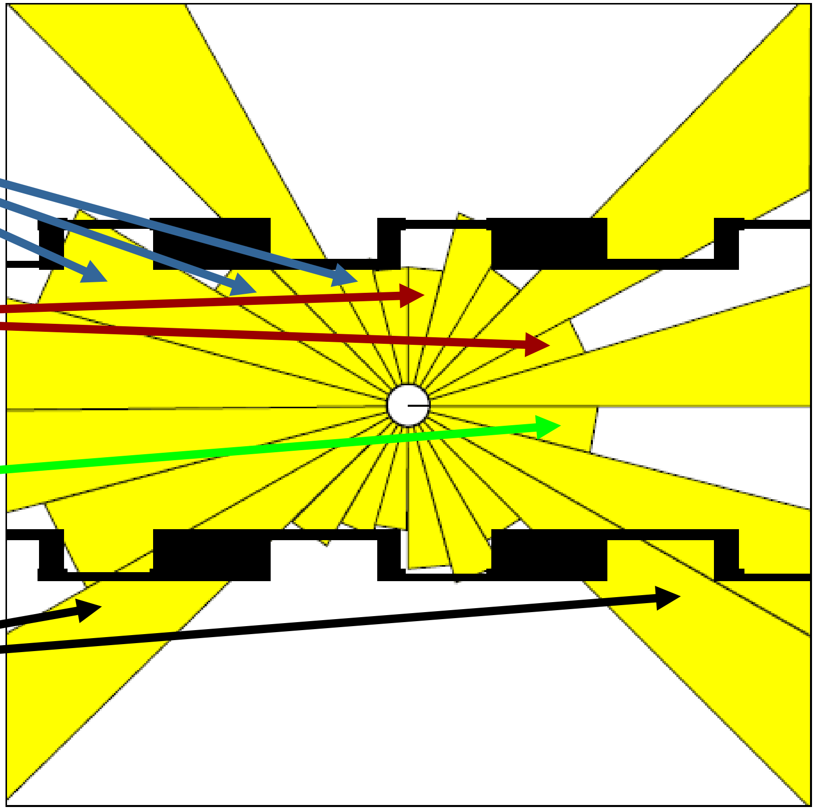




$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$

Typical Measurement Errors of an Range Measurements

1. Beams reflected by obstacles
2. Beams reflected by persons / caused by crosstalk
3. Random measurements
4. Maximum range measurements

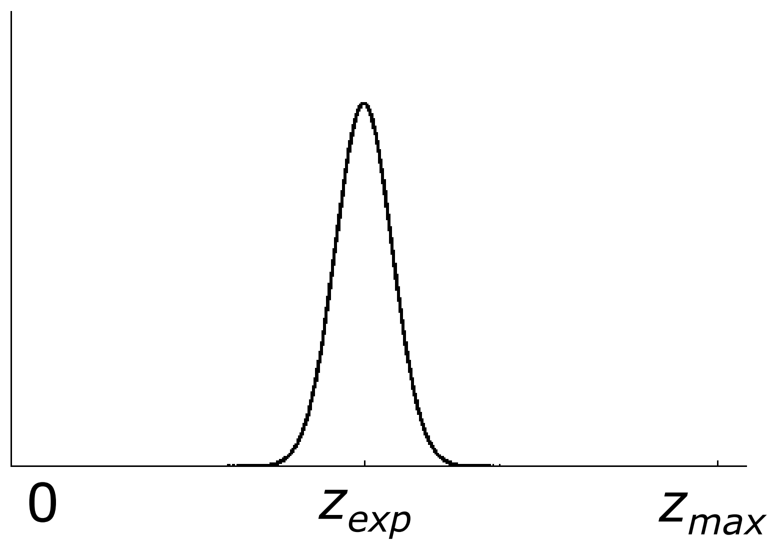


Proximity Measurement

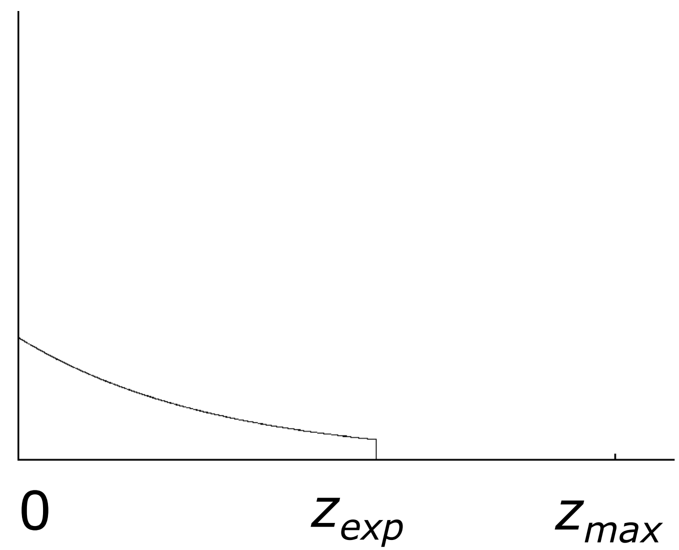
- Measurement can be caused by …
  - a known obstacle.
  - cross-talk.
  - an unexpected obstacle (people, furniture, …).
  - missing all obstacles (total reflection, glass, …).
- Noise is due to uncertainty …
  - in measuring distance to known obstacle.
  - in position of known obstacles.
  - in position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

Beam-based Proximity Model

Measurement noise



Unexpected obstacles

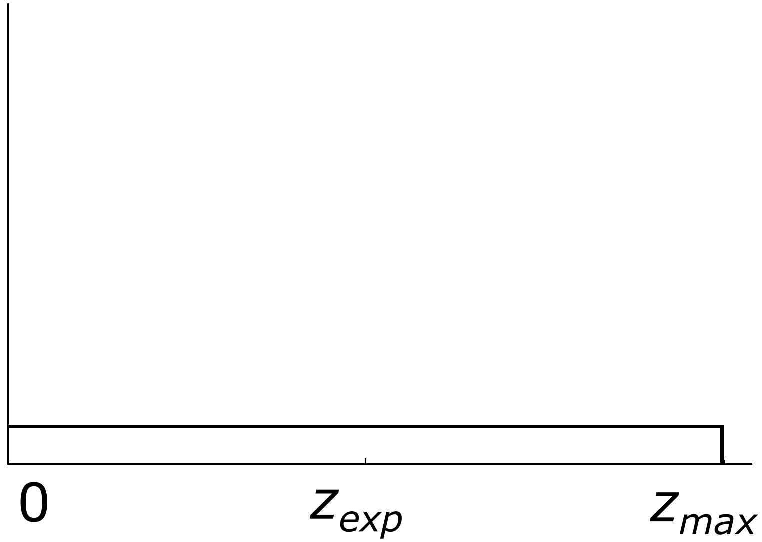


$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z-z_{\exp})^2}{b}}$$

$$P_{\text{unexp}}(z \mid x, m) = \begin{Bmatrix} \eta\ \lambda\ \mathrm{e}^{-\lambda z} & z < z_{\exp} \\ 0 & otherwise \end{Bmatrix}$$

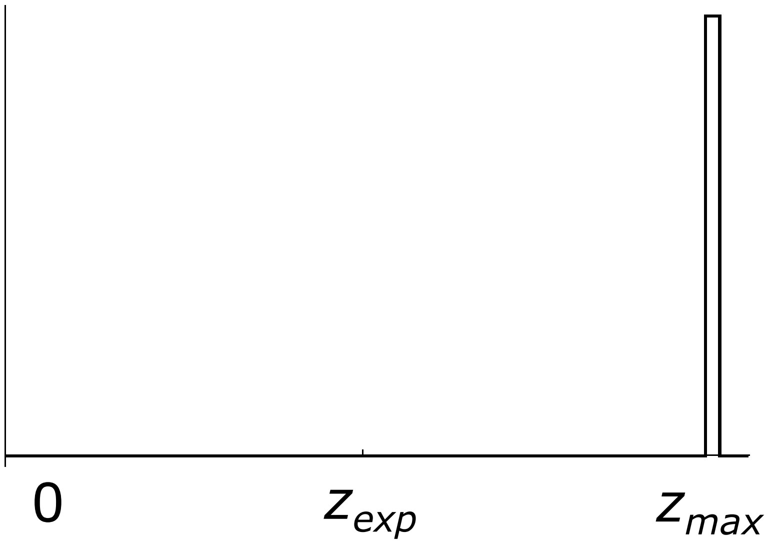
Beam-based Proximity Model

Random measurement



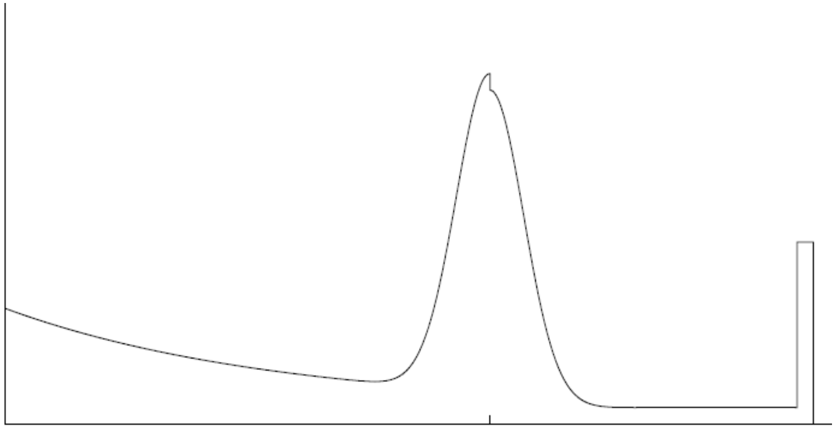
$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

Max range



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

Resulting Mixture Density

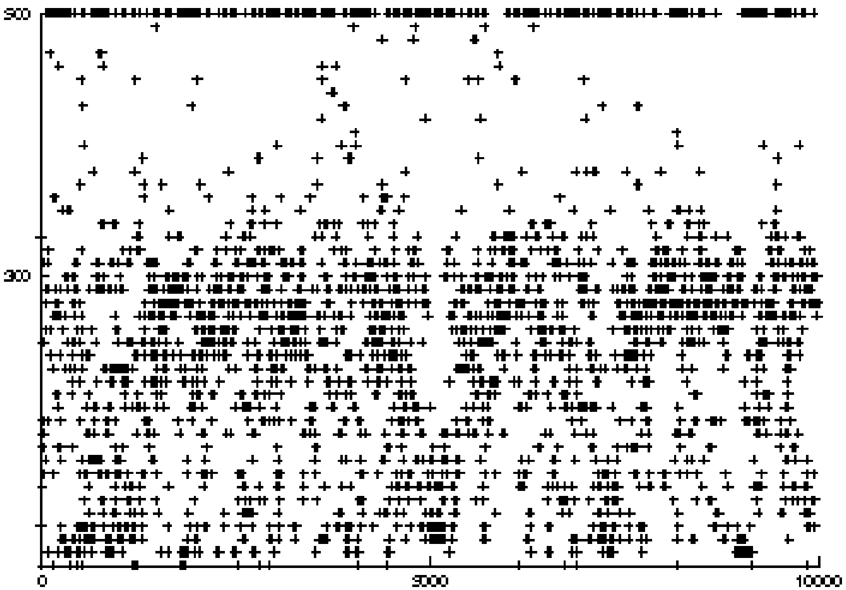


$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\text{max}}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

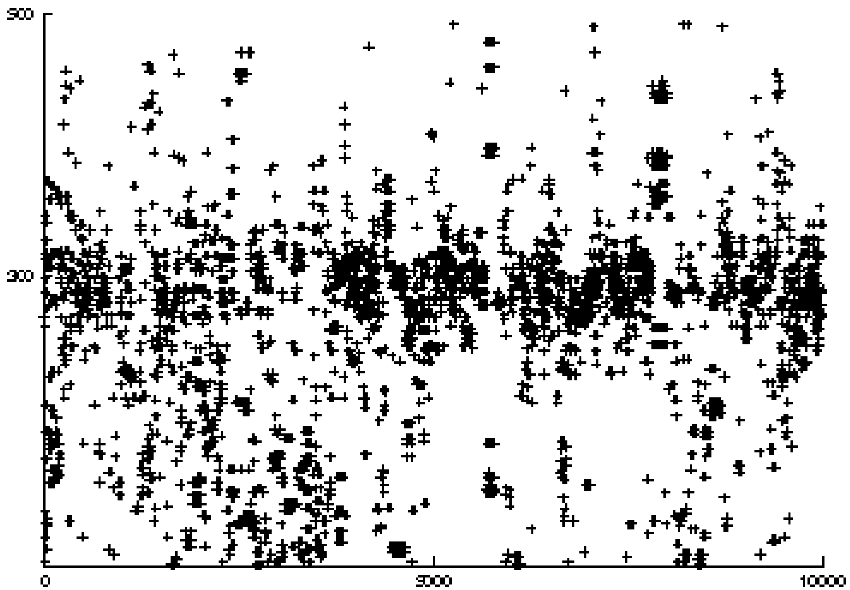
How can we determine the model parameters?

Raw Sensor Data

Measured distances for expected distance of 300 cm.



Sonar



Laser

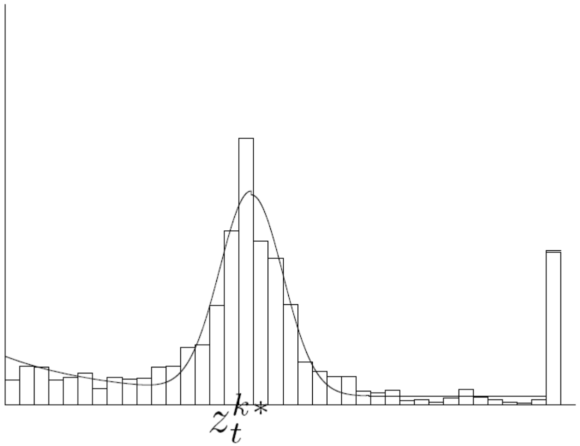
Approximation

- Maximize log likelihood of the data

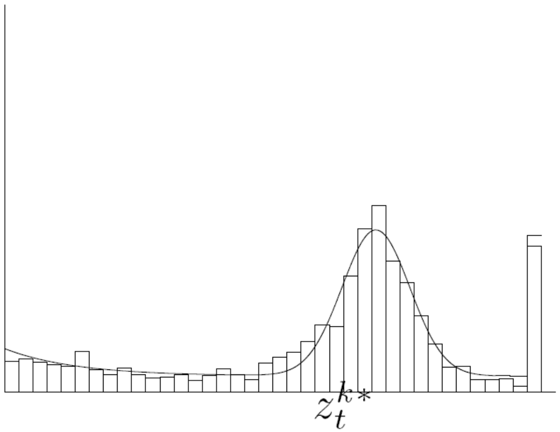
$$P(z \mid z_{\exp})$$

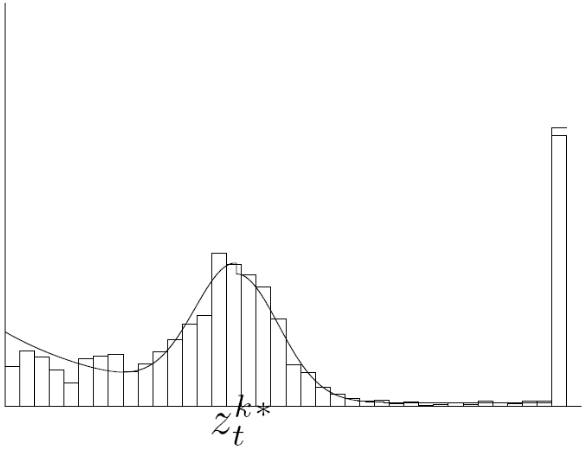
- Search space of n-1 parameters.
  - Hill climbing
  - Gradient descent
  - Genetic algorithms
  - …
- Deterministically compute the n-th parameter to satisfy normalization constraint.

Approximation Results

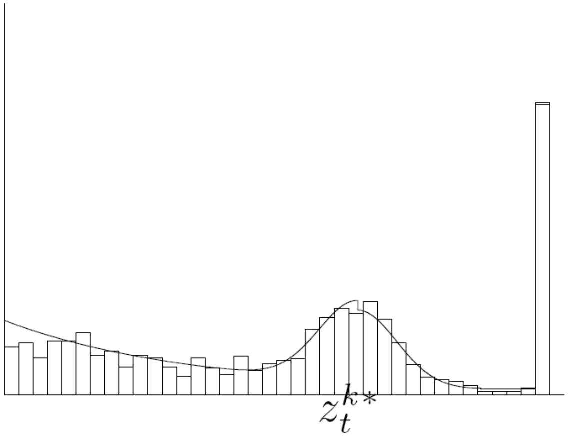


Laser





Sonar

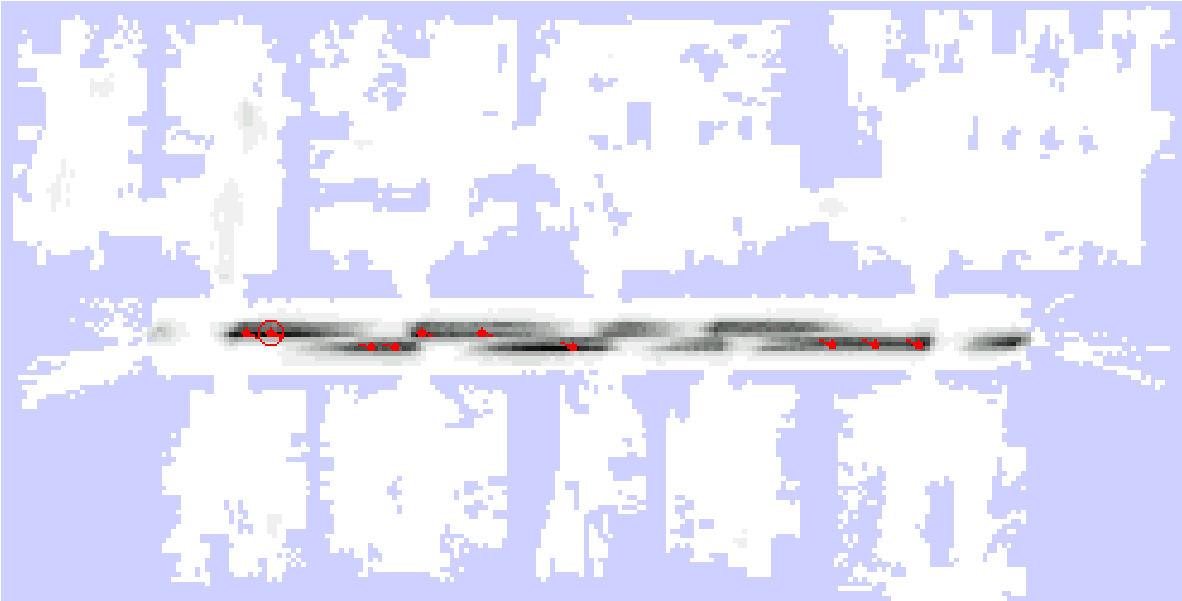


300cm

400cm

Example

z

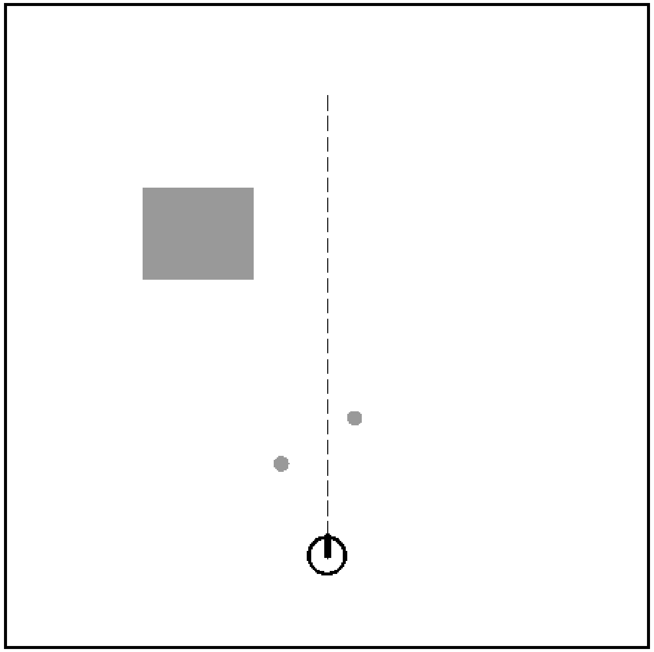


$P(z|x,m)$

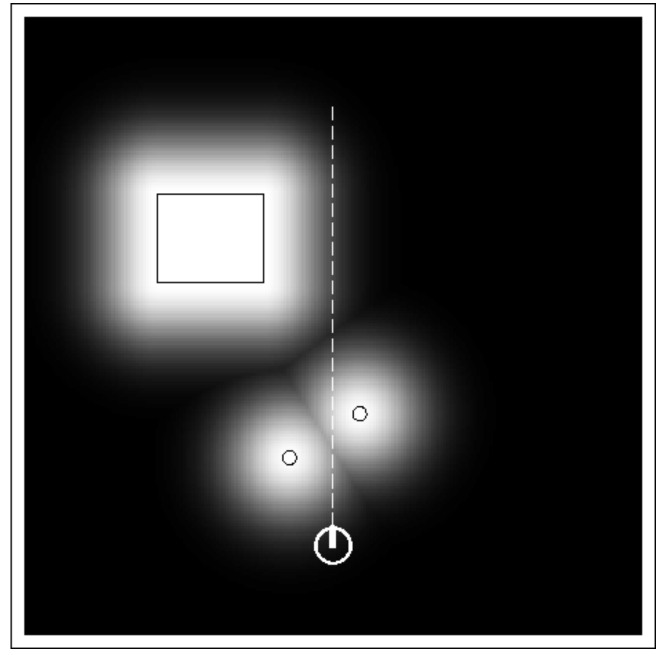
Scan-based Model

- Probability is a mixture of …
  - a Gaussian distribution with mean at distance to closest obstacle,
  - a uniform distribution for random measurements, and
  - a small uniform distribution for max range measurements.
- Again, independence between different components is assumed.

Example

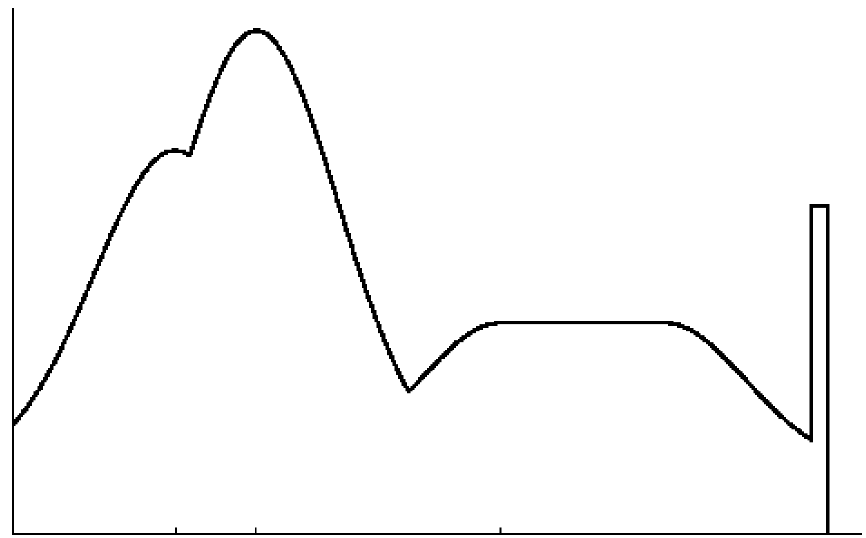


Map m

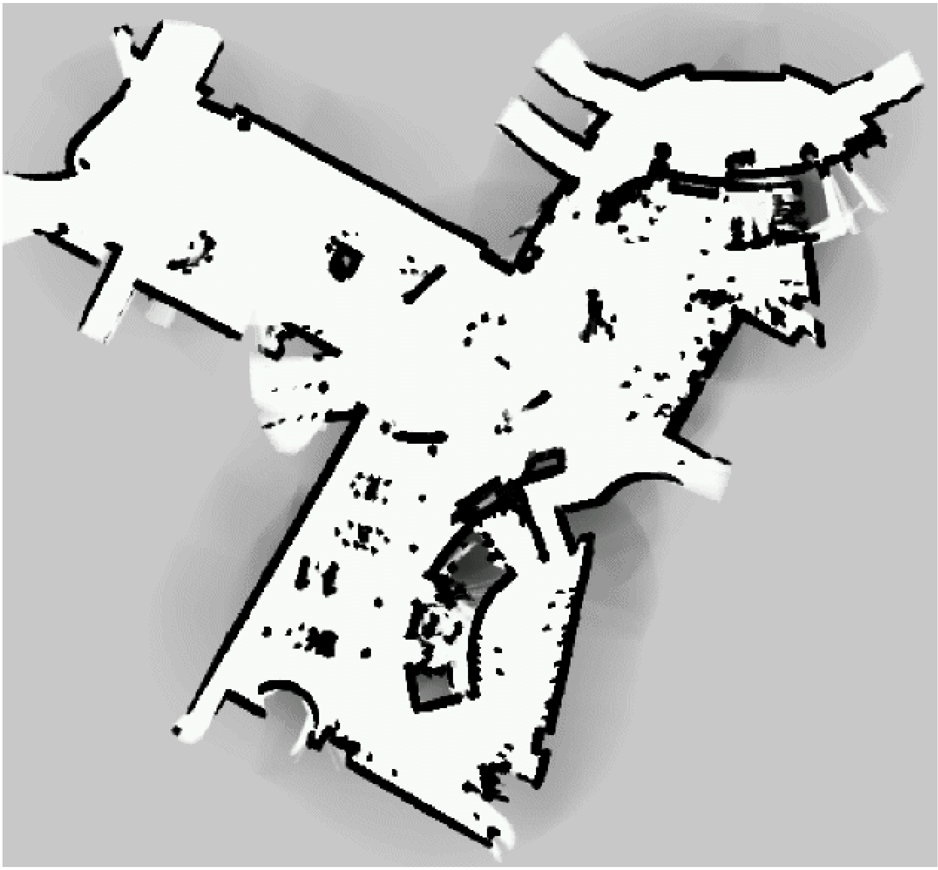


Likelihood field

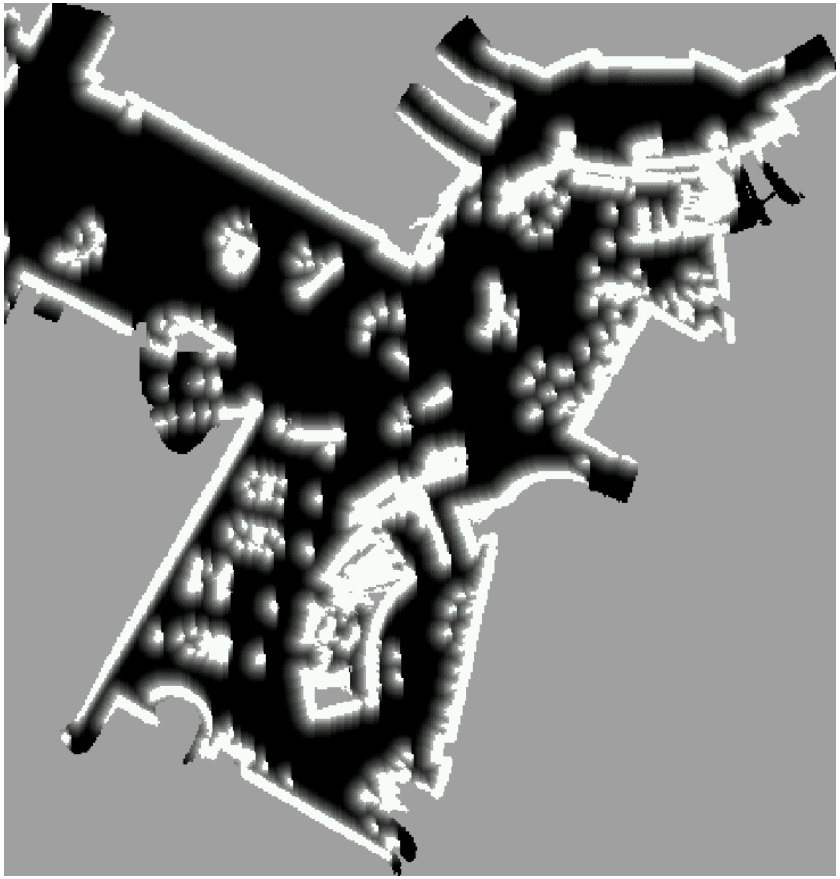
$P(z|x,m)$



San Jose Tech Museum



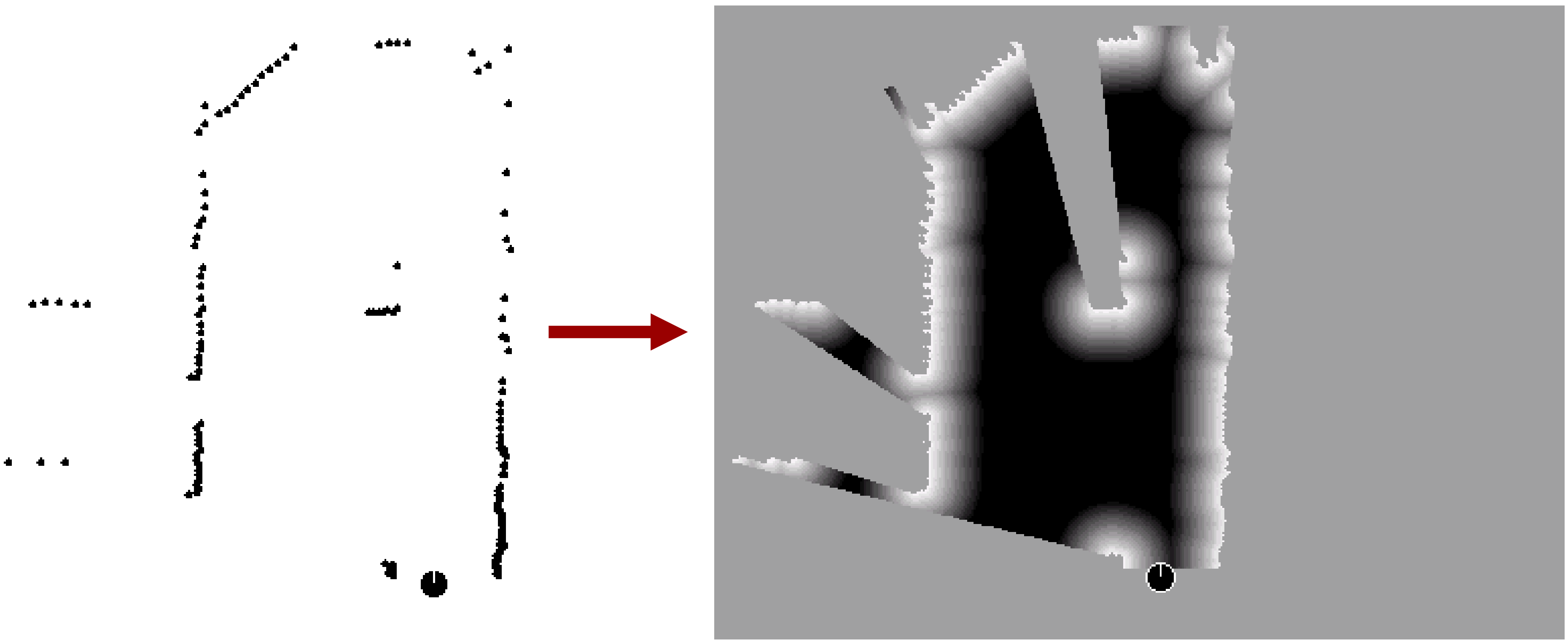
Occupancy grid map



Likelihood field

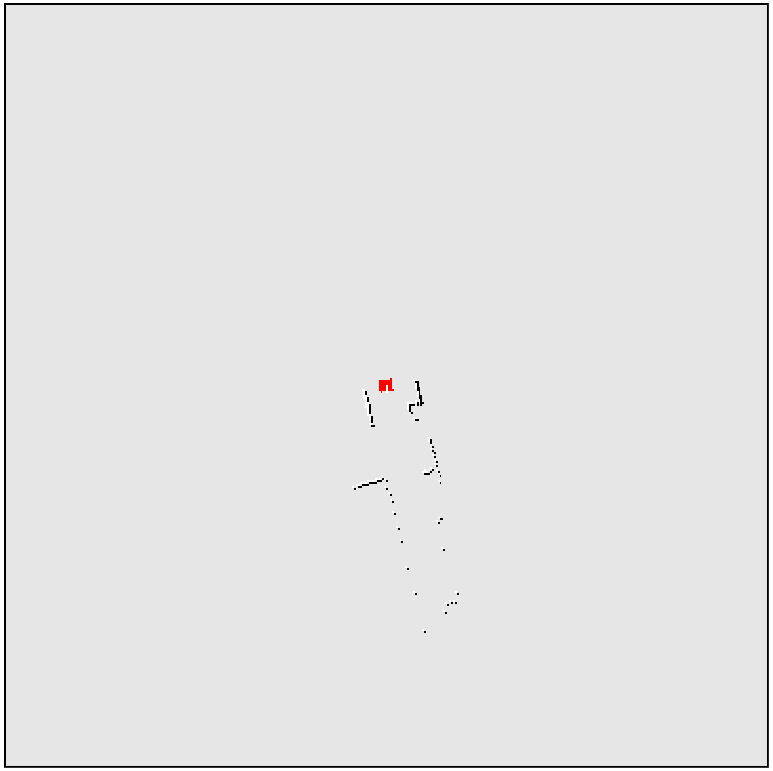
Scan Matching

- Extract likelihood field from scan and use it to match different scan.



Scan Matching

- Extract likelihood field from first scan and use it to match second scan.



~0.01 sec

Properties of Scan-based Model

- Highly efficient, uses 2D tables only.
- Smooth w.r.t. to small changes in robot position.
- Allows gradient descent, scan matching.
- Ignores physical properties of beams.
- Will it work for ultrasound sensors?

Additional Models of Proximity Sensors

- Map matching (sonar, laser): generate small, local maps from sensor data and match local maps against global model.
- Scan matching (laser): map is represented by scan endpoints, match scan into this map.
- Features (sonar, laser, vision): Extract features such as doors, hallways from sensor data.

Landmarks

- Active beacons (*e.g.*, radio, GPS)
- Passive (*e.g.*, visual, retro-reflective)
- Standard approach is triangulation

- Sensor provides
  - distance, or
  - bearing, or
  - distance and bearing.

Distance and Bearing

Summary of Sensor Models

- Explicitly modeling uncertainty in sensing is key to robustness.
- In many cases, good models can be found by the following approach:
  1. Determine parametric model of noise free 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.
- This holds for motion models as well.
- It is extremely important to be aware of the underlying assumptions!

Introduction to Mobile Robotics

Mapping with Known Poses

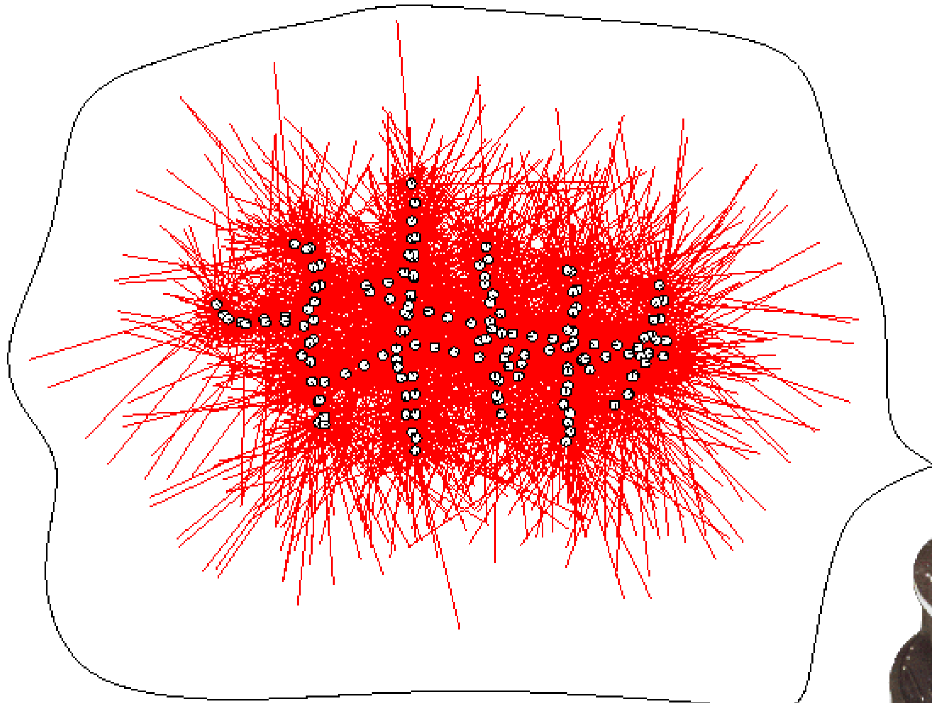
Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

Why Mapping?

- Learning maps is one of the fundamental problems in mobile robotics
- Maps allow robots to efficiently carry out their tasks, allow localization …
- Successful robot systems rely on maps for localization, path planning, activity planning etc.

The General Problem of Mapping



What does the environment look like?

The General Problem of Mapping

- Formally, mapping involves, given the sensor data,

$$d = \{u_1, z_1, u_2, z_2, \ldots, u_n, z_n\}$$

to calculate the most likely map

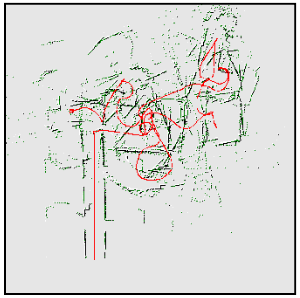
$$m^* = \arg\max_m P(m \mid d)$$

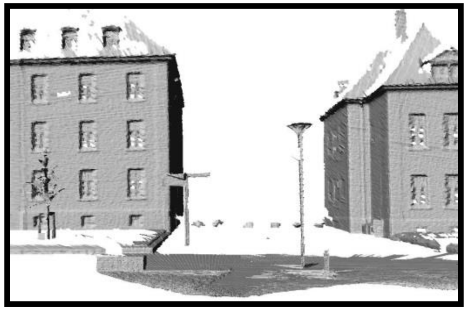
Mapping as a Chicken and Egg Problem

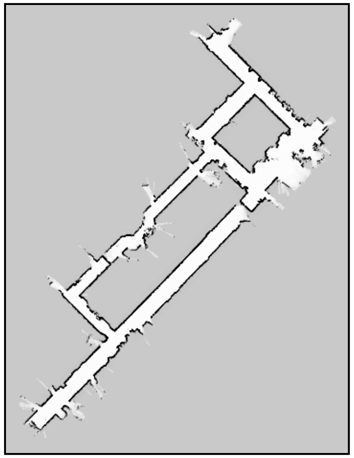
- So far we learned how to estimate the pose of the vehicle given the data and the map.
- Mapping, however, involves to simultaneously estimate the pose of the vehicle and the map.
- The general problem is therefore denoted as the simultaneous localization and mapping problem (SLAM).
- Throughout this section we will describe how to calculate a map given we know the pose of the vehicle.

Types of SLAM-Problems

- Grid maps or scans

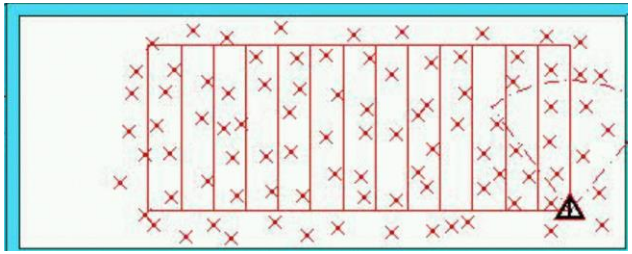


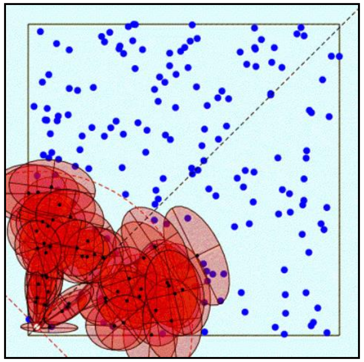


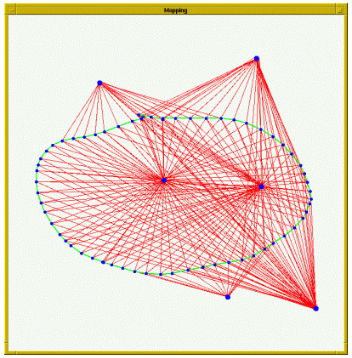


[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

- Landmark-based







[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

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 estimated
  - How can we identify that we are at a previously visited place?
  - This problem is the so-called data association problem.

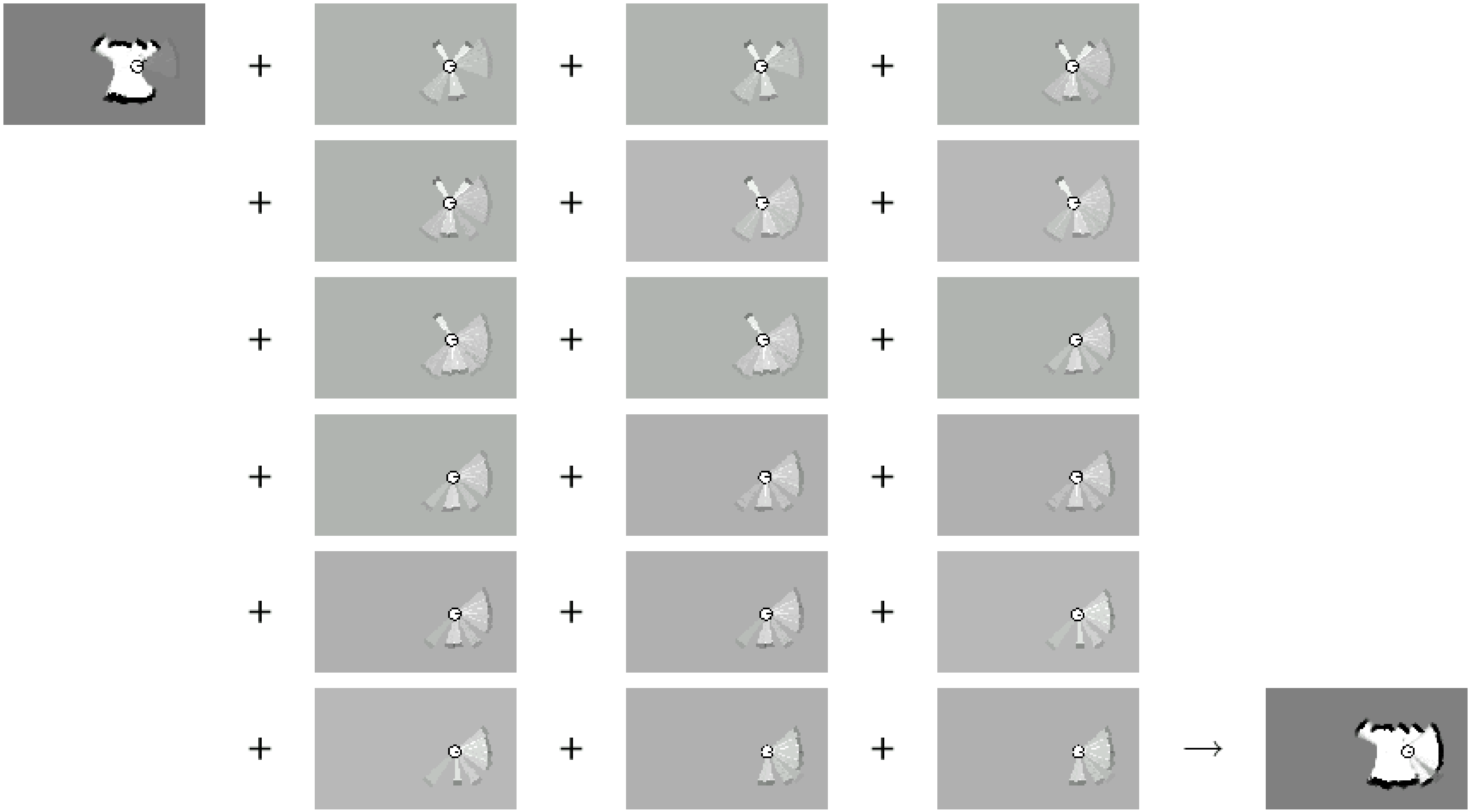
Occupancy Grid Maps

- 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 (m[xy]) is independent

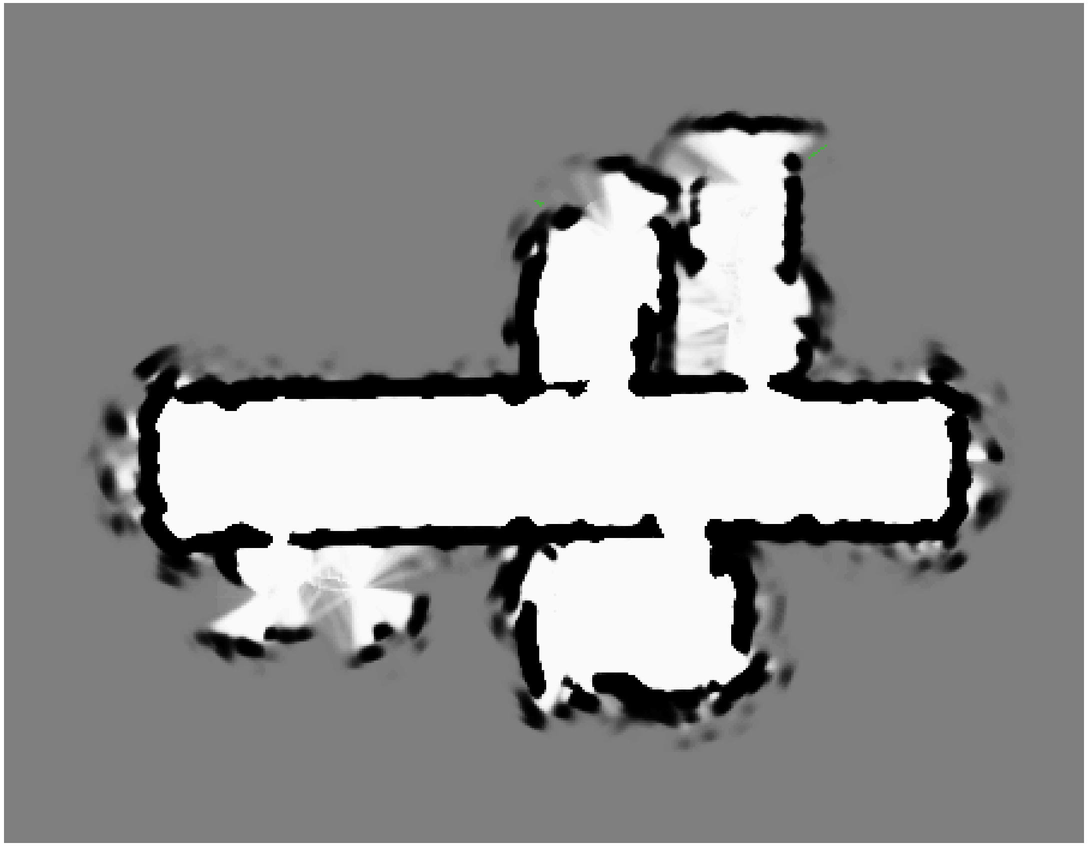
$$\boxed{\begin{aligned} Bel(m_t) &= P(m_t \mid u_1, z_2 \ldots, u_{t-1}, z_t) \\ &= \prod_{x,y} Bel(m_t^{[xy]}) \end{aligned}}$$

  - Robot positions are known!

Incremental Updating of Occupancy Grids (Example)

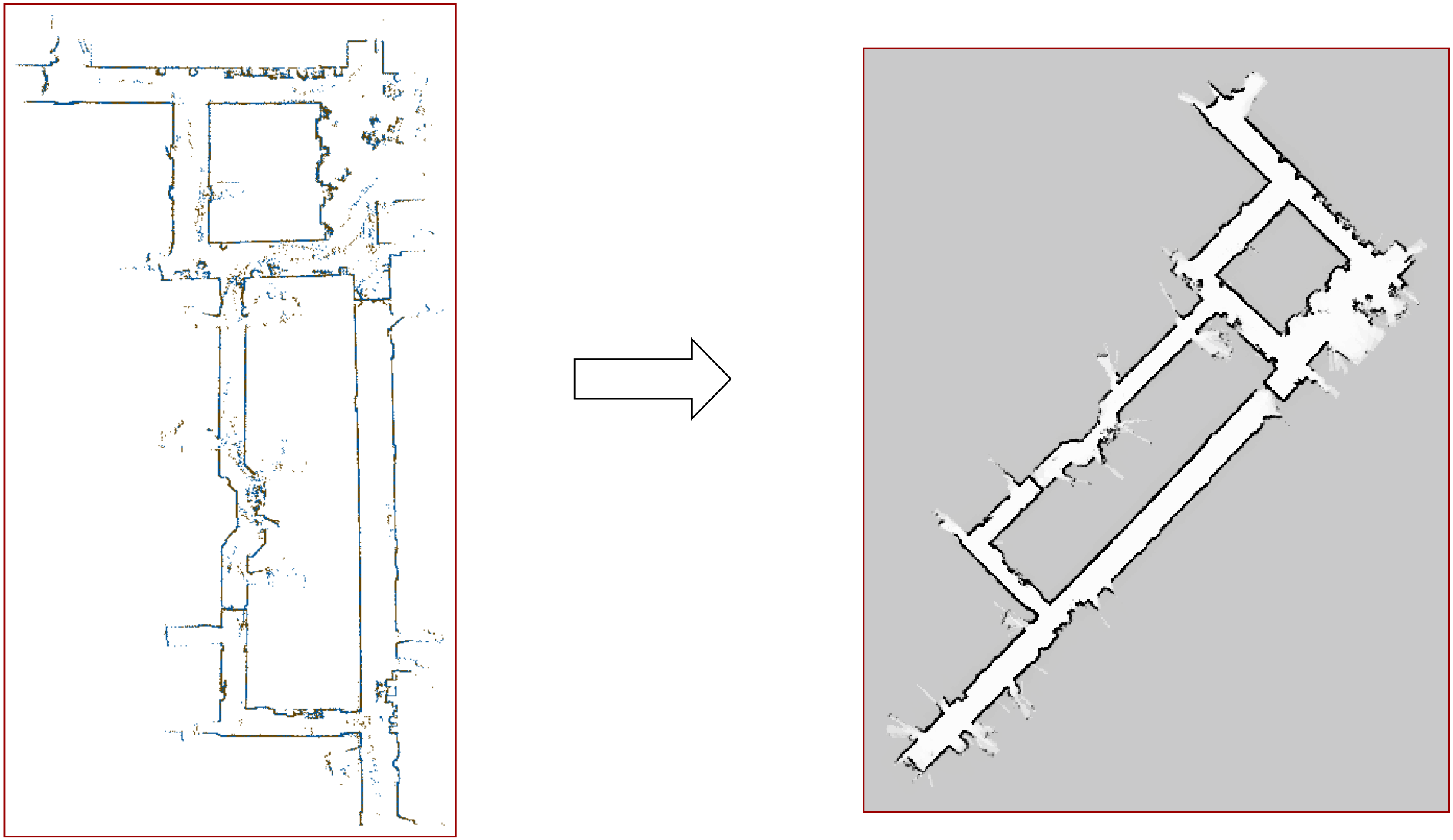


Resulting Map Obtained with Ultrasound Sensors

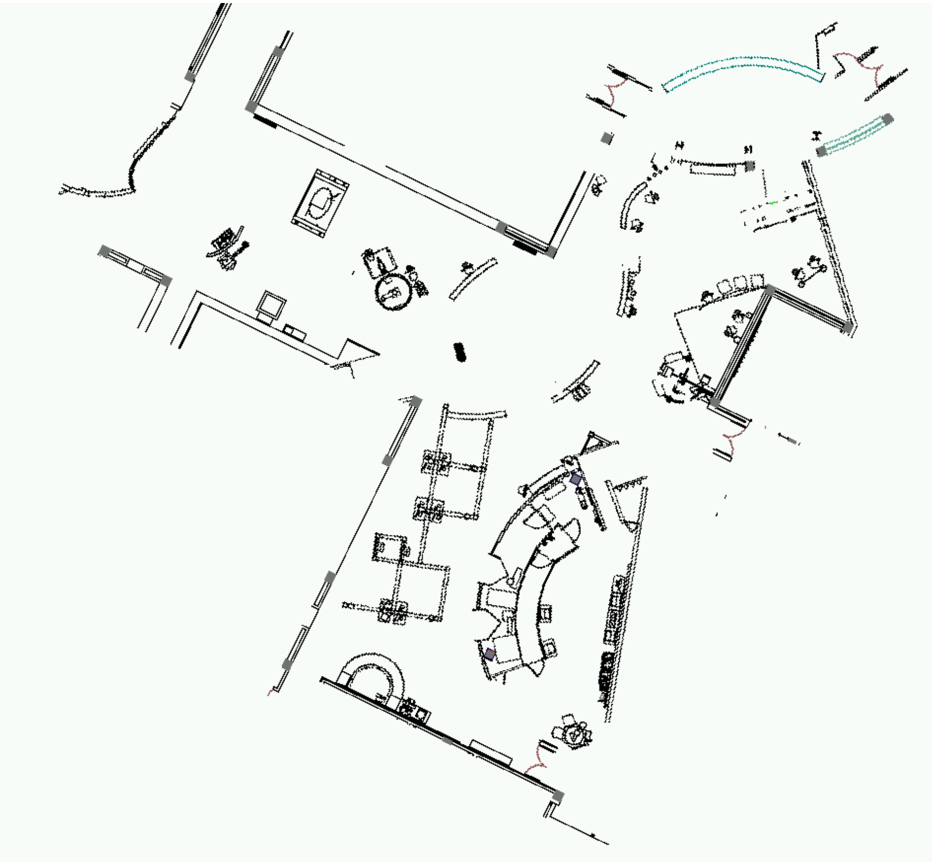



INSTITUTO
SUPERIOR
TÉCNICO
Real robot mapping using Sonars
Bernardo Costa, 65319
Diogo Rolo, 65336
Mariana Lopes, 65433
January 2012

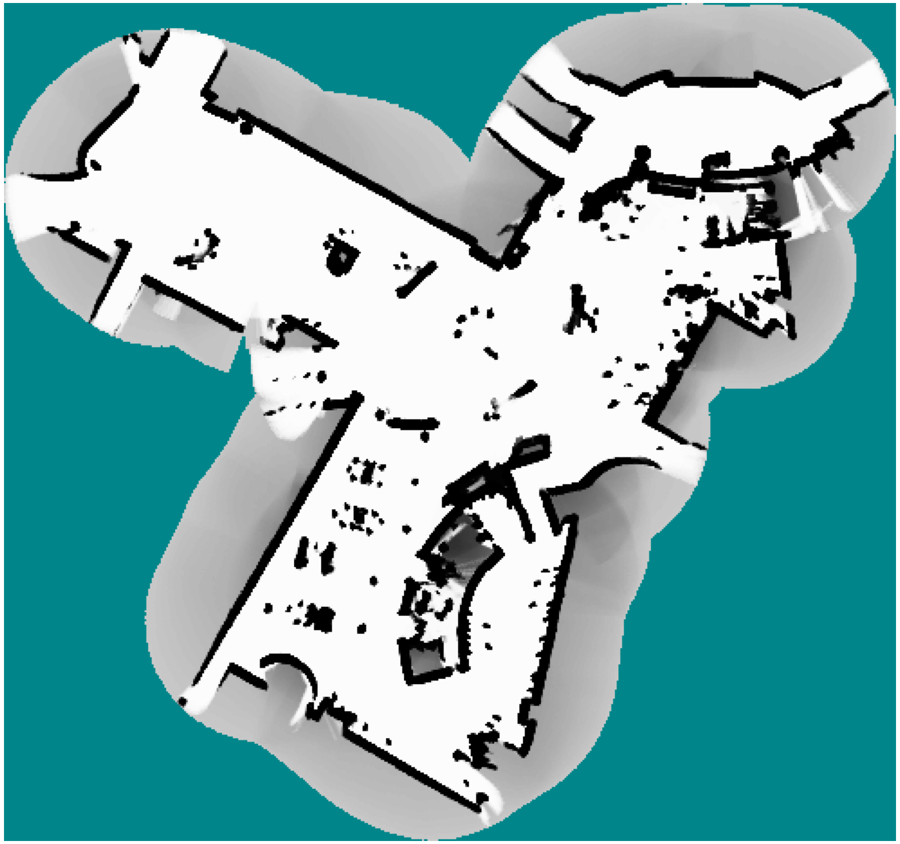
Occupancy Grids: From scans to maps



Tech Museum, San Jose



CAD map



occupancy grid map

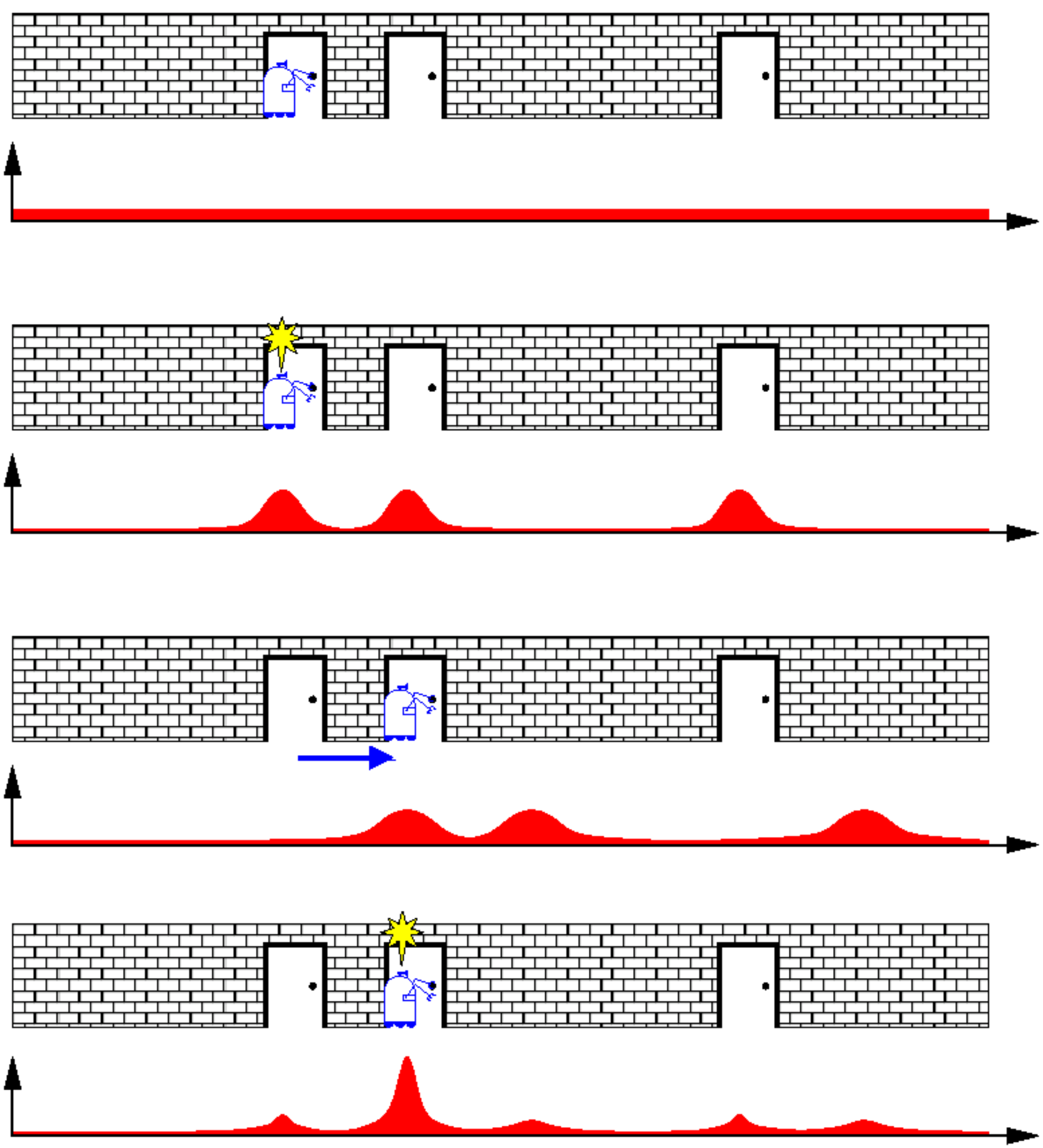
Introduction to Mobile Robotics

Bayes Filter – Discrete Filters

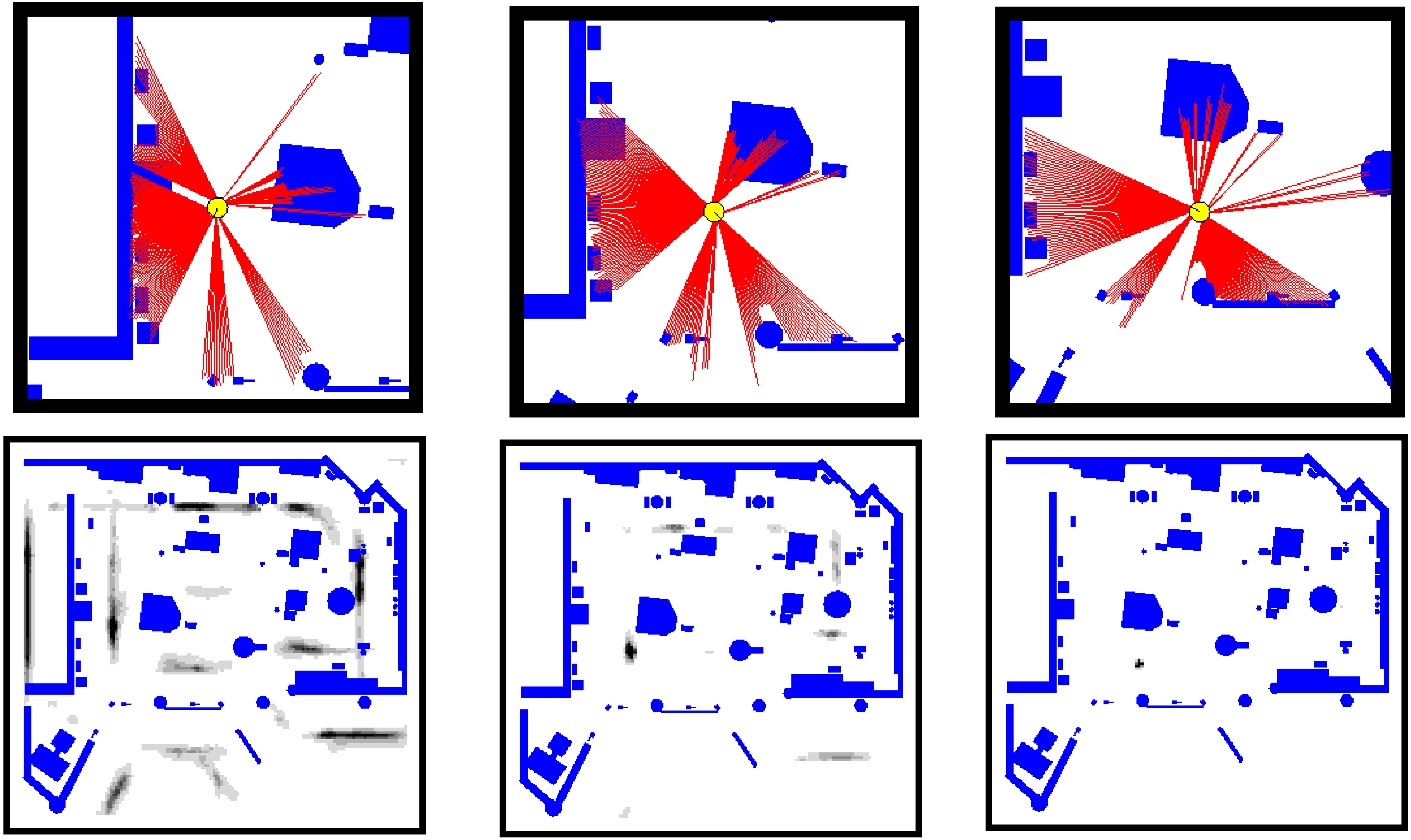
Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

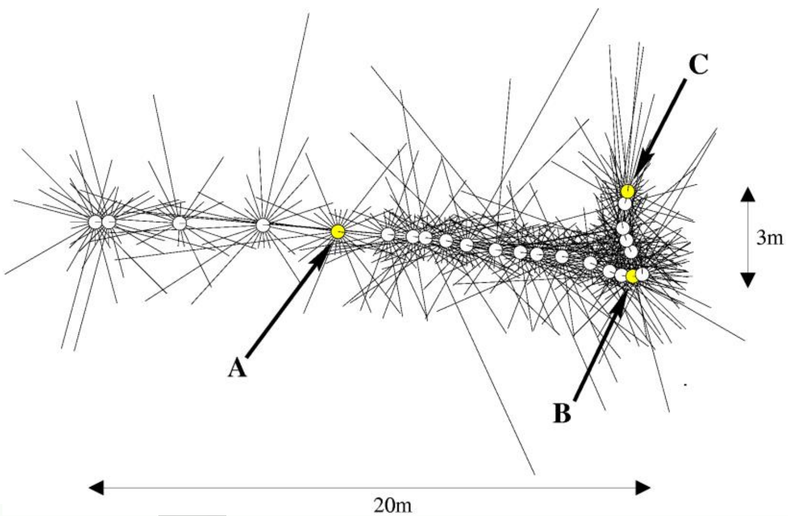
$$Bel(x \mid z, u) \;=\; \alpha\, p(z \mid x) \int_{x'} p(x \mid u, x')\; Bel(x')dx'$$

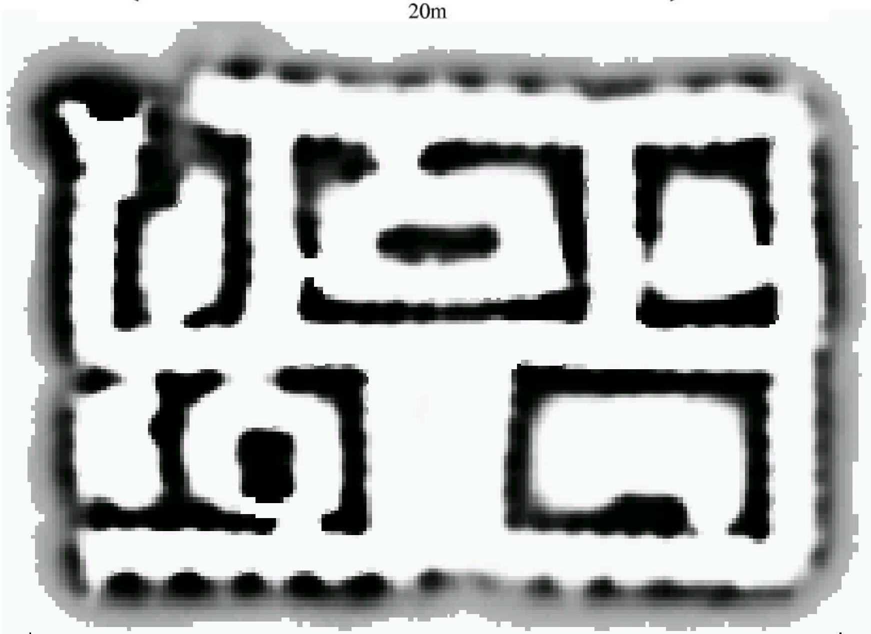


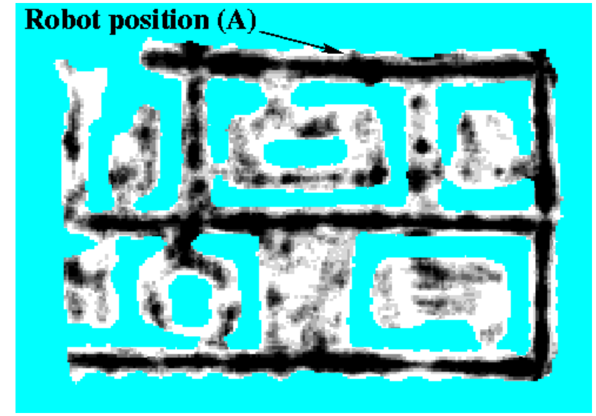
Grid-based Localization

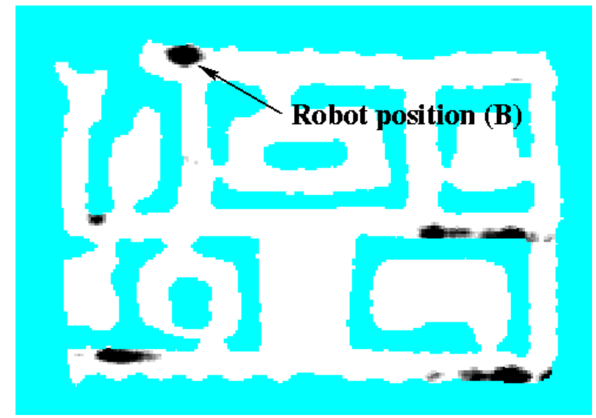


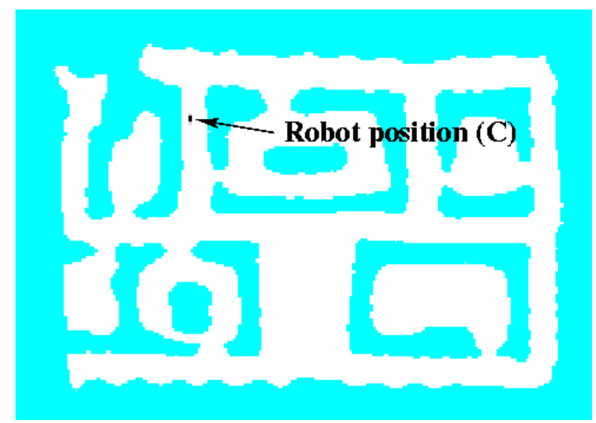
Sonars and Occupancy Grid Map











Introduction to Mobile Robotics

Bayes Filter – Particle Filter and Monte Carlo Localization

Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

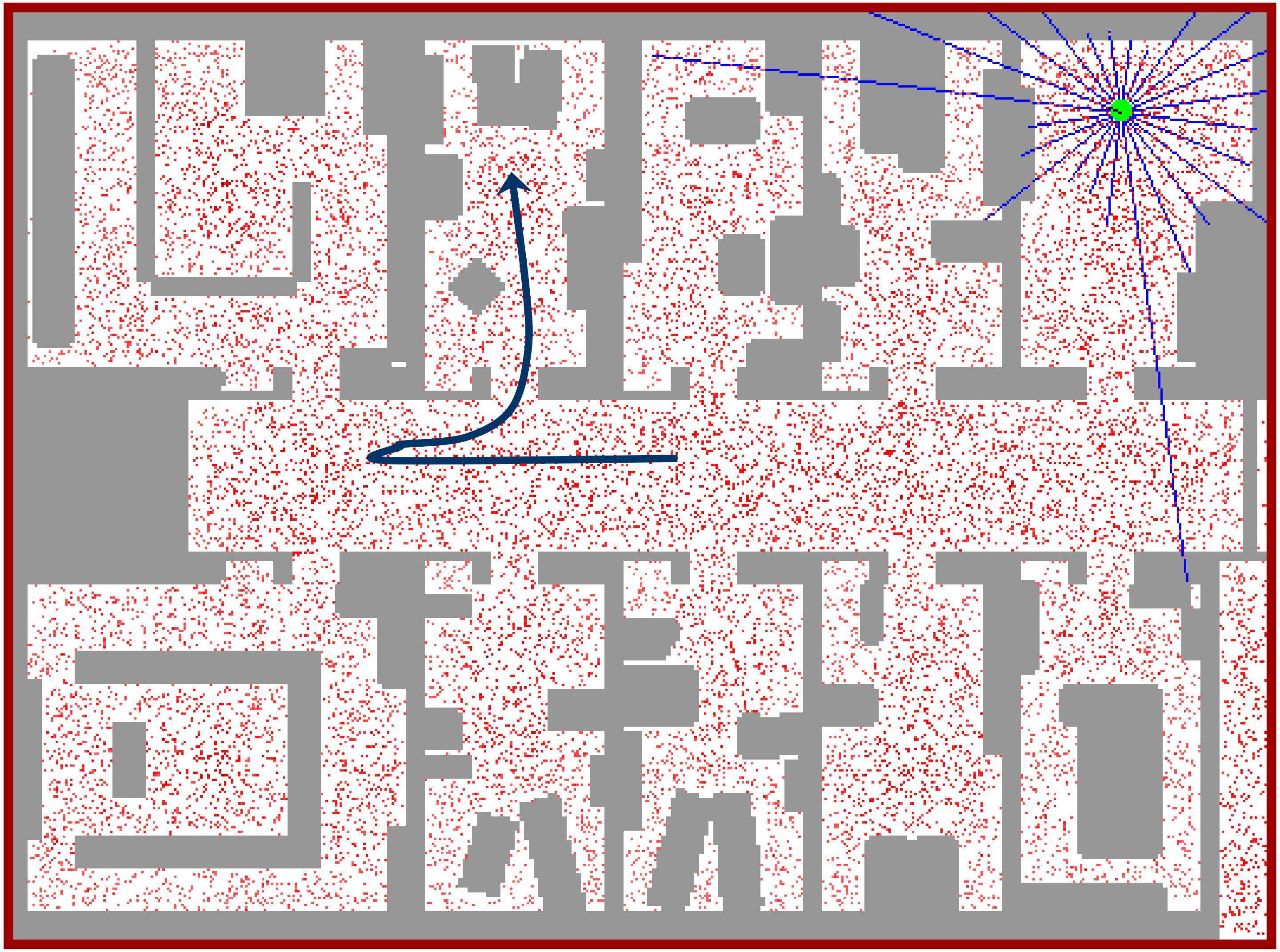
Motivation

- Recall: Discrete filter
  - Discretize the continuous state space
  - High memory complexity
  - Fixed resolution (does not adapt to the belief)

- Particle filters are a way to **efficiently** represent **non-Gaussian distribution**

- Basic principle
  - Set of state hypotheses ("particles")
  - Survival-of-the-fittest

Sample-based Localization (sonar)



Mathematical Description

- Set of weighted samples

$$S = \left\{ \left\langle s^{[i]}, w^{[i]} \right\rangle \mid i = 1, \ldots, N \right\}$$

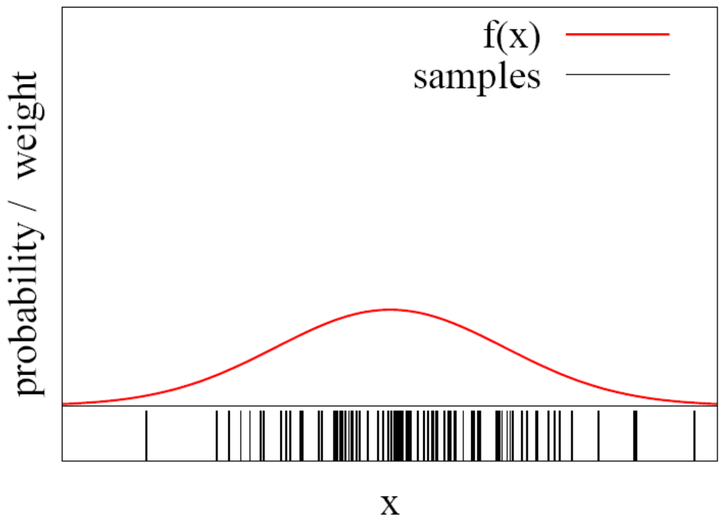
State hypothesis Importance weight

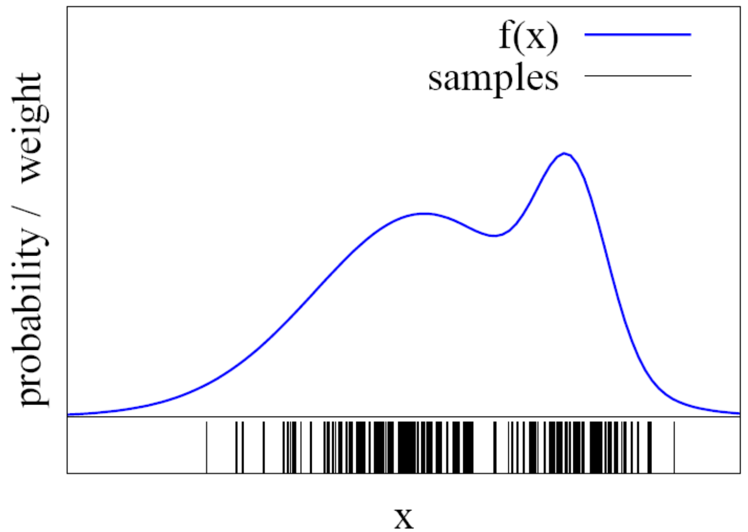
- The samples represent the posterior

$$p(x) = \sum_{i=1}^{N} w_i \cdot \delta_{s^{[i]}}(x)$$

Function Approximation

- Particle sets can be used to approximate functions



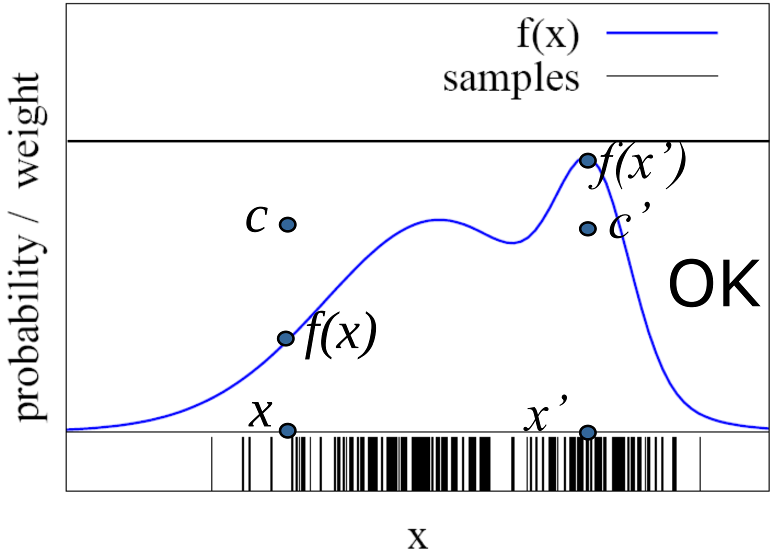


- The more particles fall into an interval, the higher the probability of that interval

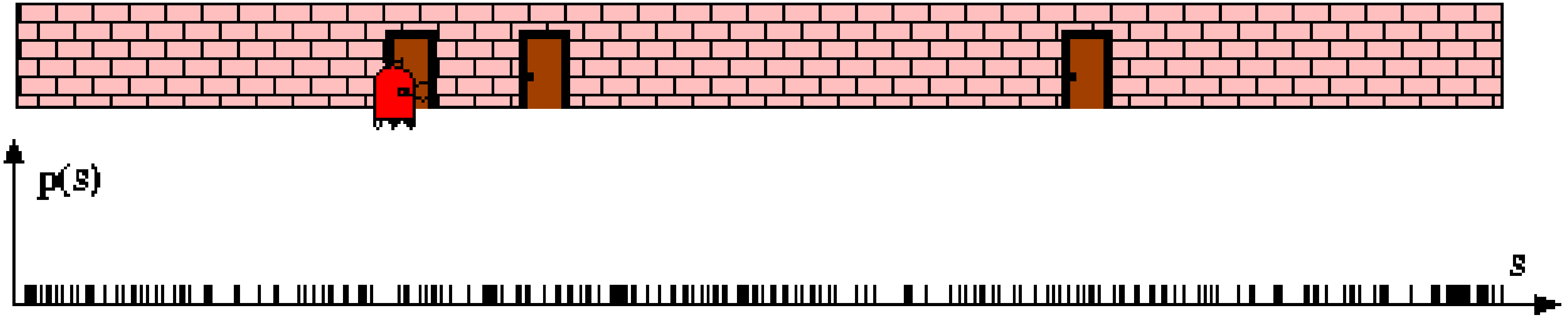
- How to draw samples form a function/distribution?

Rejection Sampling

- Let us assume that $f(x) < 1$ for all x
- Sample x from a uniform distribution
- Sample c from [0,1]
- if $f(x) > c$ keep the sample
  otherwise reject the sampe



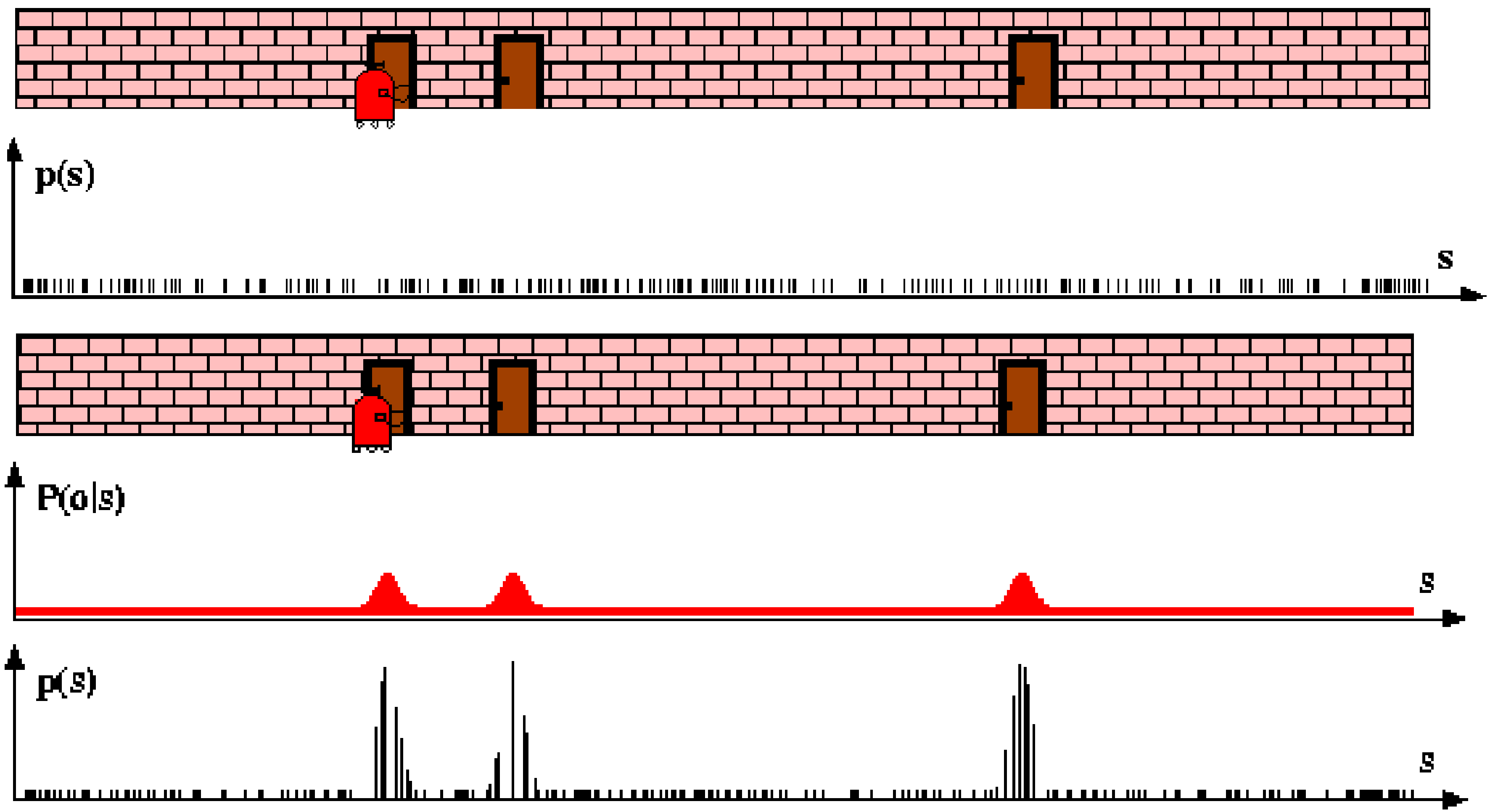
Particle Filters



Sensor Information: Importance Sampling

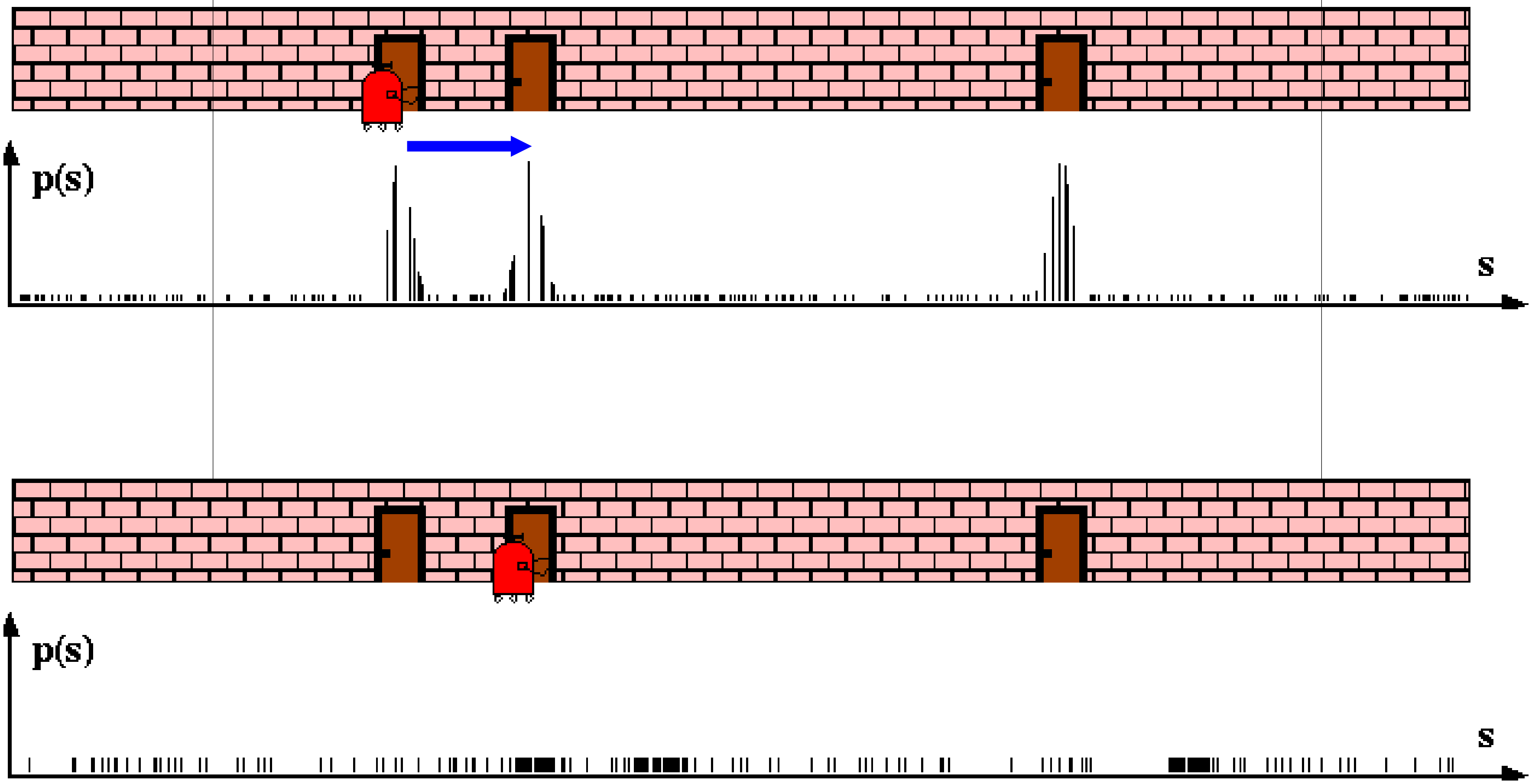
$$Bel(x) \leftarrow \alpha\ p(z\,|\,x)\,Bel^{-}(x)$$

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



Robot Motion

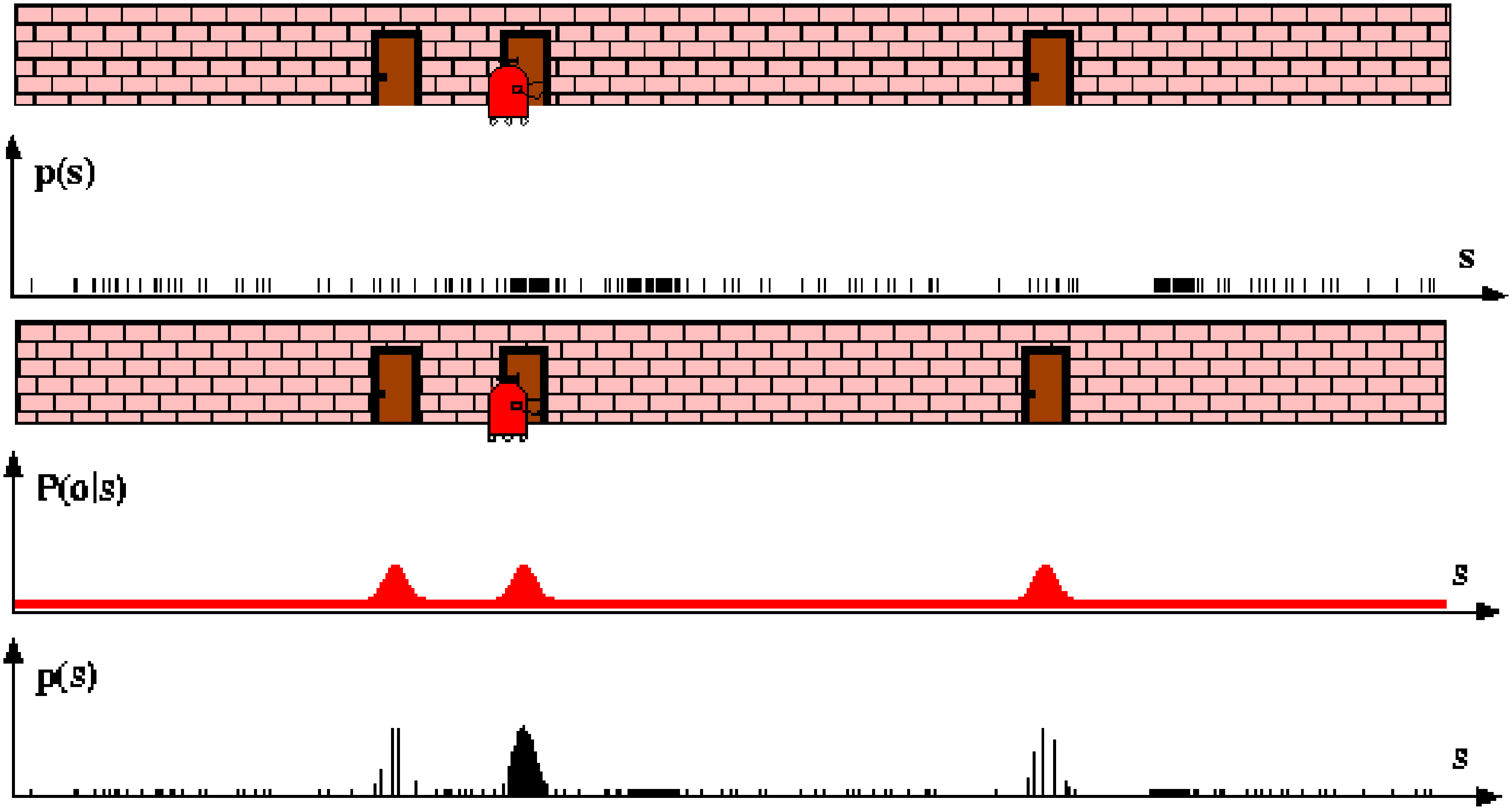
$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



Sensor Information: Importance Sampling

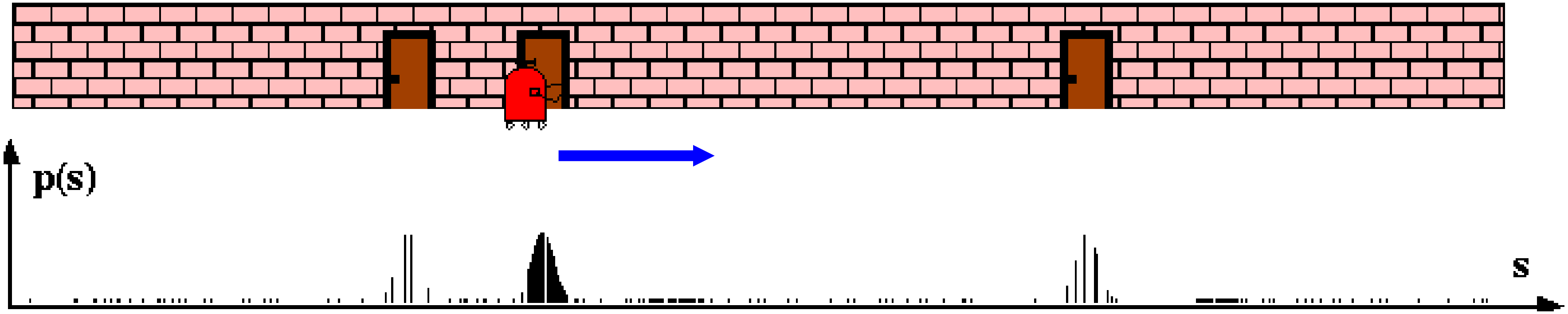
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

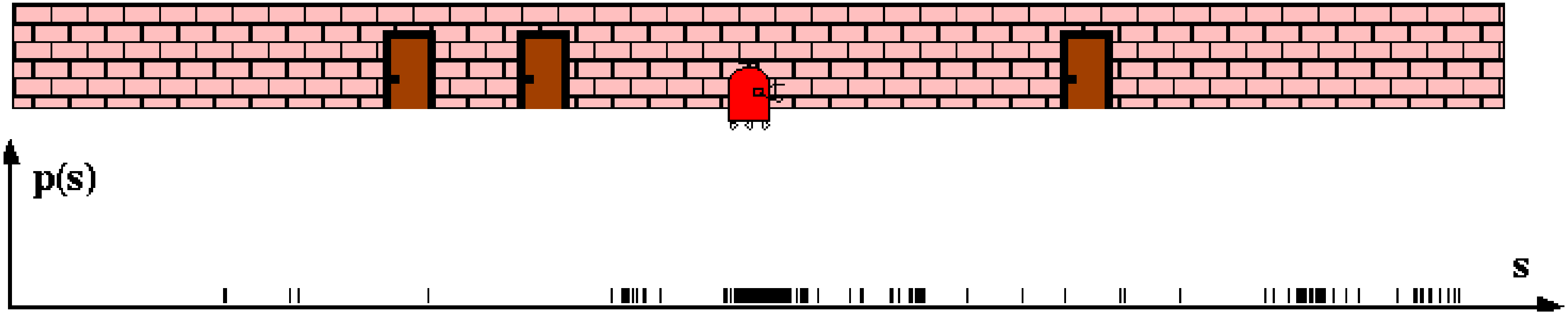
$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



Robot Motion

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





Particle Filter Algorithm

- Sample the next generation for particles using the proposal distribution
- Compute the importance weights :

  $$weight = target\ distribution\ /\ proposal\ distribution$$

- Resampling: “Replace unlikely samples by more likely ones”

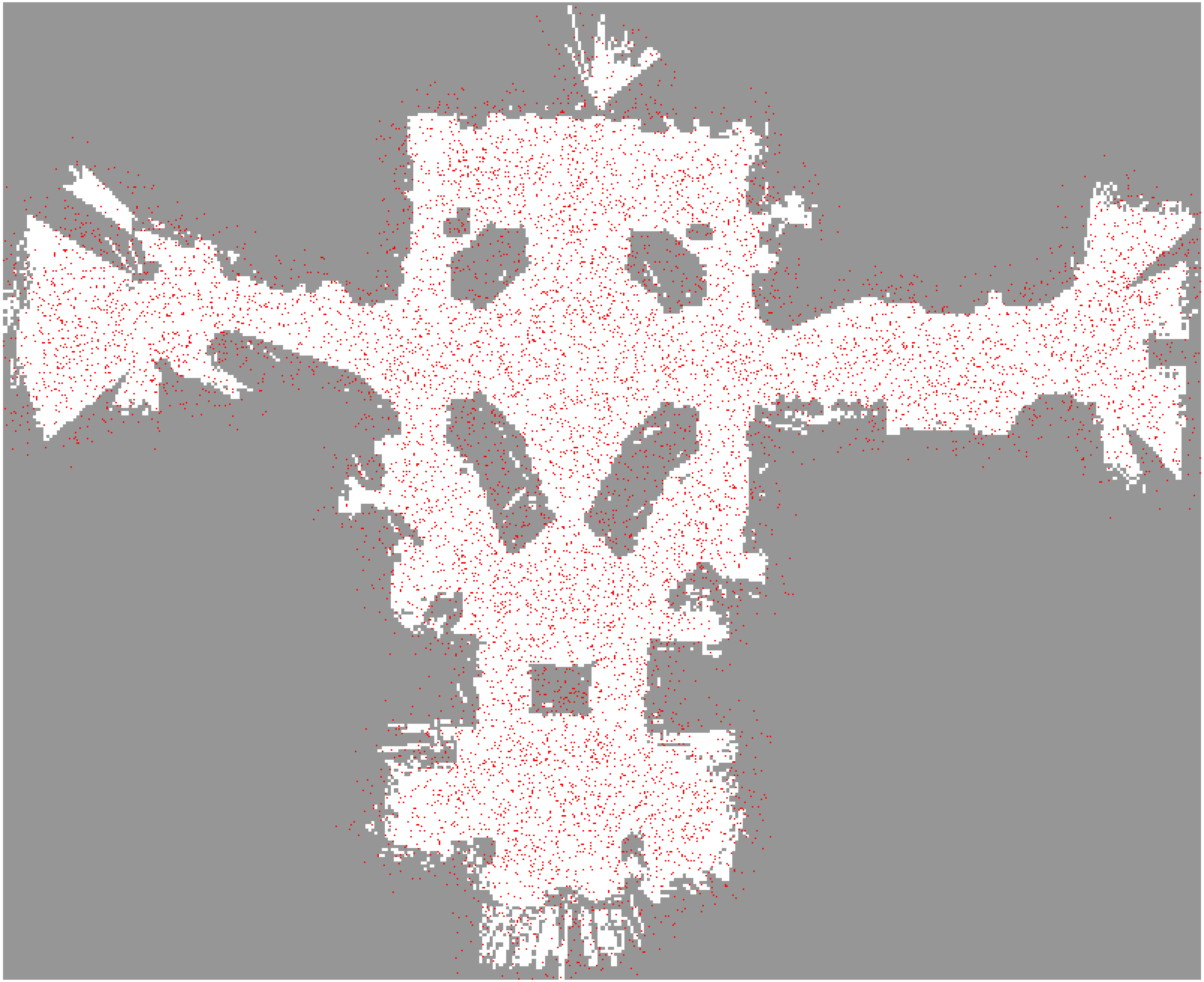
Particle Filter Algorithm

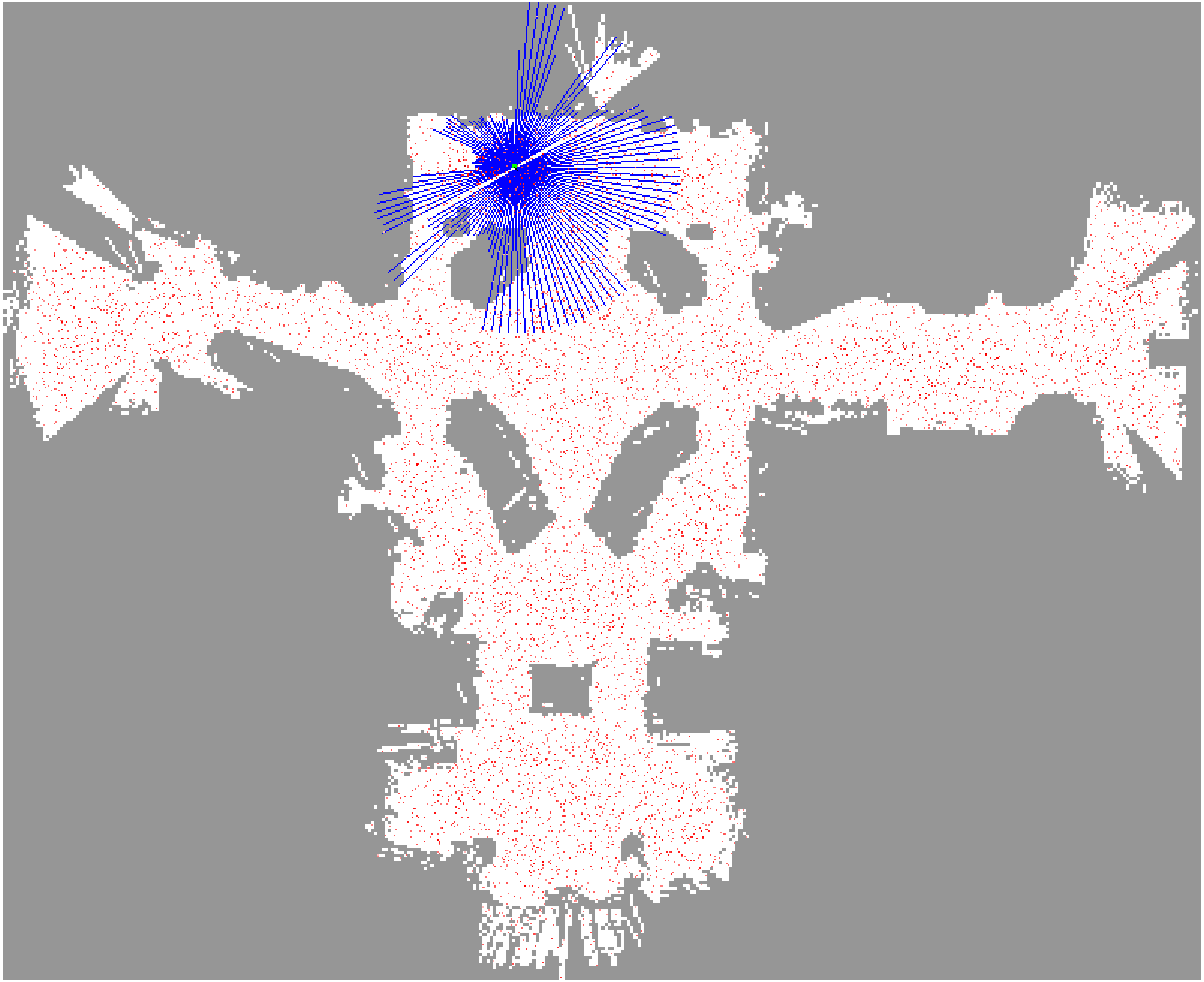
1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Generate new samples***
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
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***

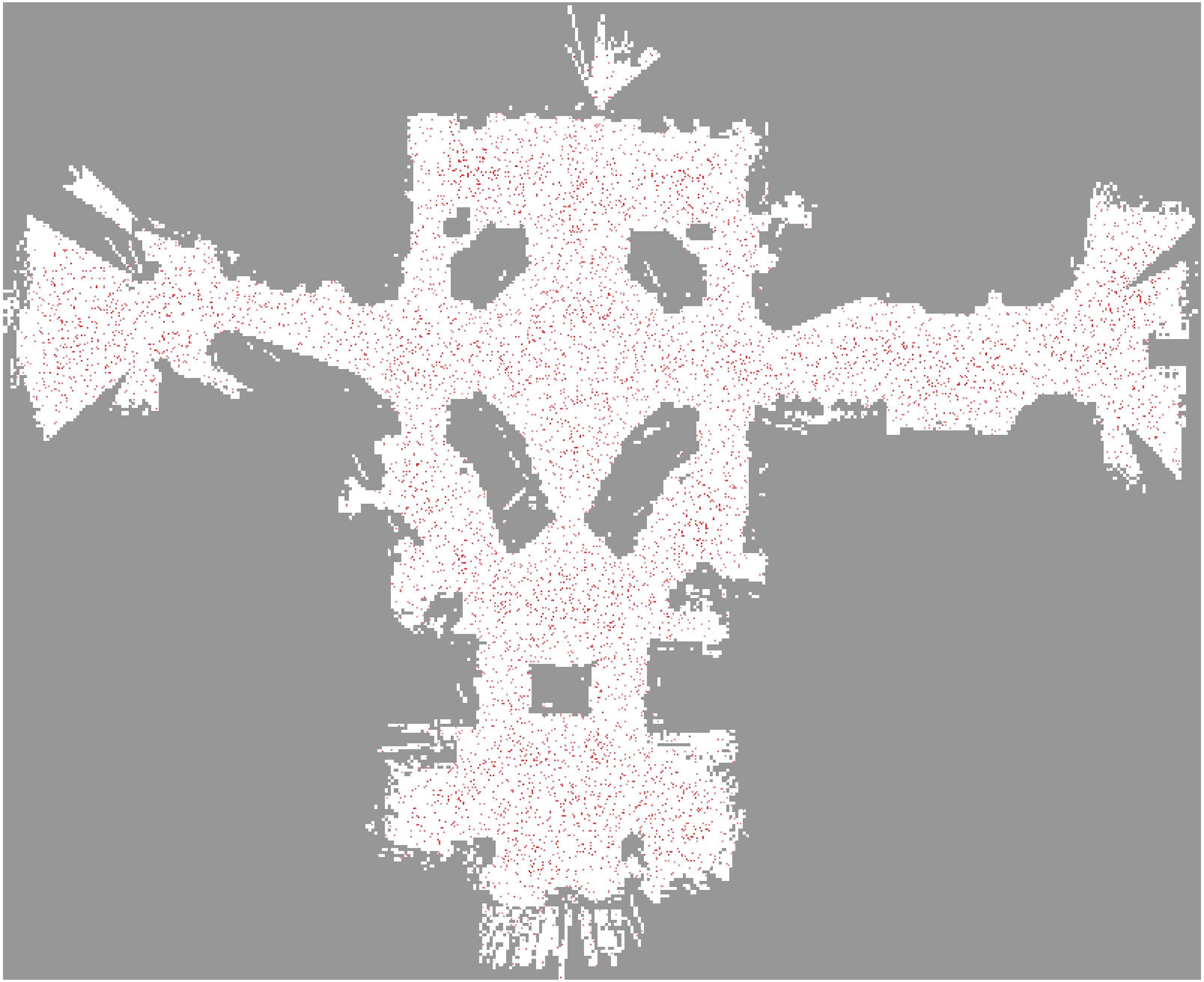
Mobile Robot Localization

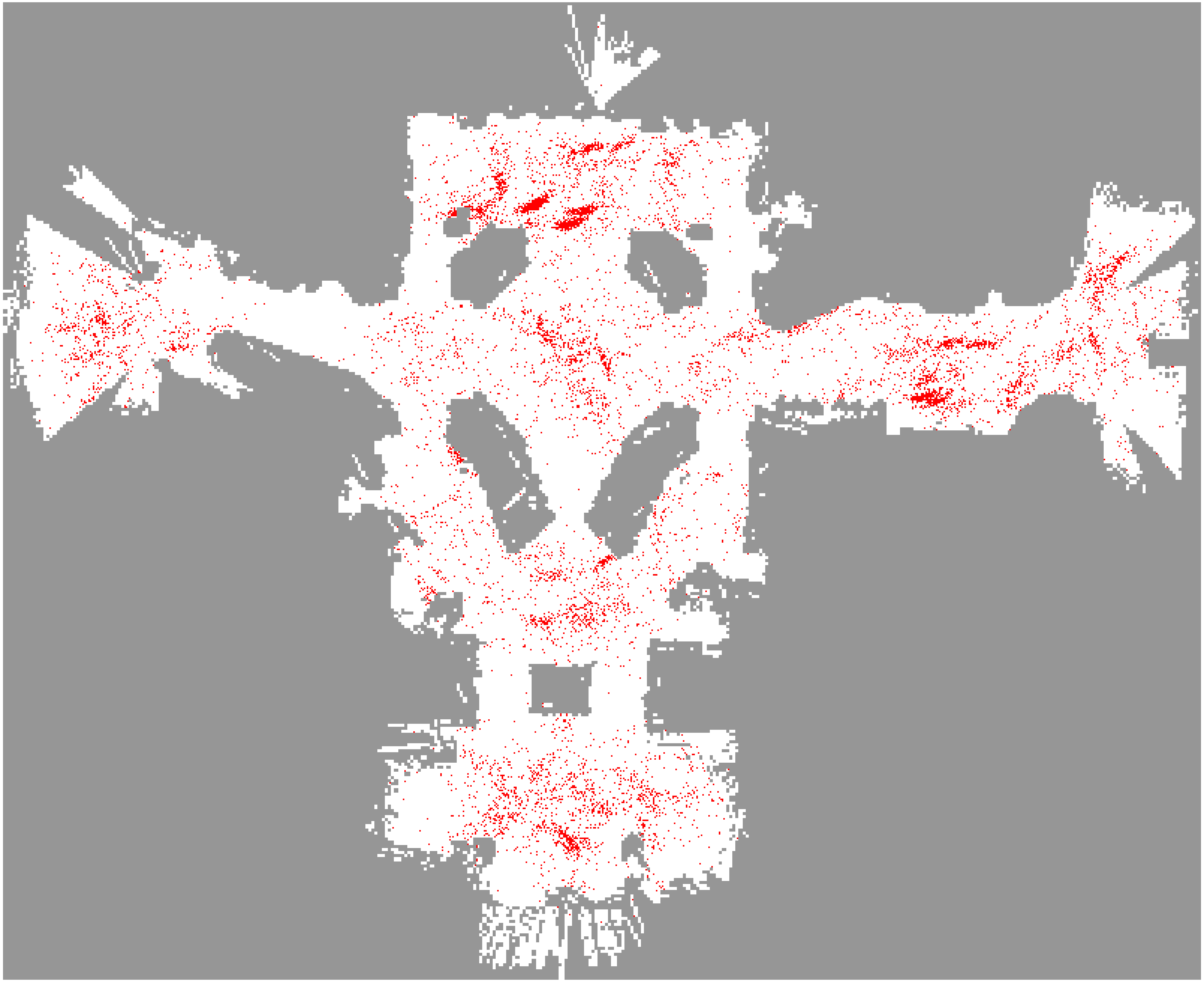
- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute the importance weight (correction step)

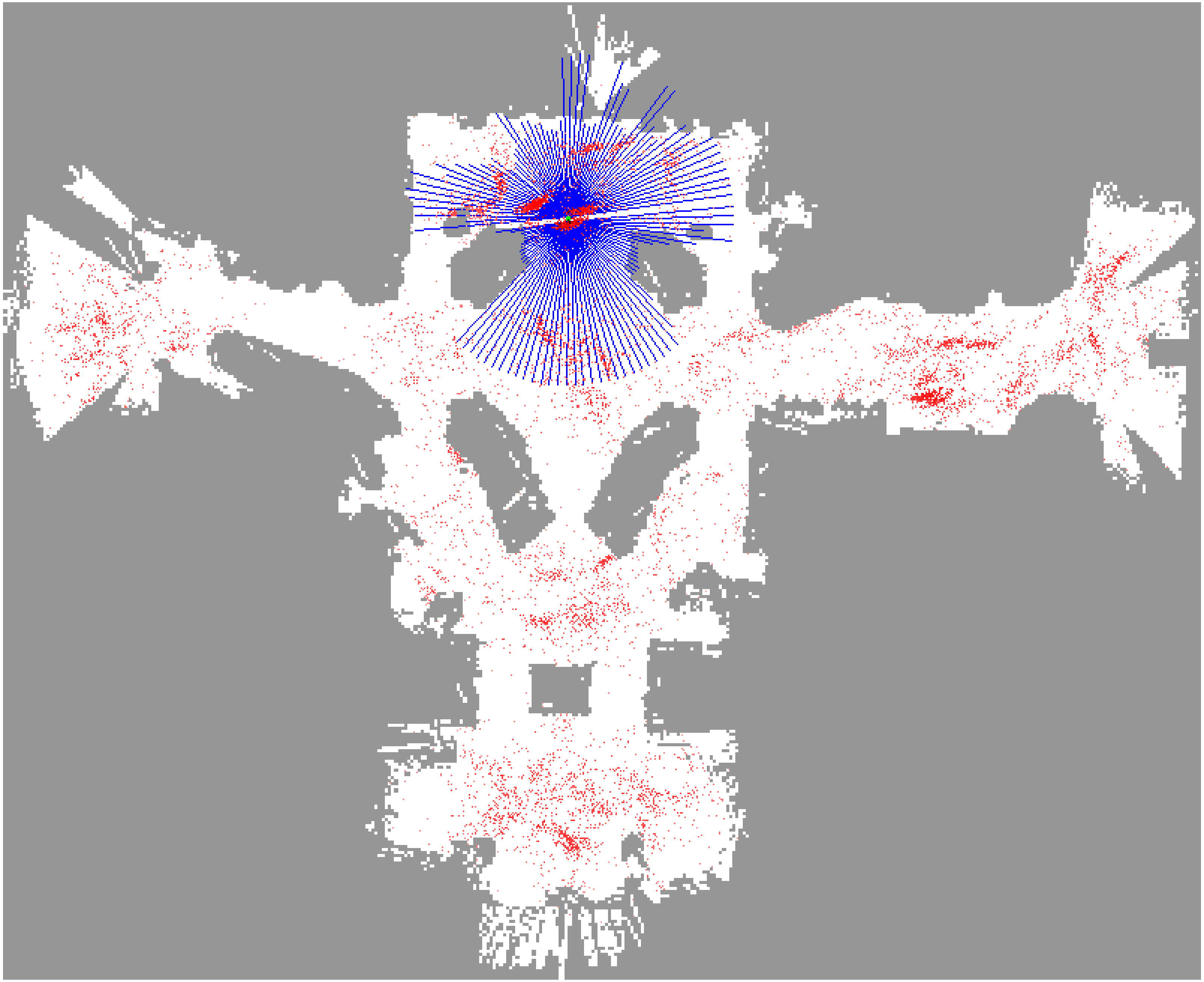
[For details, see PDF file on the lecture web page]

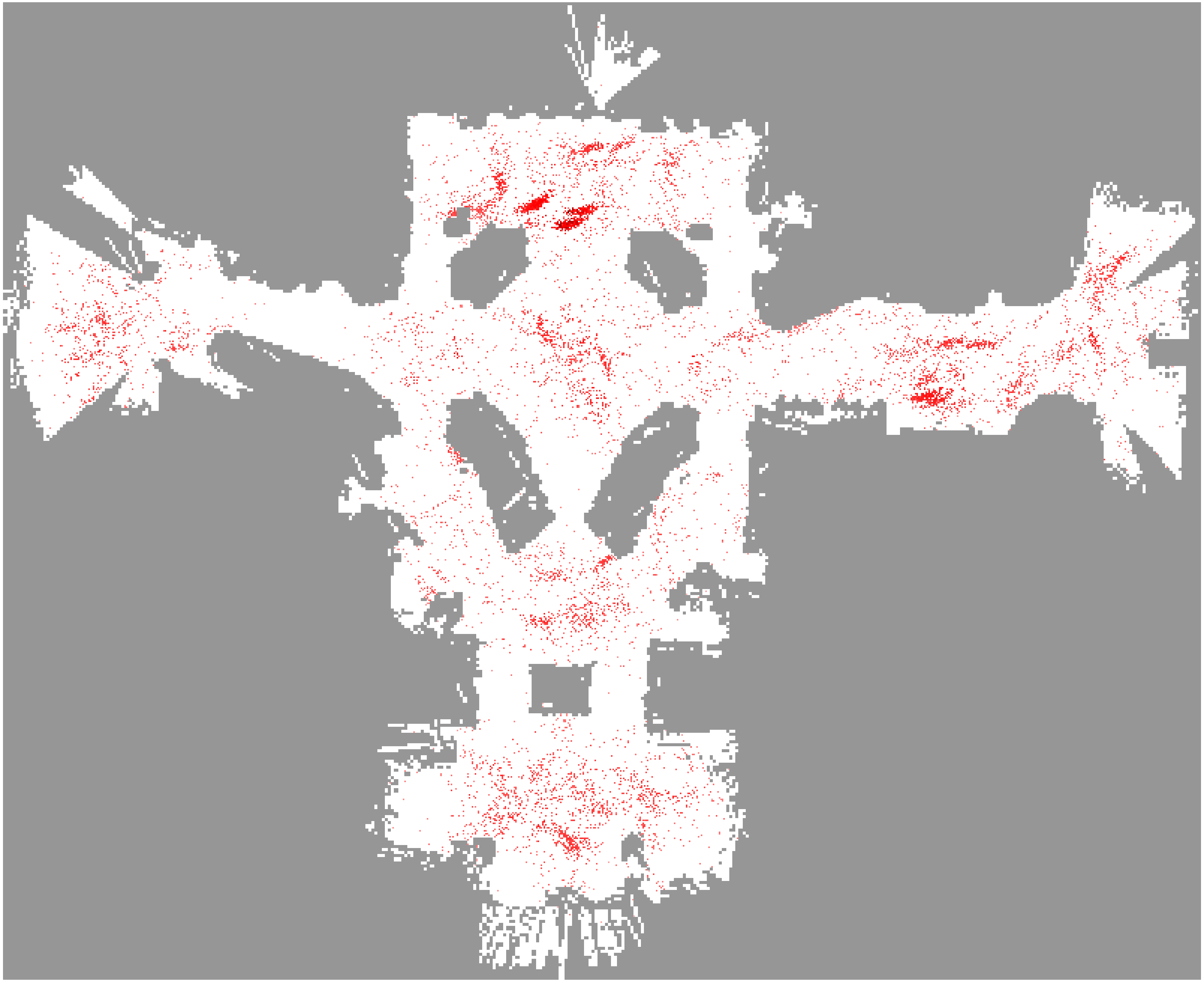


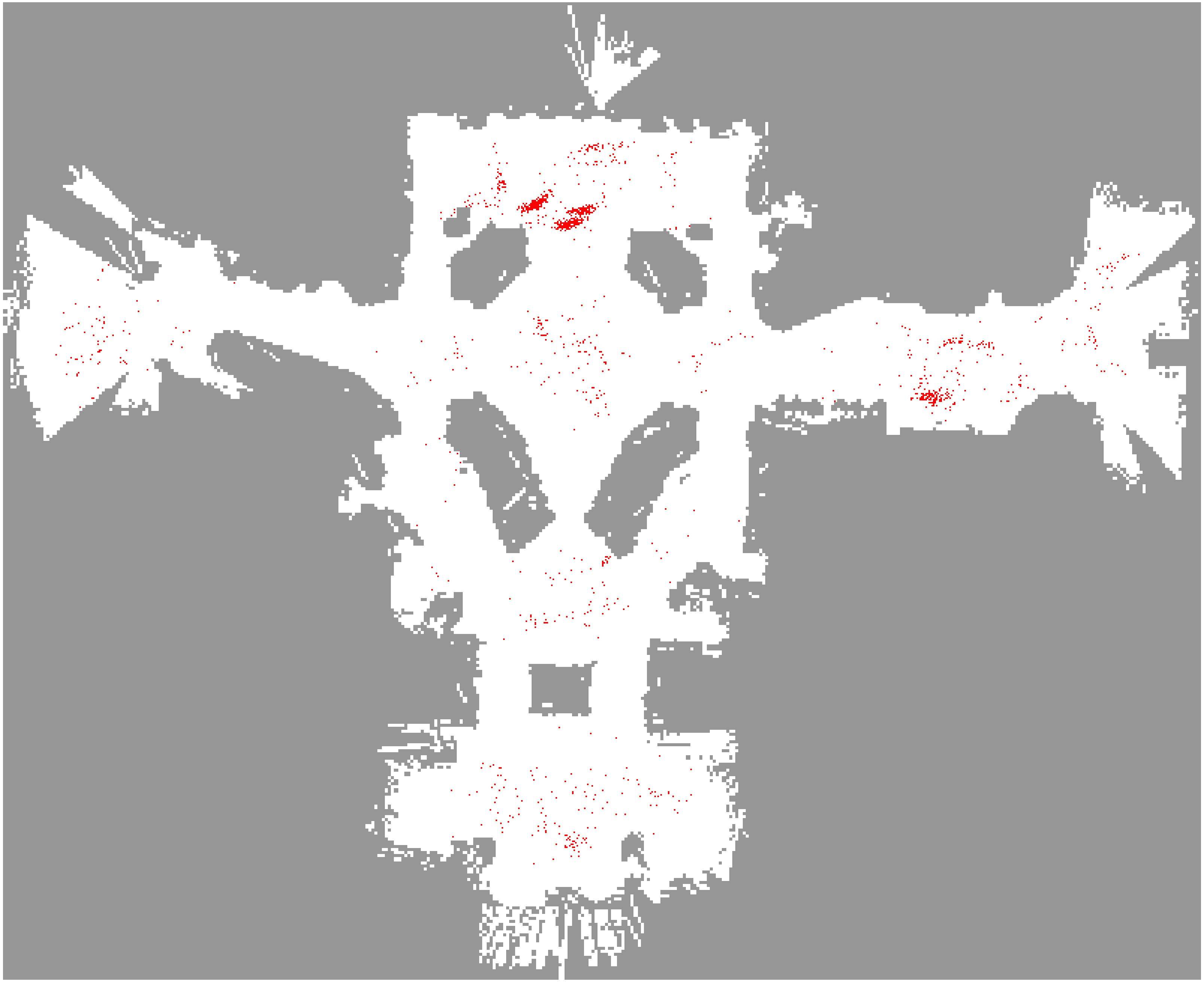


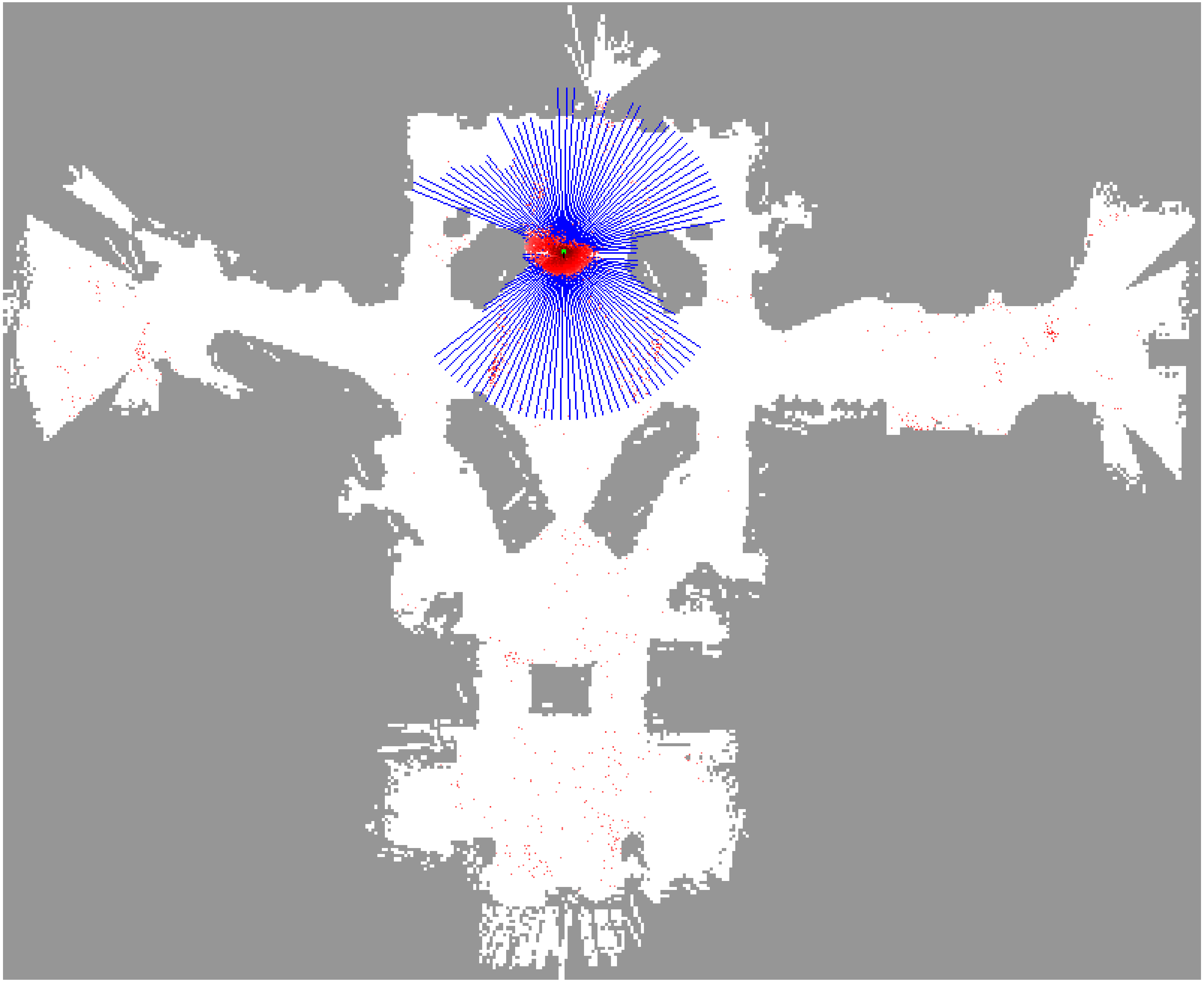


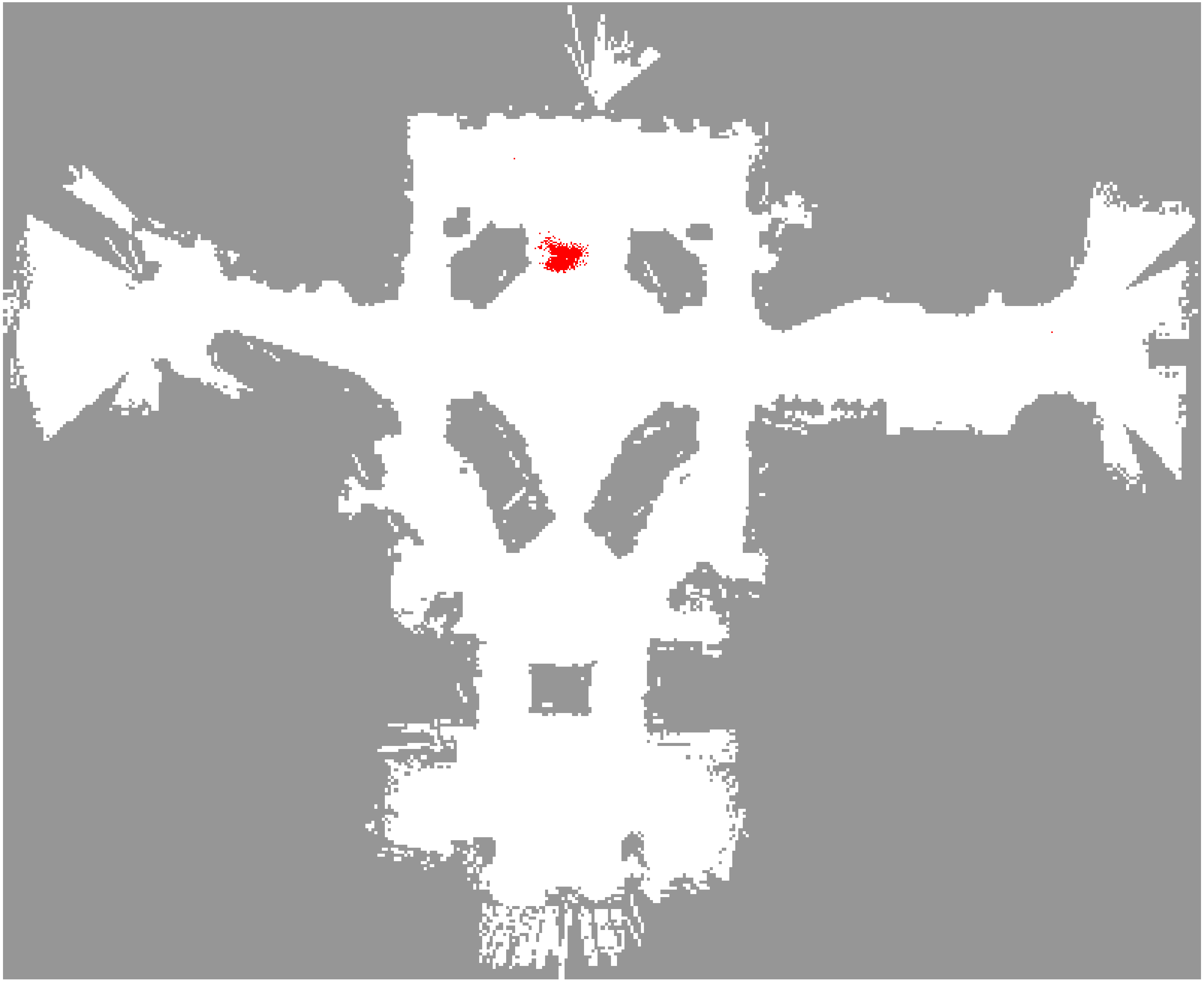


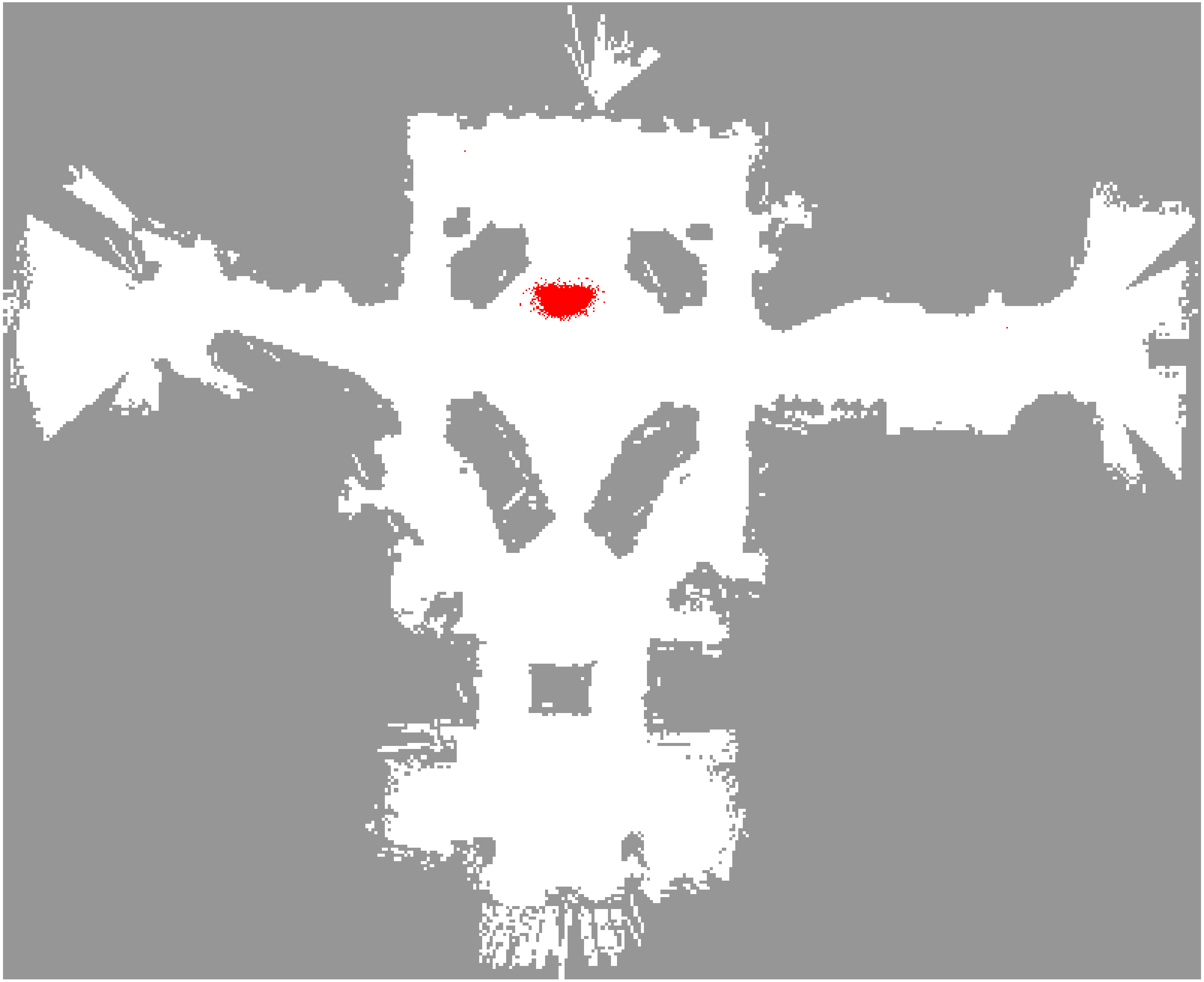


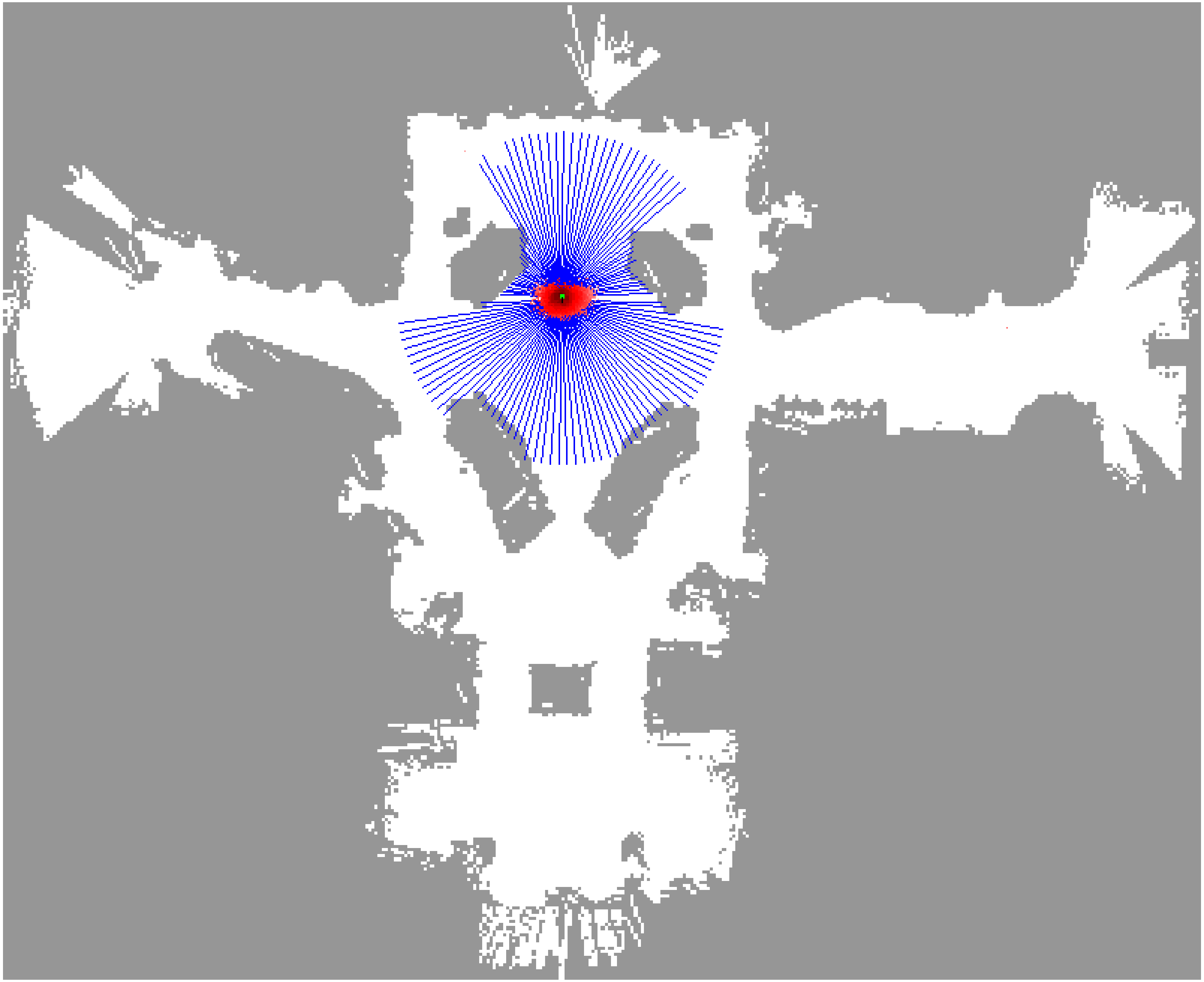


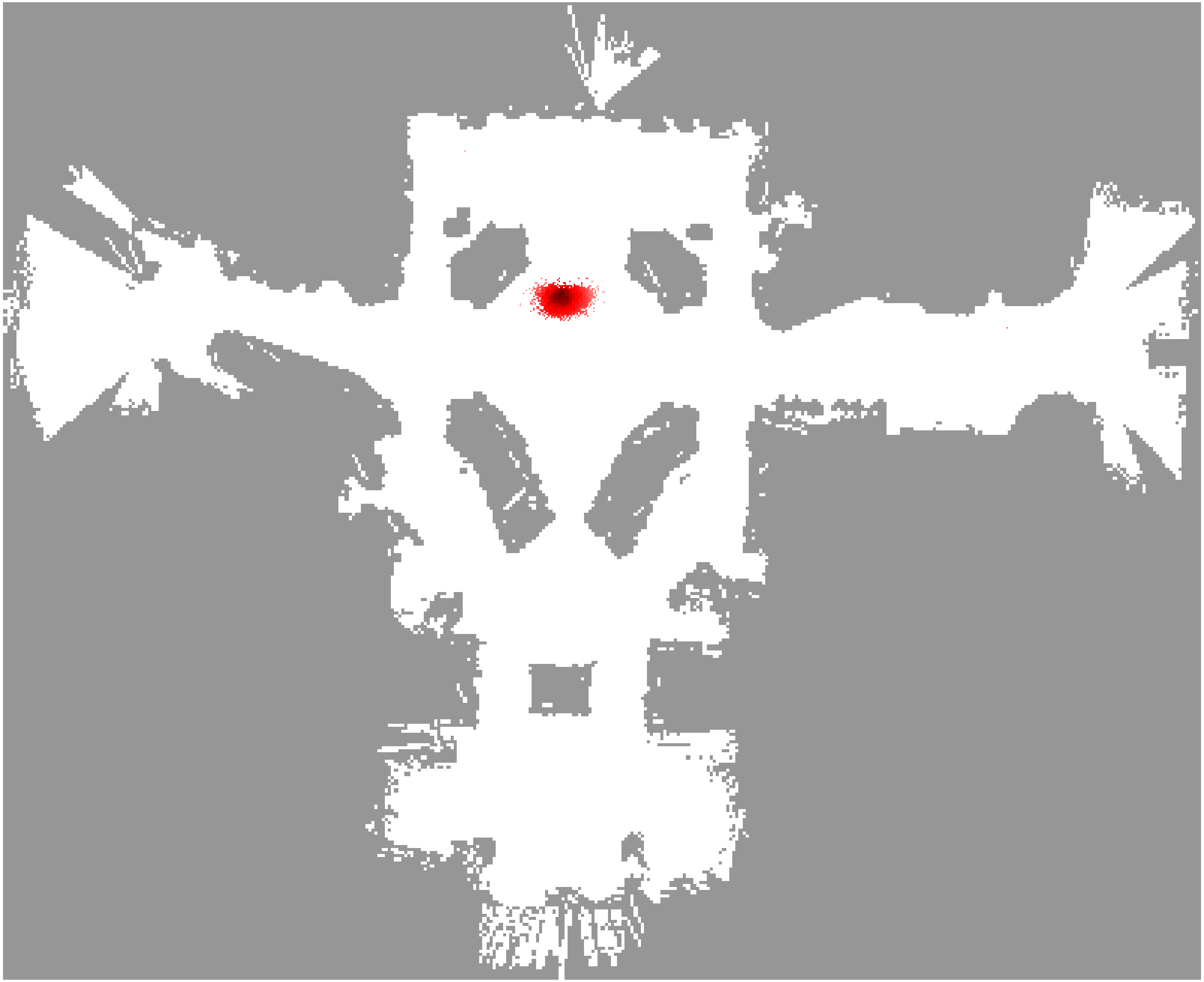


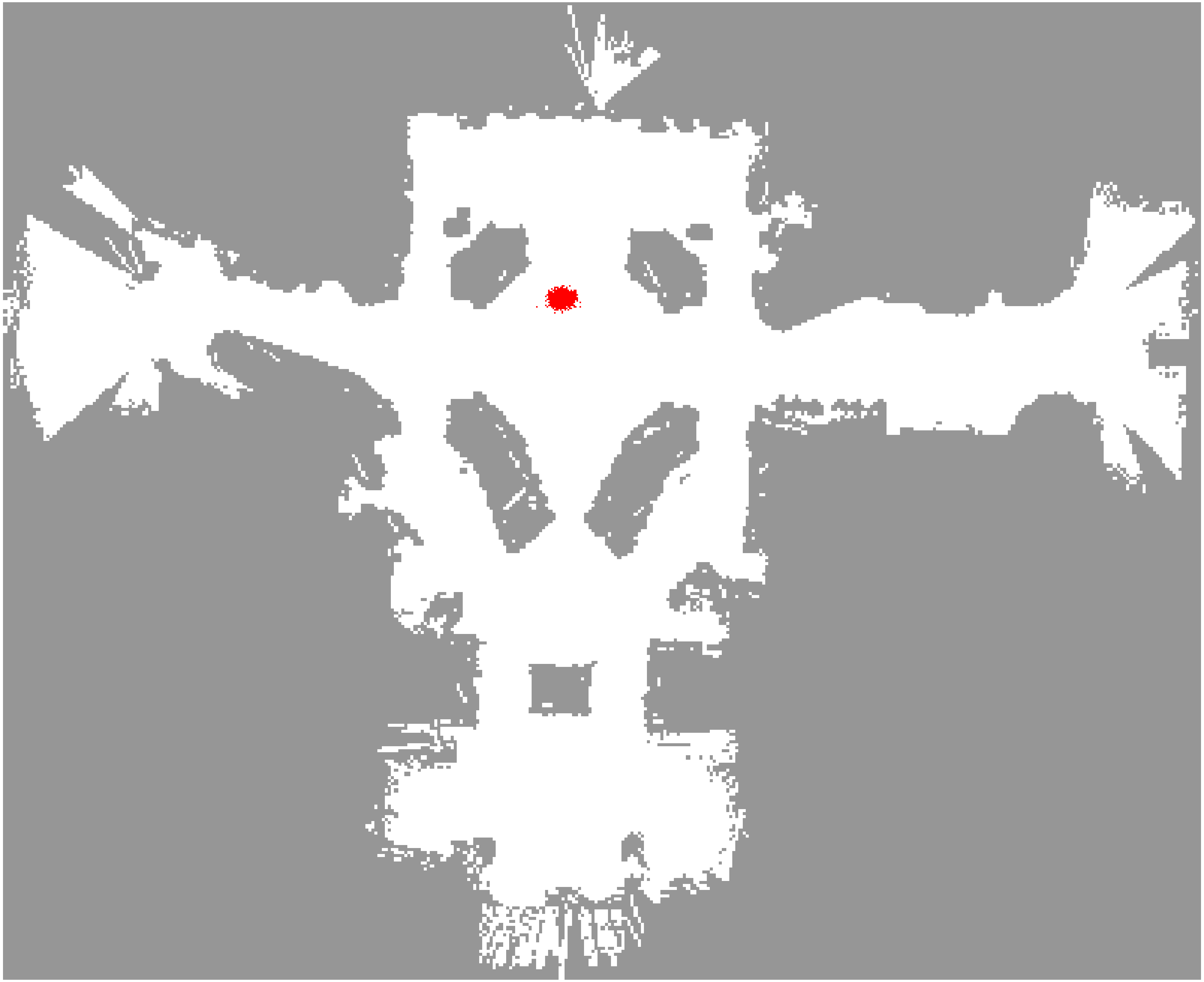


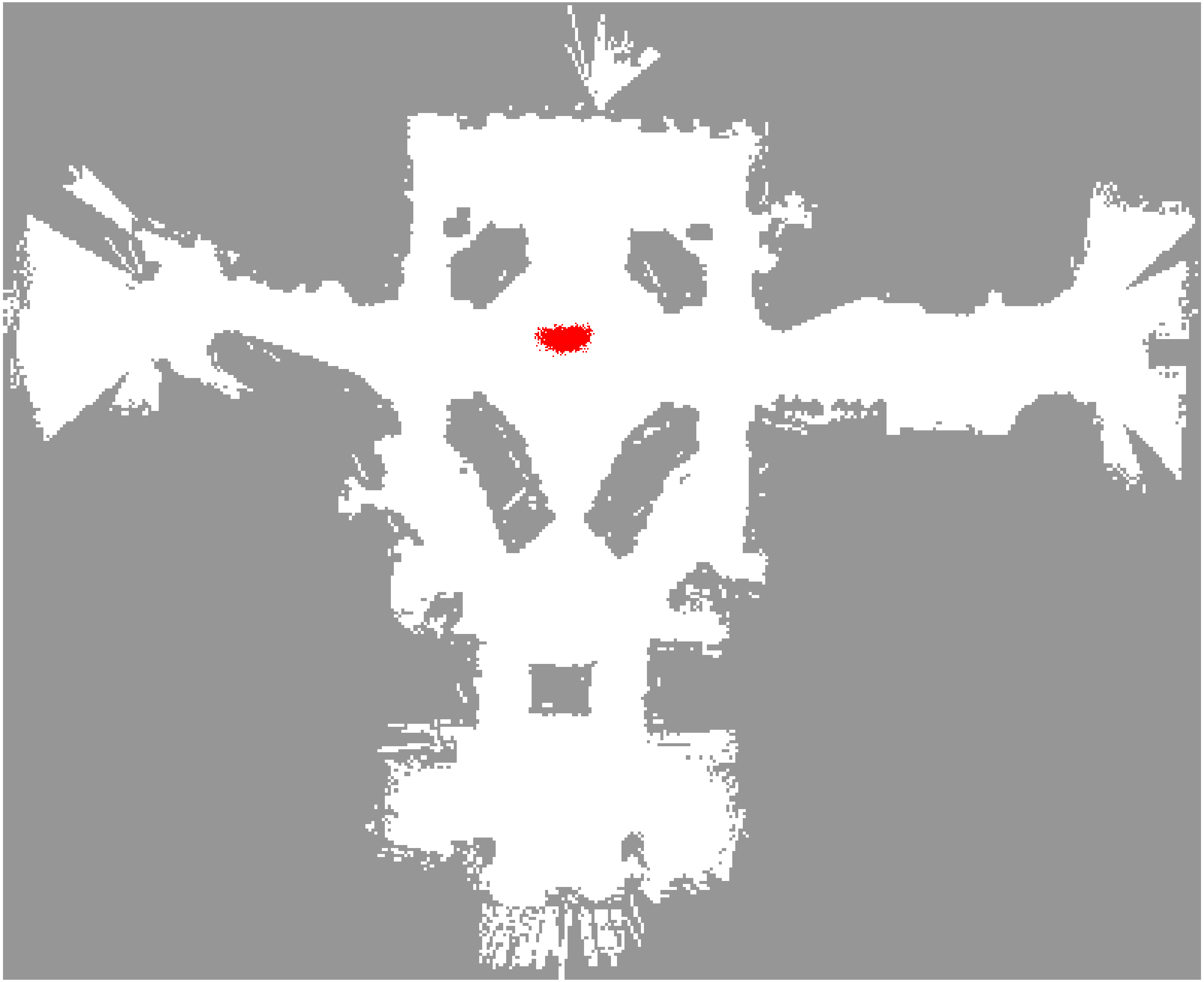


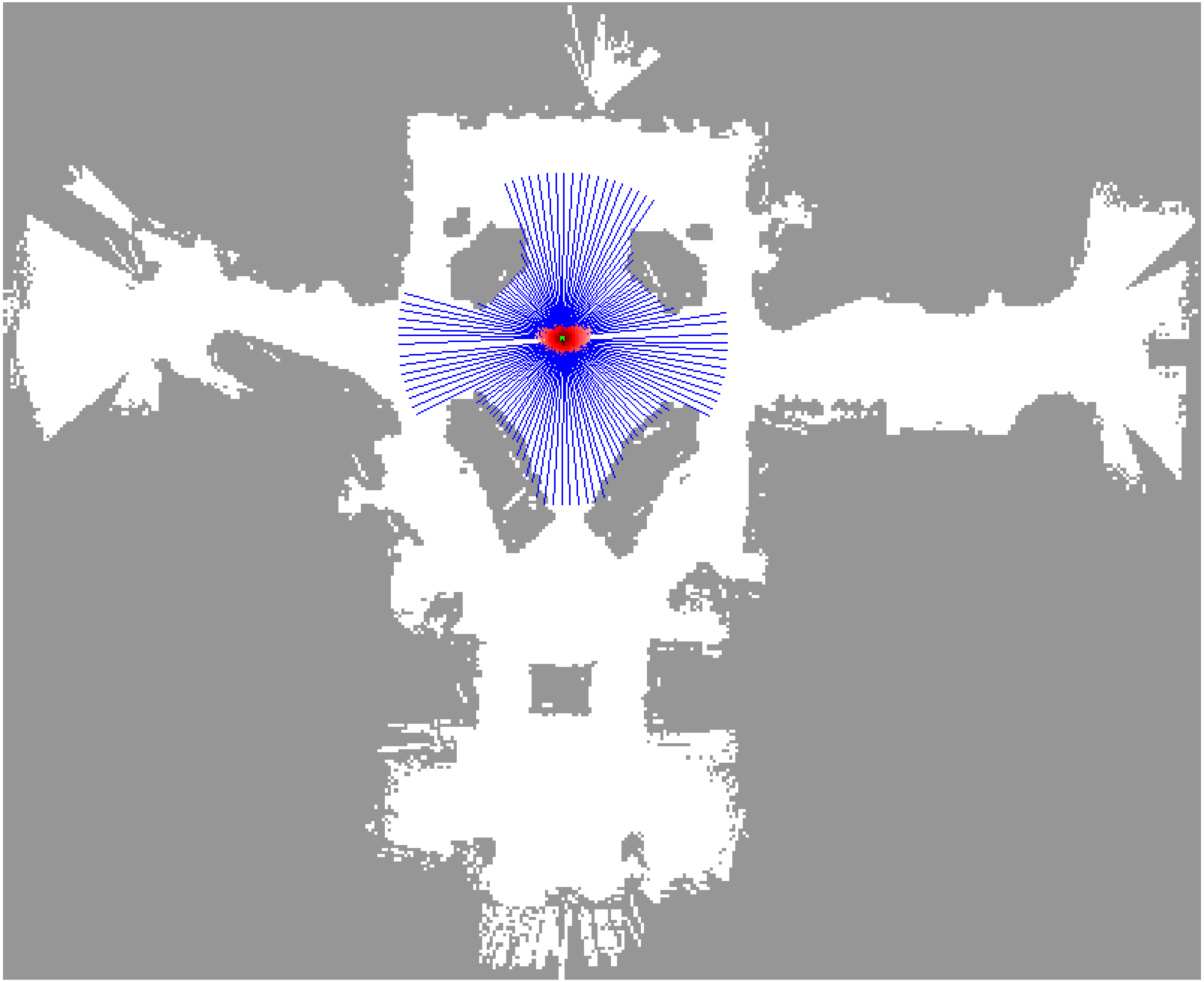


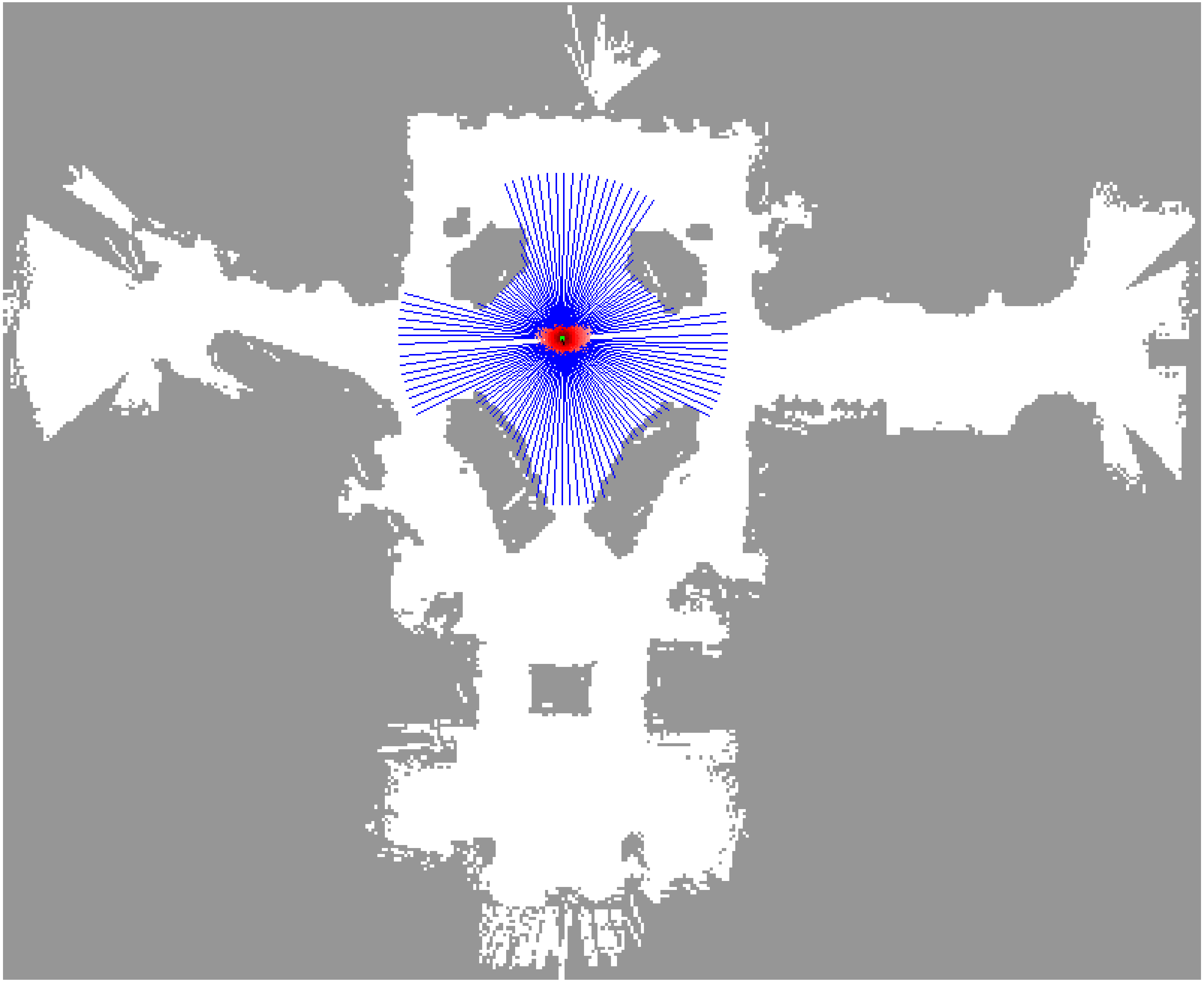




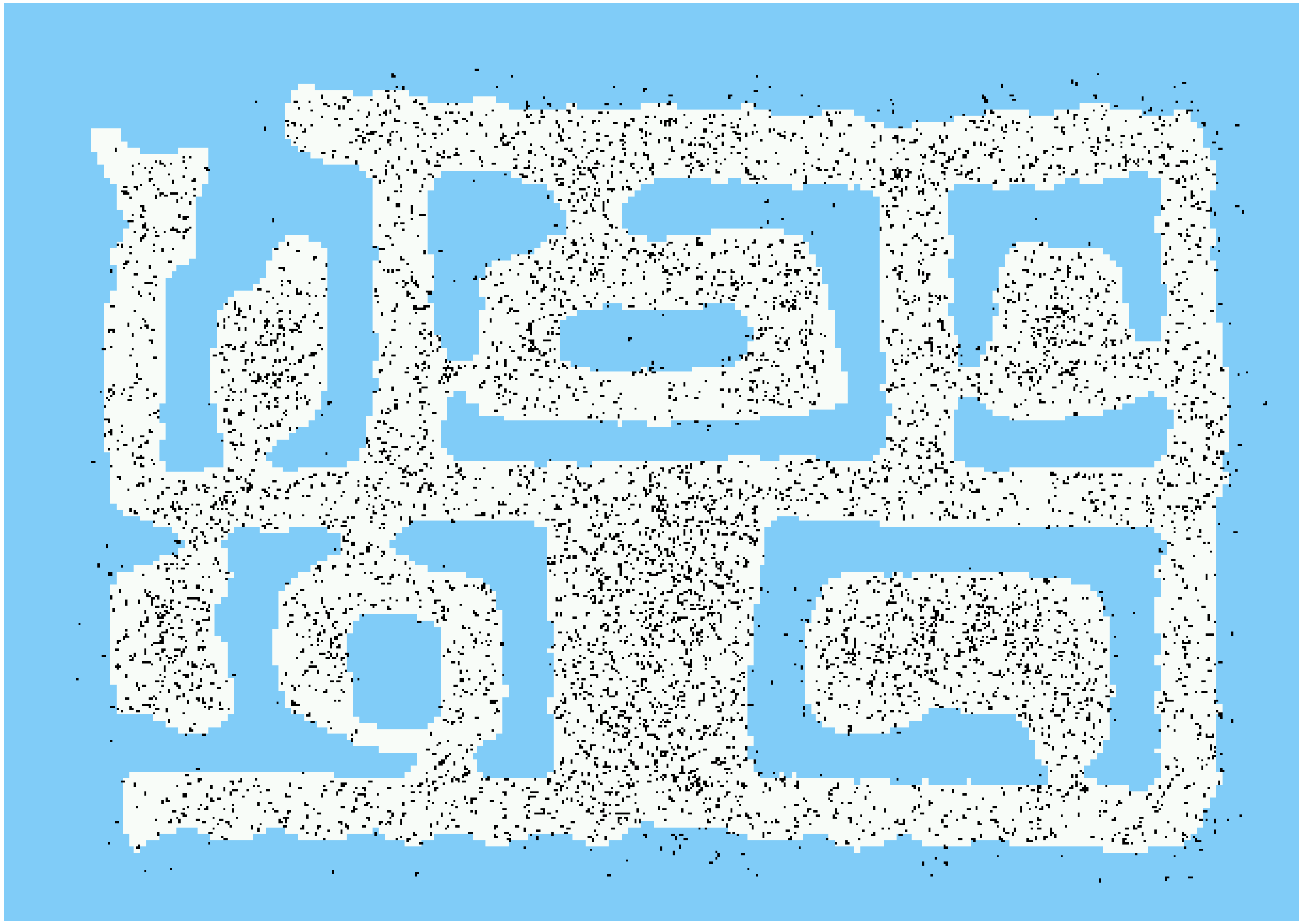




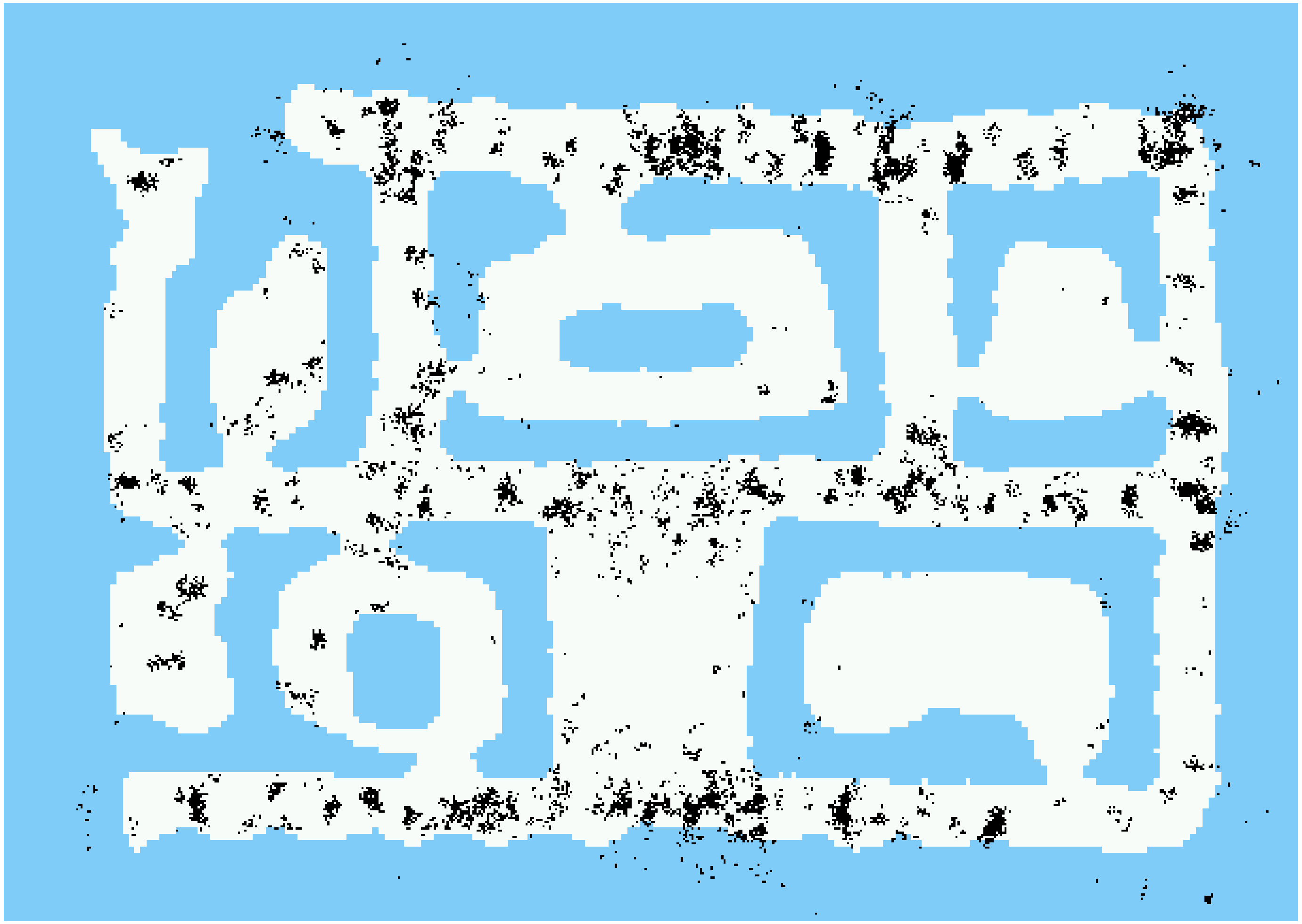




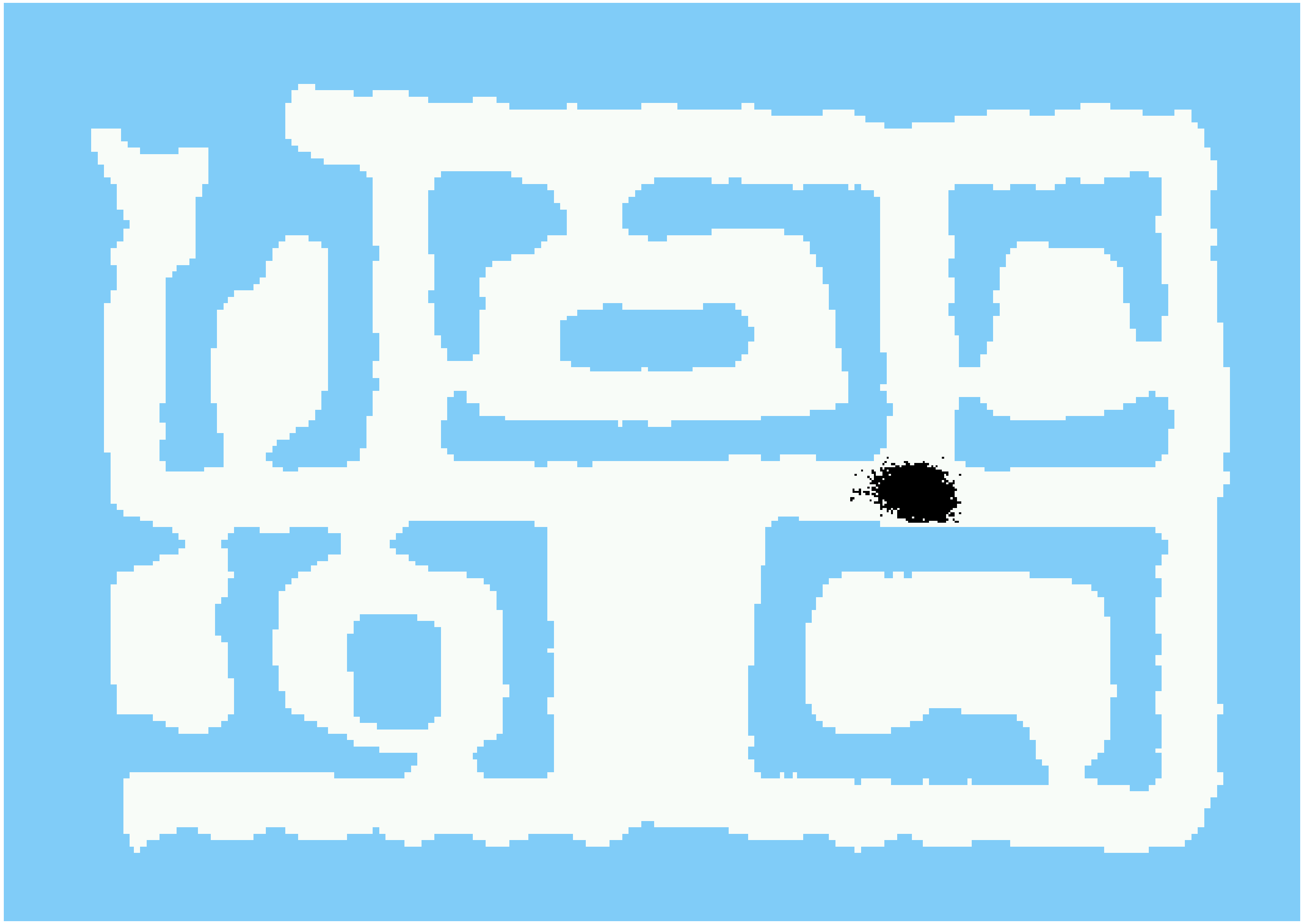
Initial Distribution



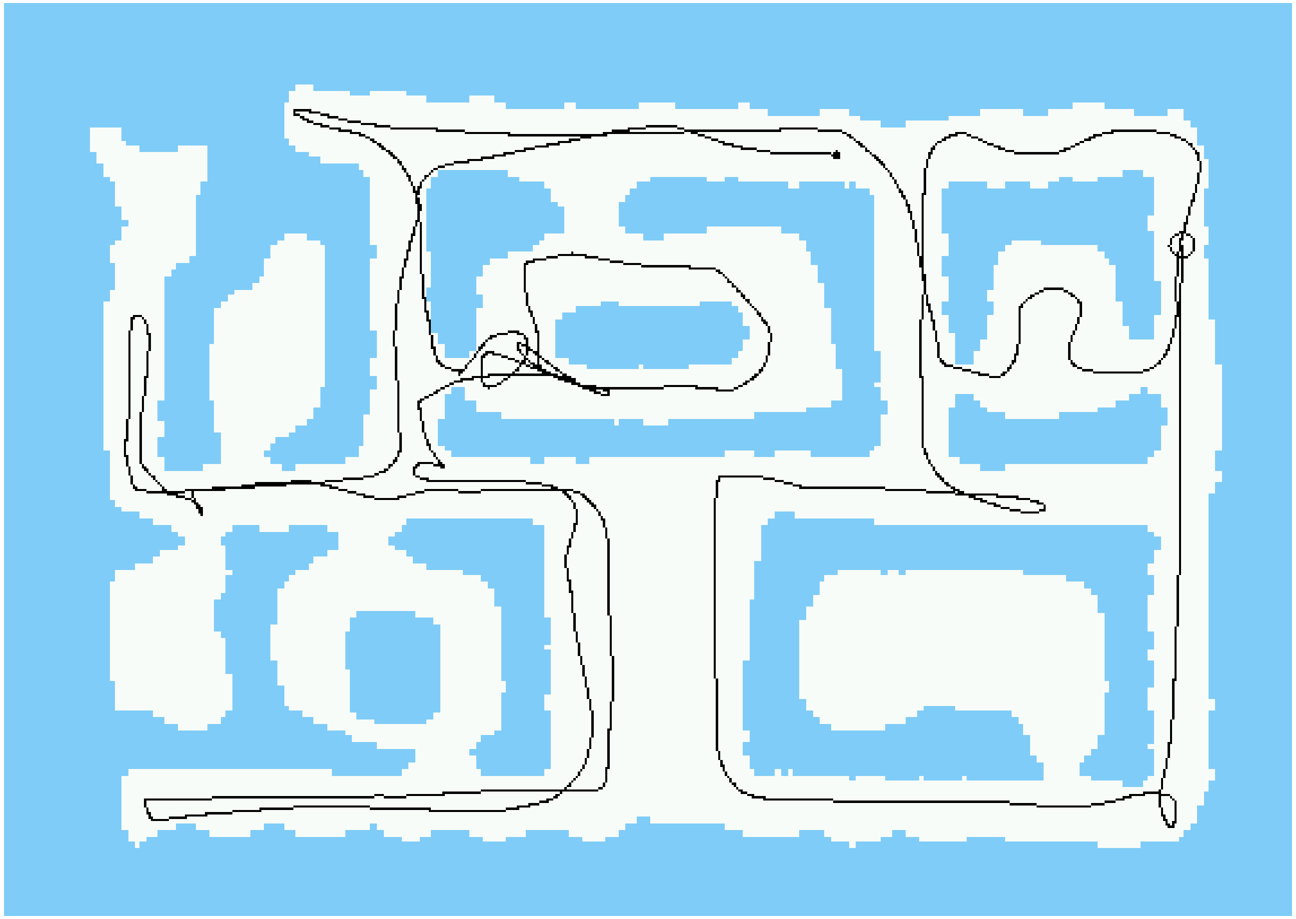
After Incorporating Ten Ultrasound Scans



After Incorporating 65 Ultrasound Scans



Estimated Path



Summary – Particle Filters

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples
- They can model non-Gaussian distributions
- Proposal to draw new samples
- Weight to account for the differences between the proposal and the target
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter

Summary – PF Localization

- In the context of localization, the particles are propagated according to the motion model.
- They are then weighted according to the likelihood of the observations.
- In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.

Introduction to Mobile Robotics

SLAM: Simultaneous Localization and Mapping

Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

Slides by Kai Arras and Wolfram Burgard

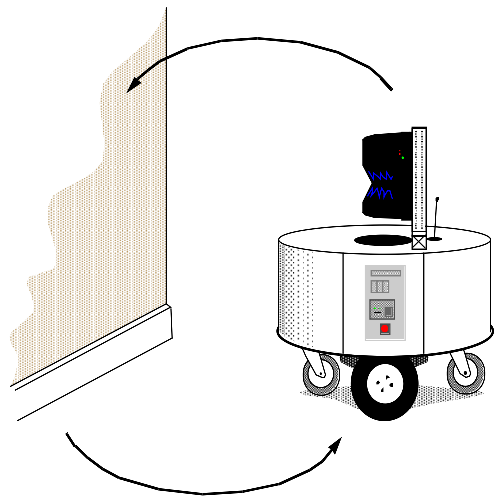
Last update: June 2010

The SLAM Problem

SLAM is the process by which a robot **builds a map** of the environment and, at the same time, uses this map to **compute its location**

- **Localization:** inferring location given a map
- **Mapping:** inferring a map given a location
- **SLAM:** learning a map and locating the robot simultaneously

The SLAM Problem



- SLAM is a **chicken-or-egg problem**:
  - → A map is needed for localizing a robot
  - → A pose estimate is needed to build a map
- Thus, SLAM is (regarded as) a **hard problem** in robotics

The SLAM Problem

- SLAM is considered **one of the most fundamental problems** for robots to become truly autonomous
- A variety of different approaches to address the SLAM problem have been presented
- **Probabilistic methods** rule
- History of SLAM dates back to the **mid-eighties** (stone-age of mobile robotics)

The SLAM Problem

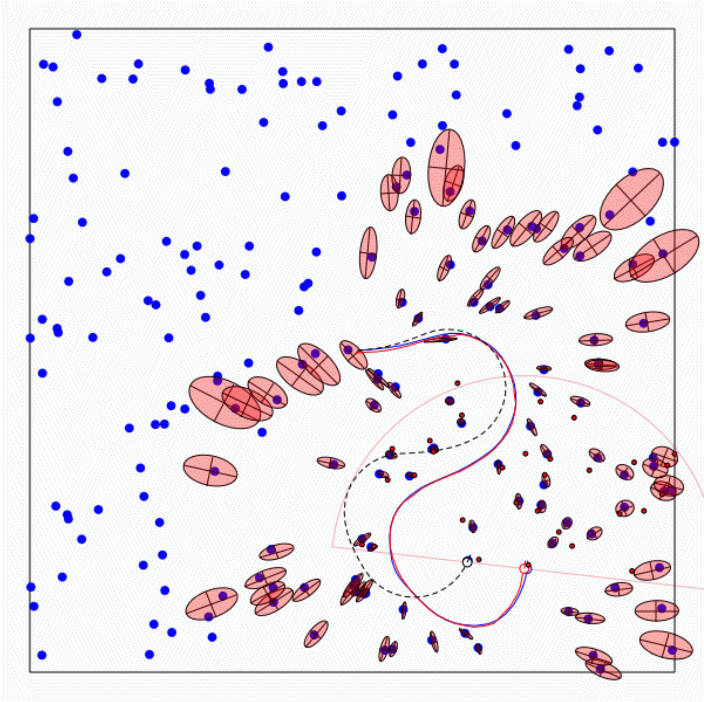
Given:

- The robot's controls

  $$\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\}$$

- Relative observations

  $$\mathbf{Z}_{0:k} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\}$$



Wanted:

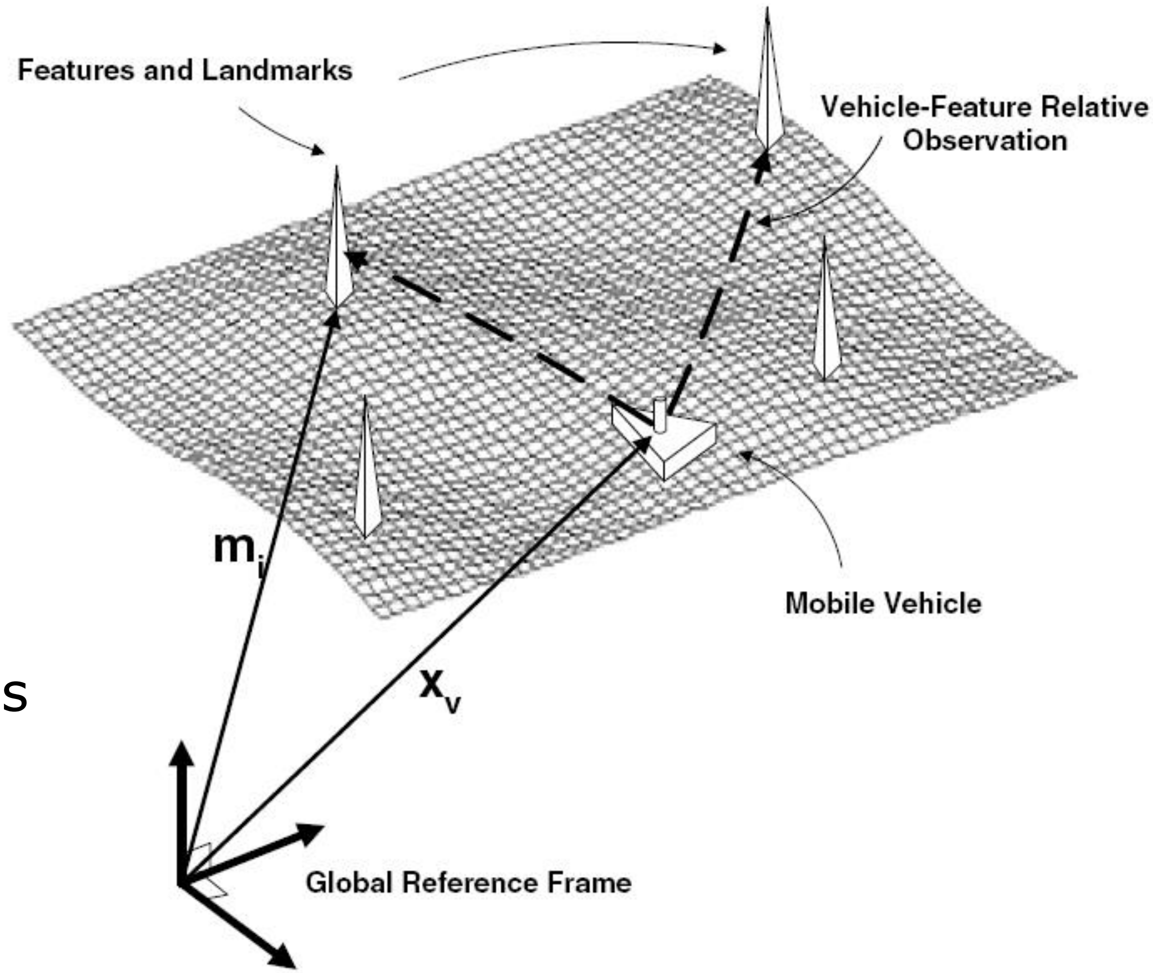
- Map of features

  $$\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$$

- Path of the robot

  $$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\}$$

The SLAM Problem



- **Absolute** robot pose
- **Absolute** landmark positions
- But only **relative** measurements of landmarks

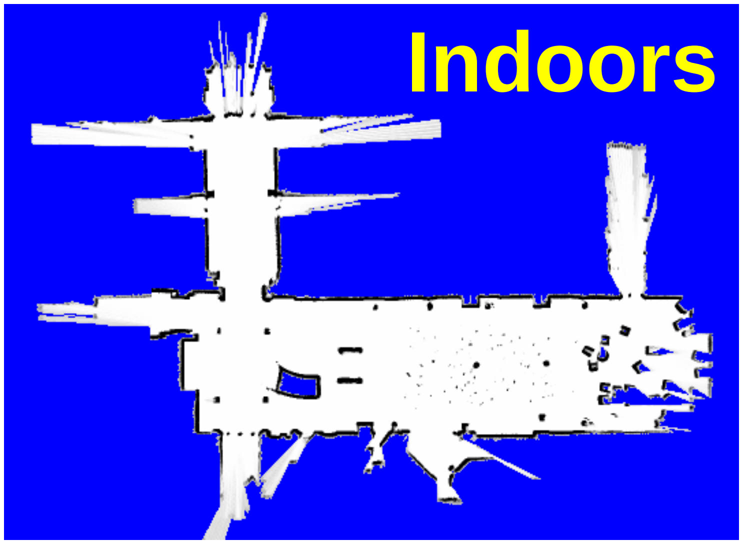
SLAM Applications

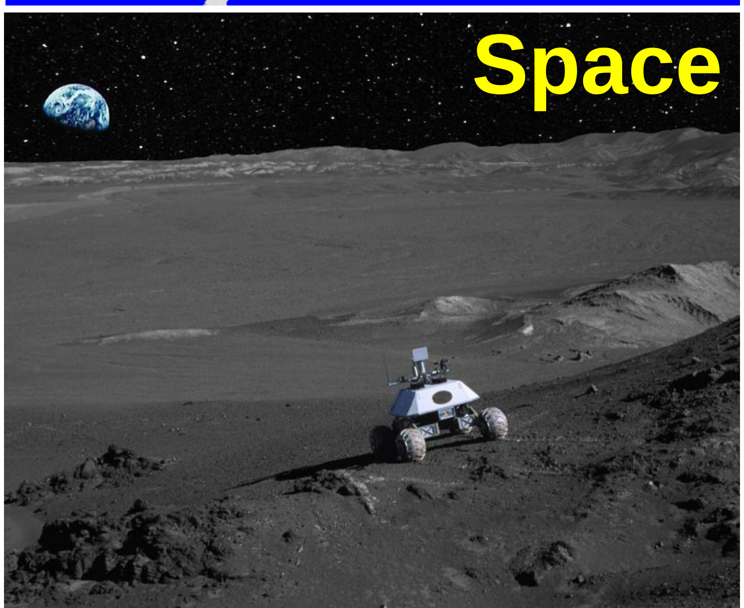
SLAM is central to a range of indoor, outdoor, in-air and underwater **applications** for both manned and autonomous vehicles.

Examples:

- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of abandoned mines
- Space: terrain mapping for localization

SLAM Applications

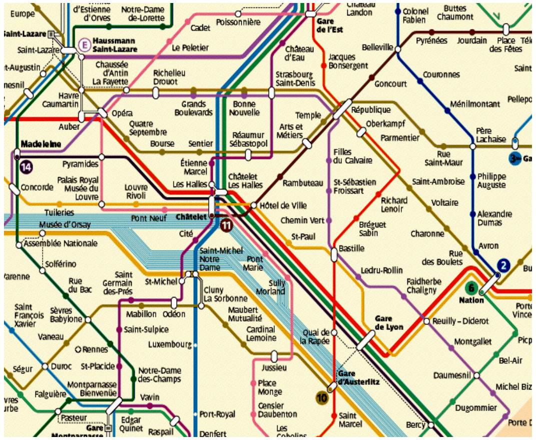


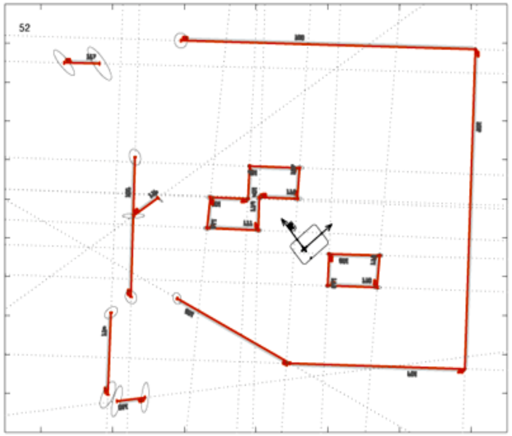


Map Representations

Examples:

Subway map, city map, landmark-based map

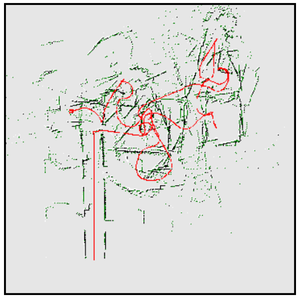
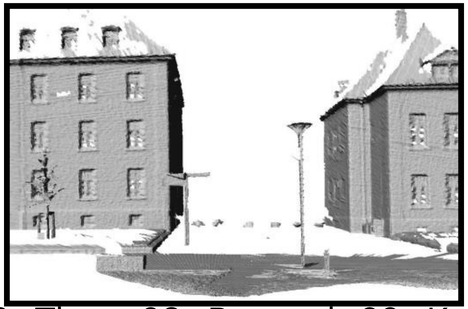
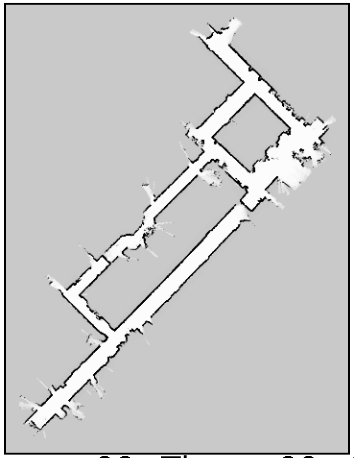




Maps are **topological** and/or **metric models** of the environment

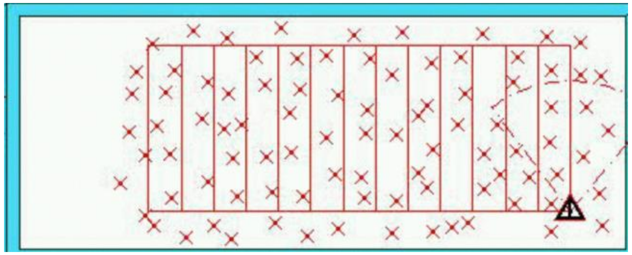
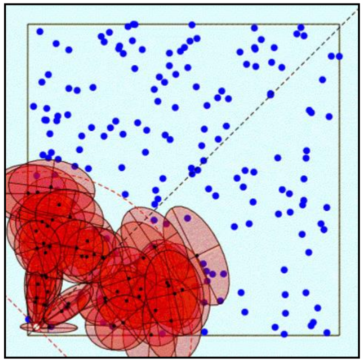
Map Representations

- Grid maps or scans, 2d, 3d

[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

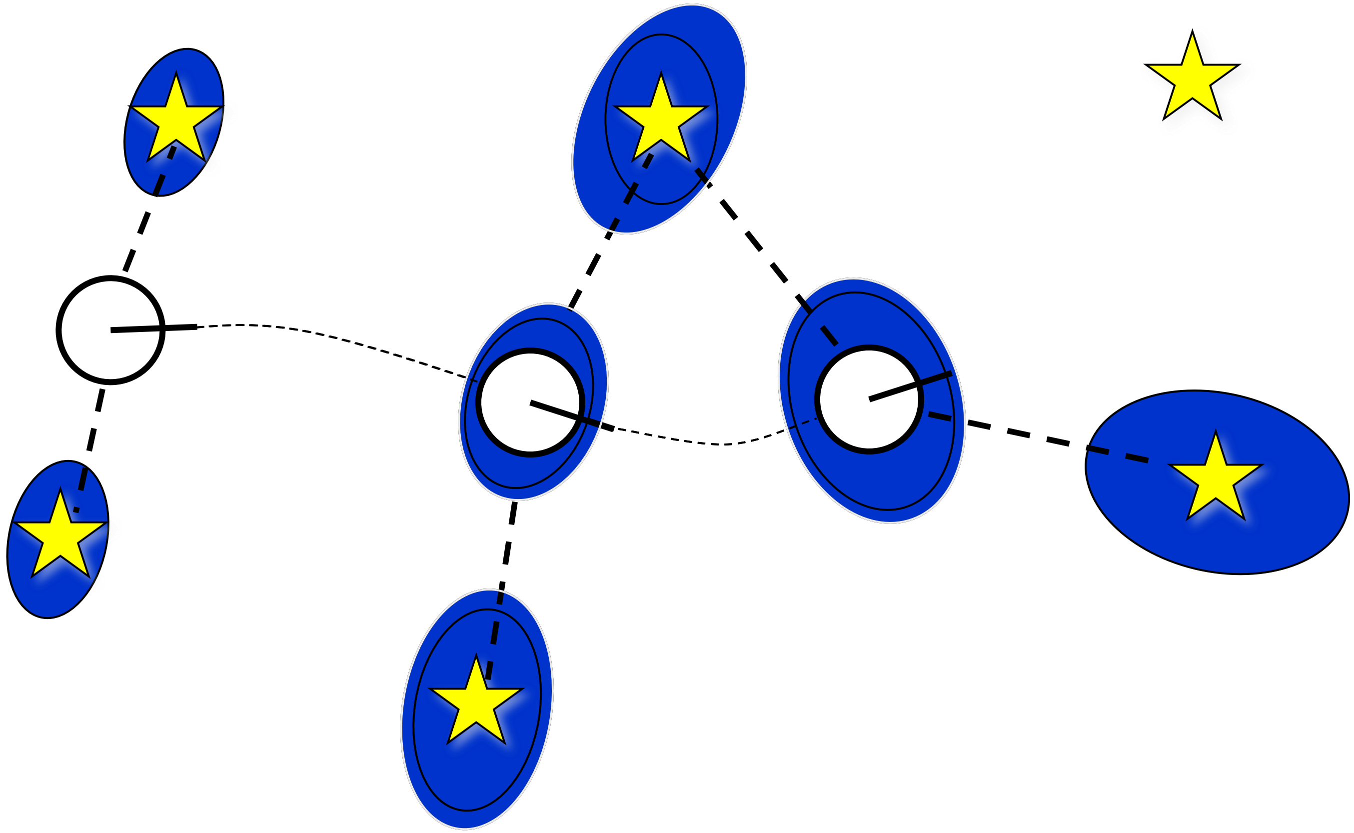
- Landmark-based

[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

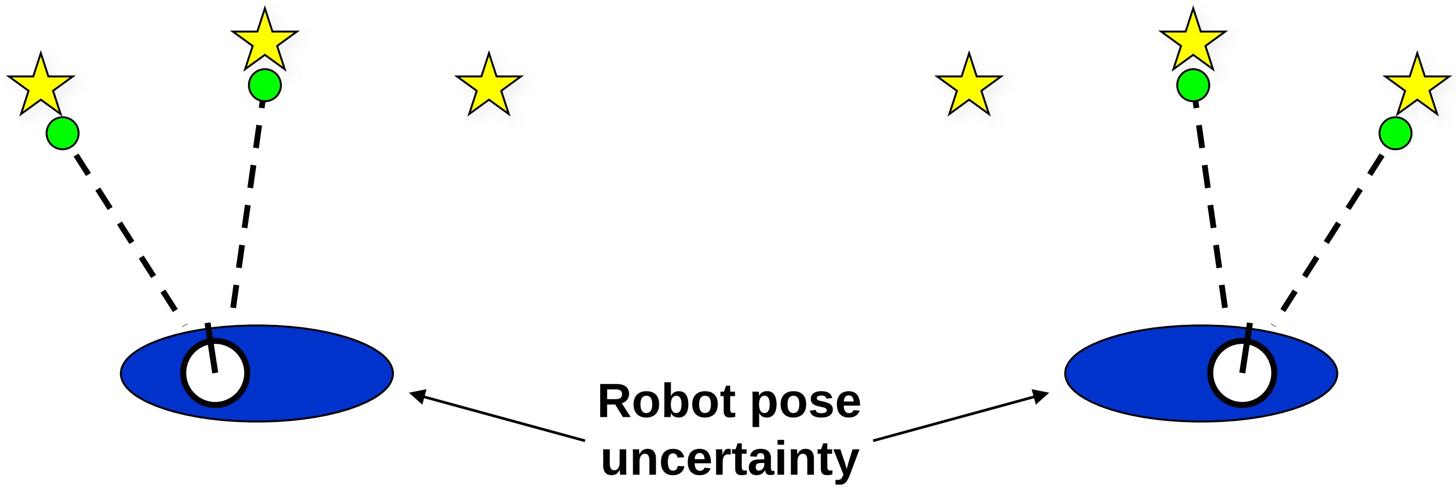
Why is SLAM a hard problem?

1. Robot path and map are both **unknown**



2. Errors in map and pose estimates correlated

Why is SLAM a hard problem?



- In the real world, the **mapping between observations and landmarks is unknown** (origin uncertainty of measurements)
- **Data Association**: picking **wrong** data associations can have **catastrophic** consequences (divergence)

SLAM:
Simultaneous Localization And Mapping

- Full SLAM:

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

Estimates entire path and map!

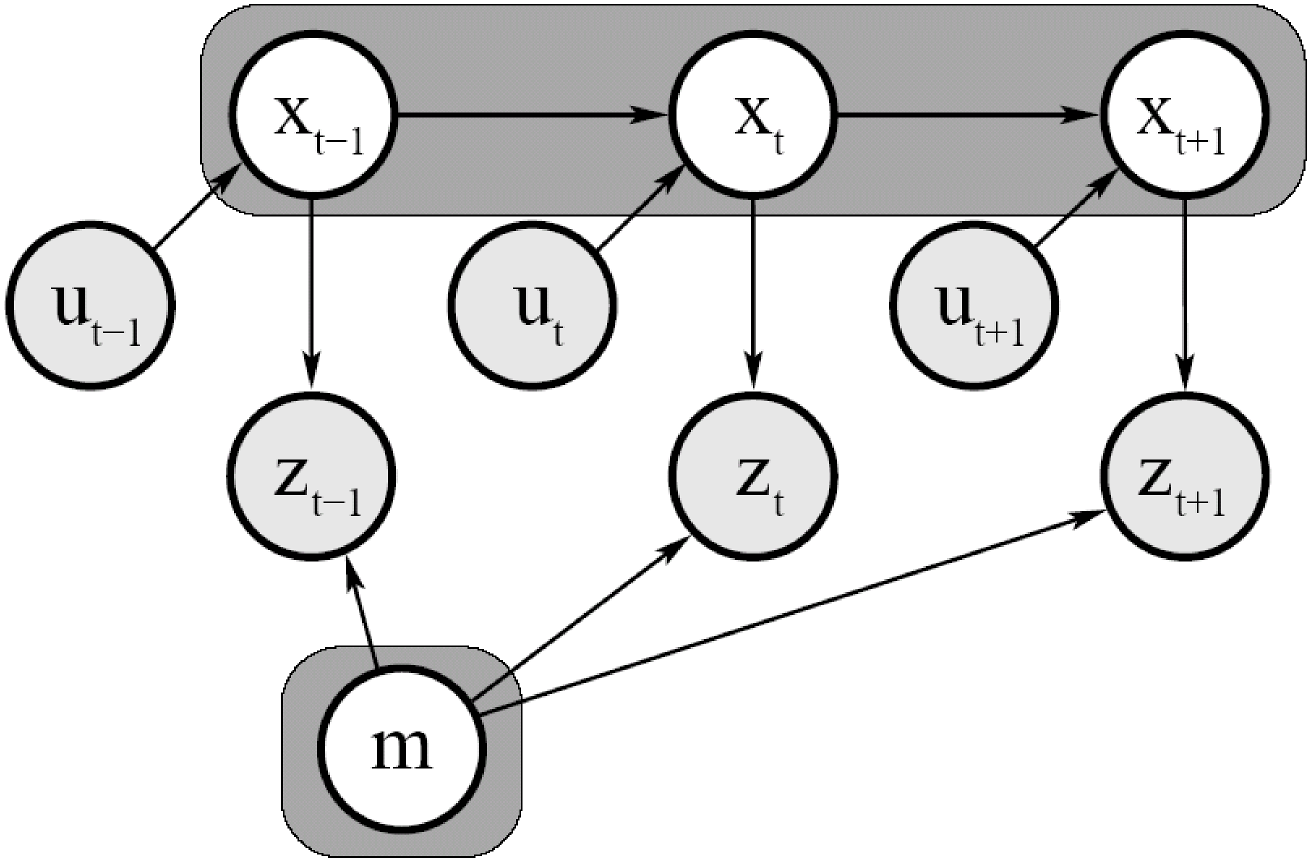
- Online SLAM:

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

Integrations (marginalization) typically done recursively, one at a time

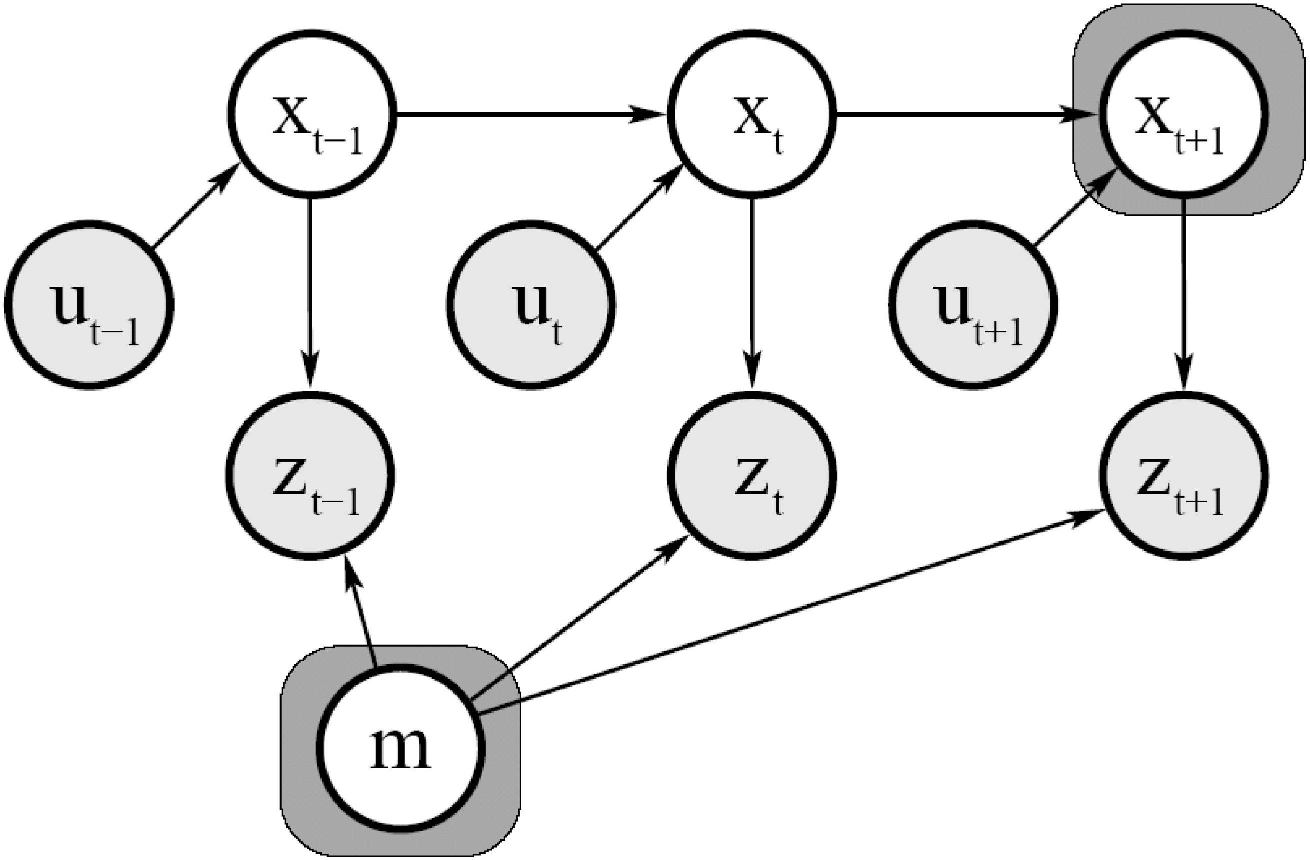
Estimates most recent pose and map!

Graphical Model of Full SLAM



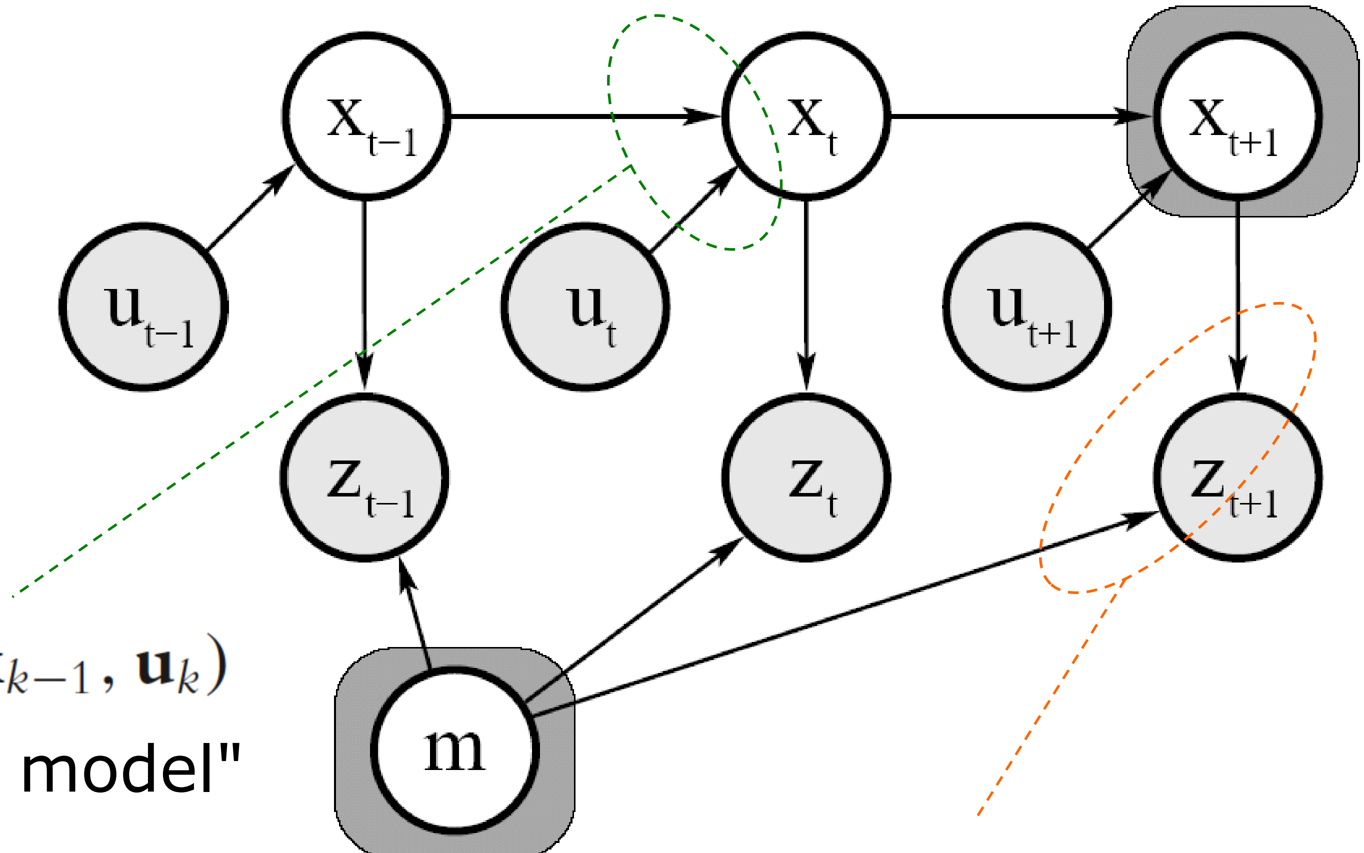
$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

Graphical Model of Online SLAM



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

Graphical Model: Models



$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k)$$

"Motion model"

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{m})$$

"Observation model"

EKF SLAM: State representation

- **Localization**

  3x1 pose vector
  3x3 cov. matrix

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \qquad C_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{bmatrix}$$

- **SLAM**

  Landmarks are **simply added** to the state.
  Growing state vector and covariance matrix!

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

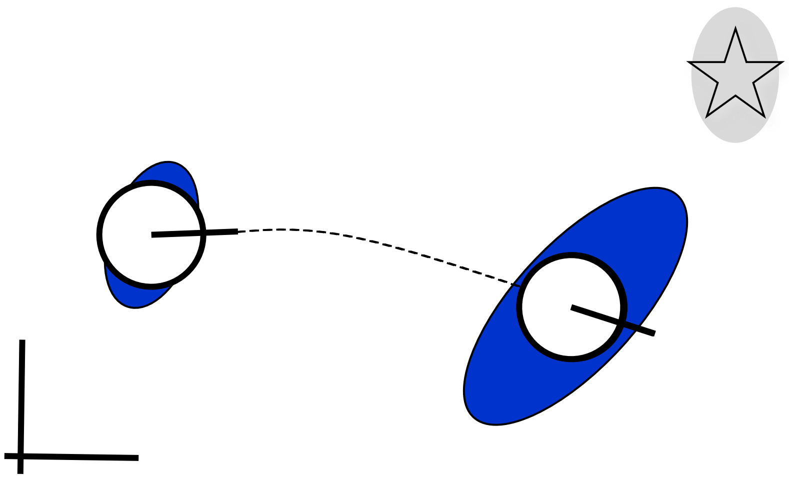
EKF SLAM: Building the Map

Filter Cycle, Overview:

1. State prediction (odometry)
2. Measurement prediction
3. Observation
4. Data Association
5. Update
6. Integration of new landmarks

EKF SLAM: Building the Map

- State Prediction



Odometry:

$$\hat{\mathbf{x}}_R = f(\mathbf{x}_R, \mathbf{u})$$

$$\hat{C}_R = F_x \, C_R \, F_x^T + F_u \, U \, F_u^T$$

Robot-landmark cross-covariance prediction:

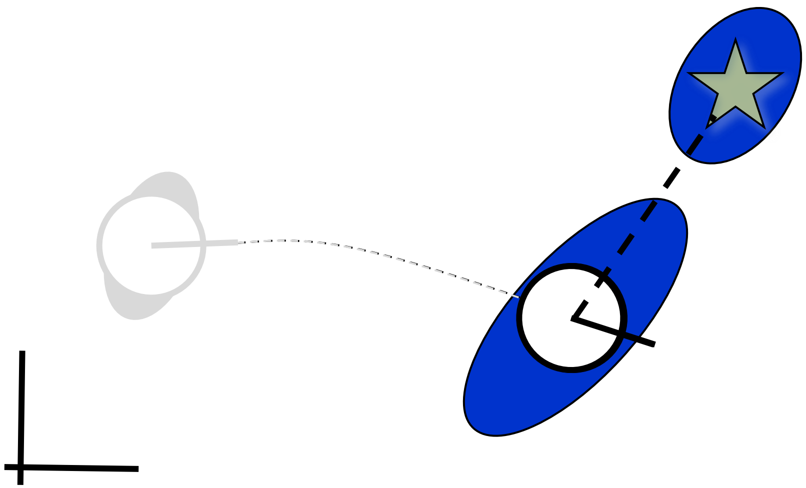
$$\hat{C}_{RM_i} = F_x \, C_{RM_i}$$

(skipping time index k)

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

EKF SLAM: Building the Map

- Measurement Prediction



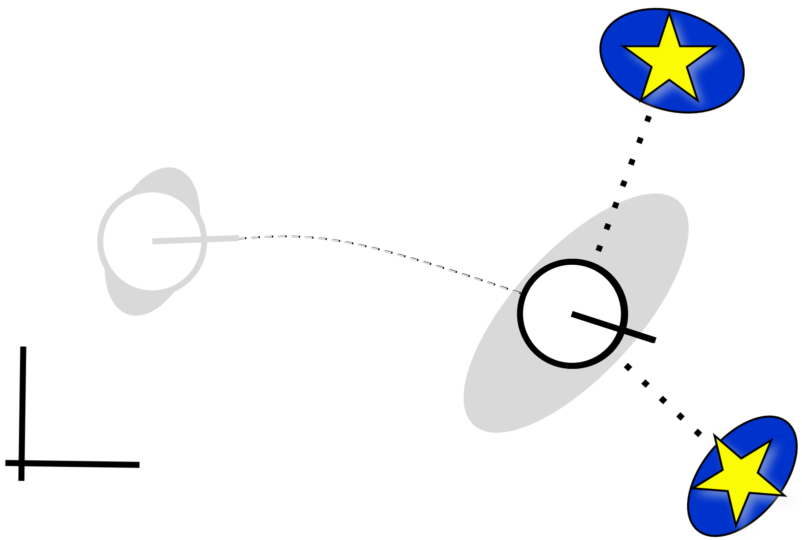
Global-to-local frame transform h

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k)$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

EKF SLAM: Building the Map

- Observation



(x,y)-point landmarks

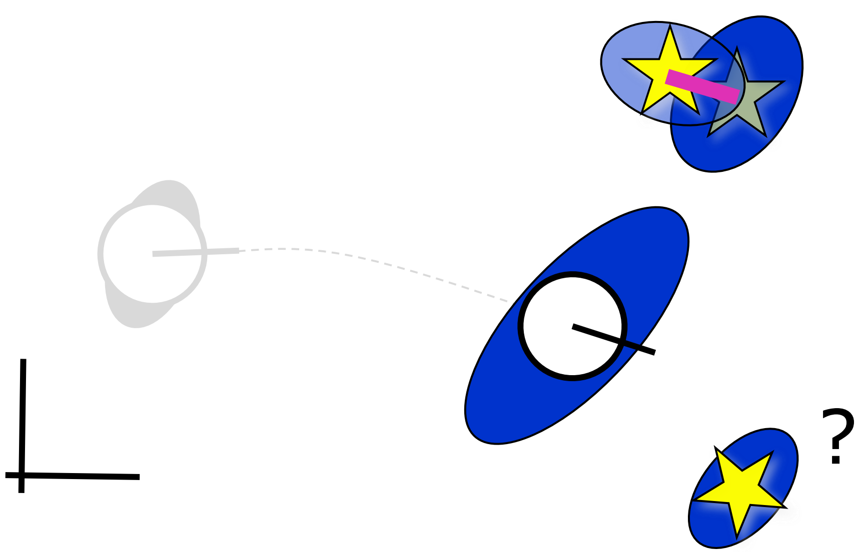
$$\mathbf{z}_k = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}$$

$$R_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

EKF SLAM: Building the Map

- Data Association



Associates predicted measurements $\hat{\mathbf{z}}_k^i$ with observation $\mathbf{z}_k^j$

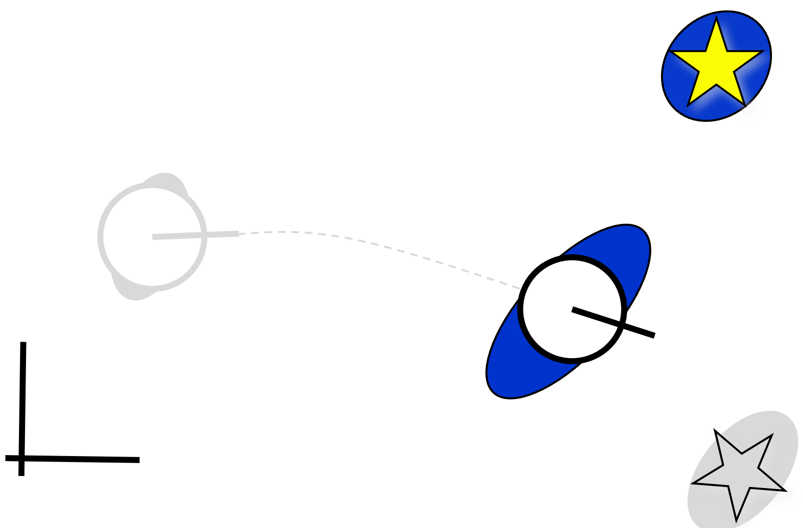
$$\begin{aligned} \nu_k^{ij} &= \mathbf{z}_k^j - \hat{\mathbf{z}}_k^i \\ S_k^{ij} &= R_k^j + H^i \, \hat{C}_k \, H^{i\,T} \end{aligned}$$

(Gating)

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

EKF SLAM: Building the Map

- Filter Update



The usual Kalman filter expressions

$$K_k = \hat{C}_k \, H^T S_k^{-1}$$

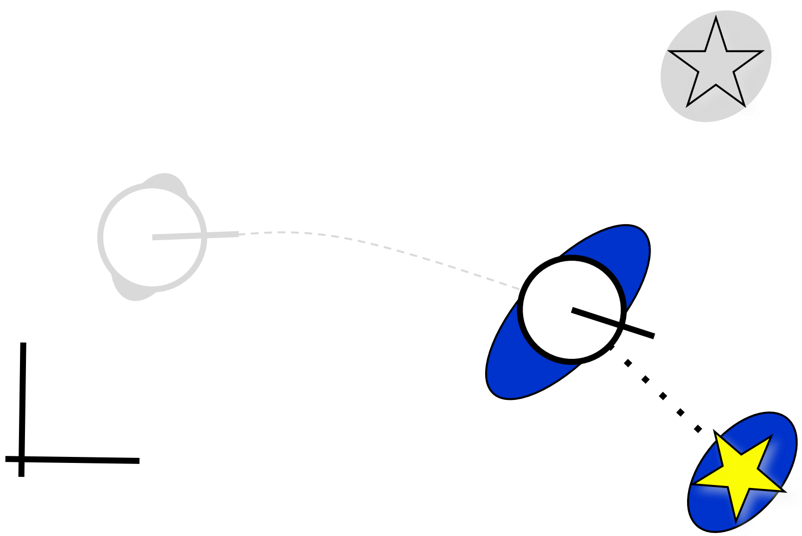
$$\mathbf{x}_k = \hat{\mathbf{x}}_k + K_k \, \nu_k$$

$$C_k = (I - K_k \, H) \, \hat{C}_k$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

EKF SLAM: Building the Map

- Integrating New Landmarks



State augmented by

$$\mathbf{m}_{n+1} = g(\mathbf{x}_R, \mathbf{z}_j)$$

$$C_{M_{n+1}} = G_R\, C_R\, G_R^T + G_z\, R_j\, G_z^T$$

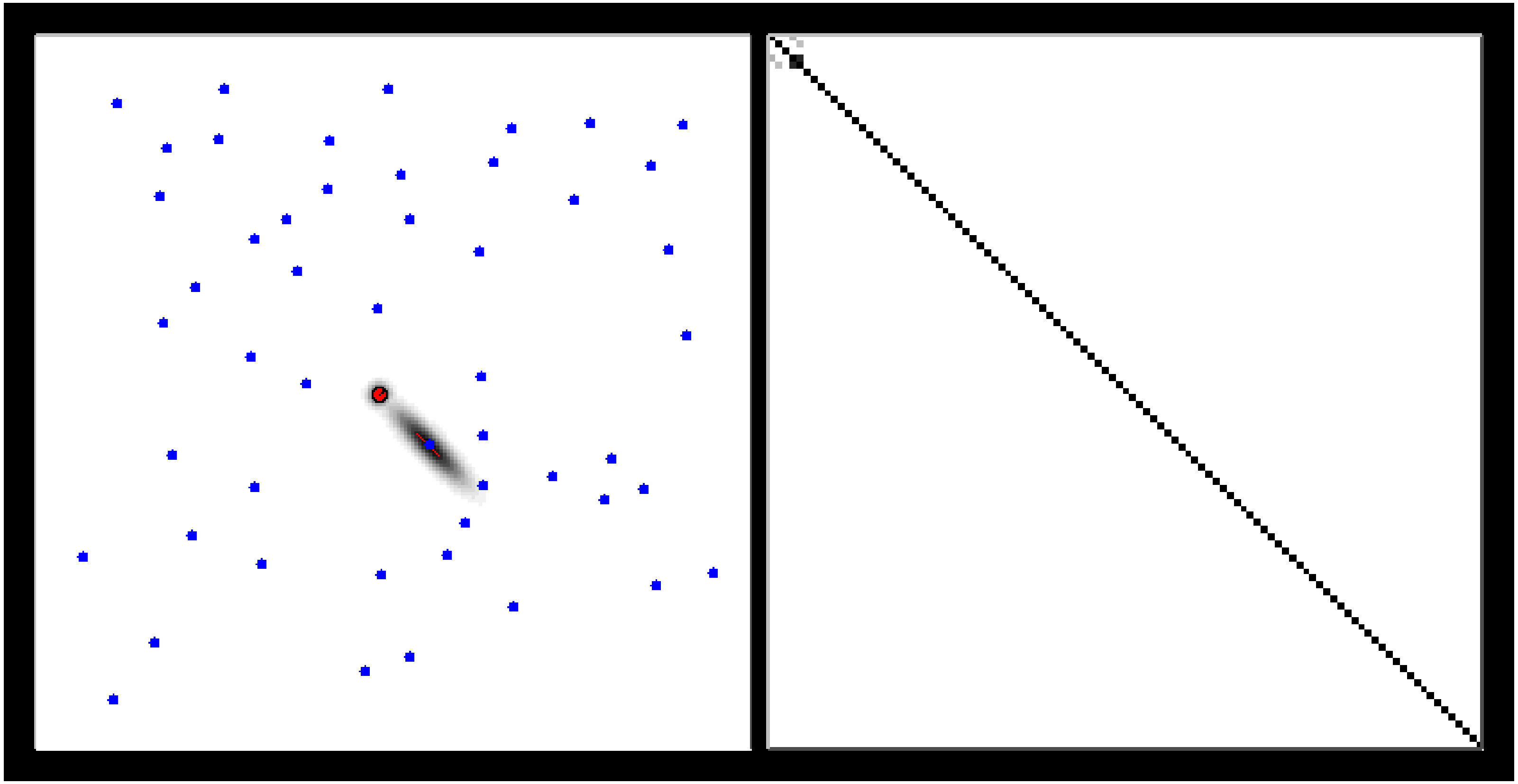
Cross-covariances:

$$C_{M_{n+1} M_i} = G_R\, C_{RM_i}$$

$$C_{M_{n+1} R} = G_R\, C_R$$

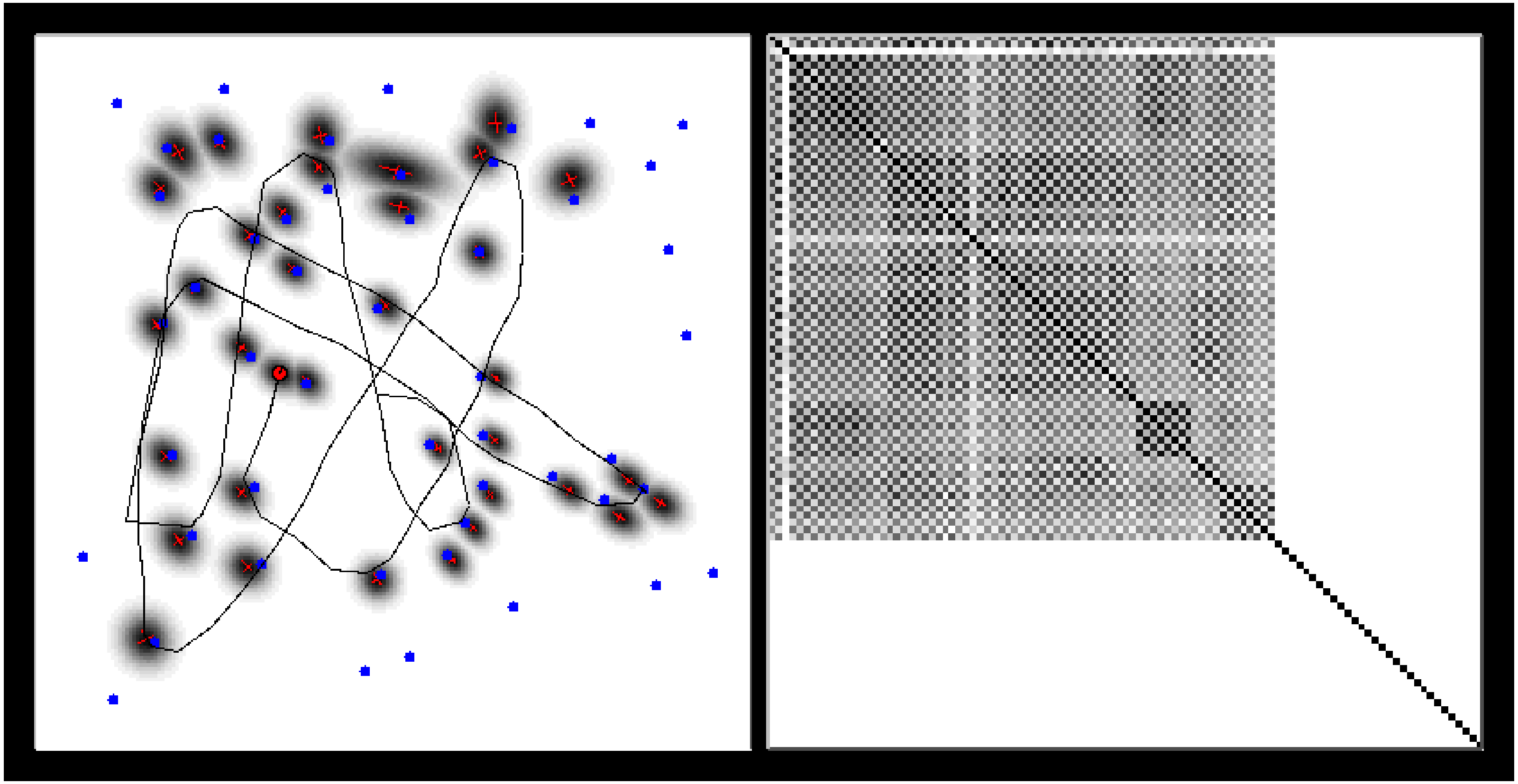
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \\ \mathbf{m}_{n+1} \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} & C_{RM_{n+1}} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} & C_{M_1M_{n+1}} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} & C_{M_2M_{n+1}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} & C_{M_nM_{n+1}} \\ C_{M_{n+1}R} & C_{M_{n+1}M_1} & C_{M_{n+1}M_2} & \cdots & C_{M_{n+1}M_n} & C_{M_{n+1}} \end{bmatrix}_k$$

EKF SLAM



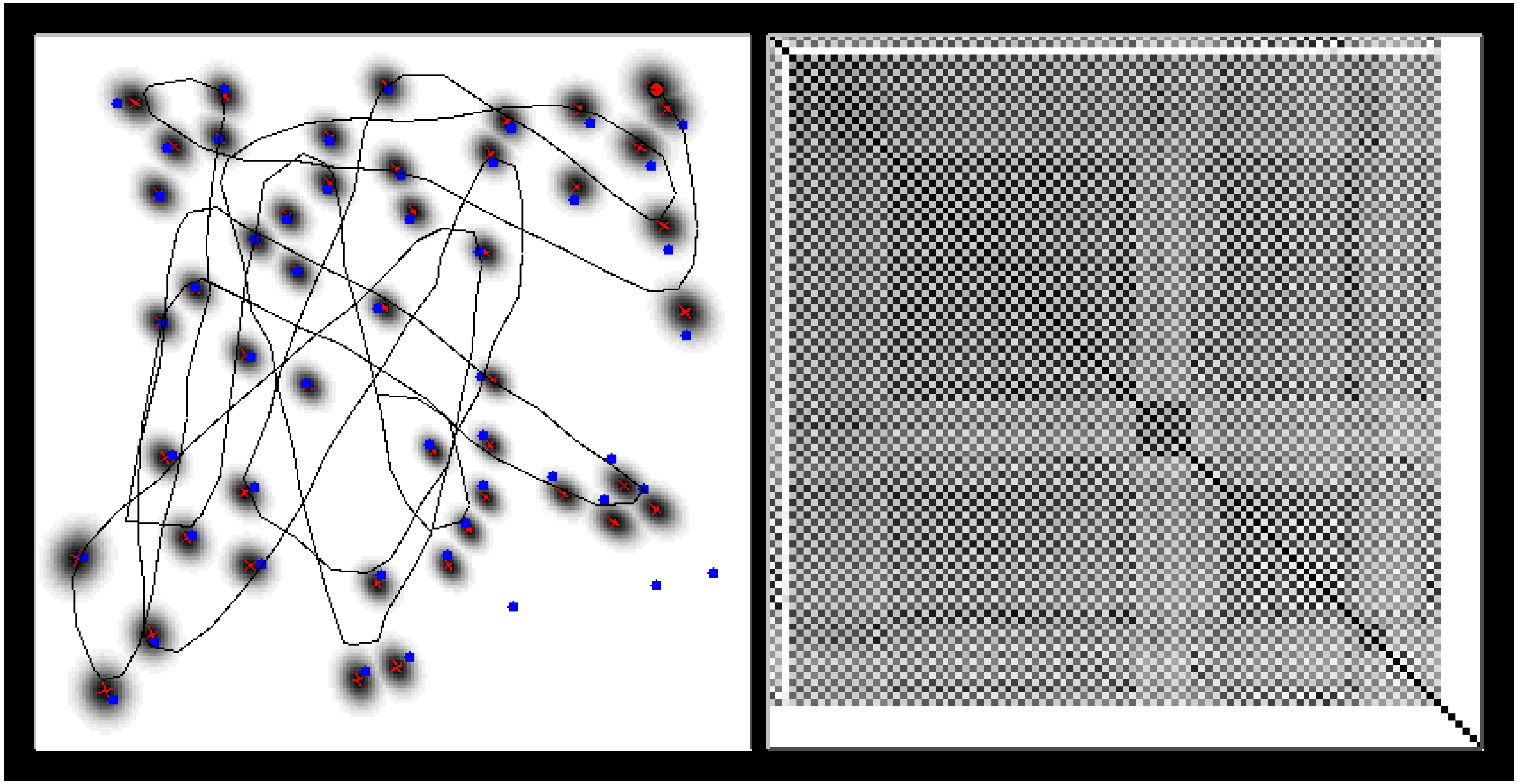
Map Correlation matrix

EKF SLAM



Map Correlation matrix

EKF SLAM



Map Correlation matrix

SLAM: Loop Closure

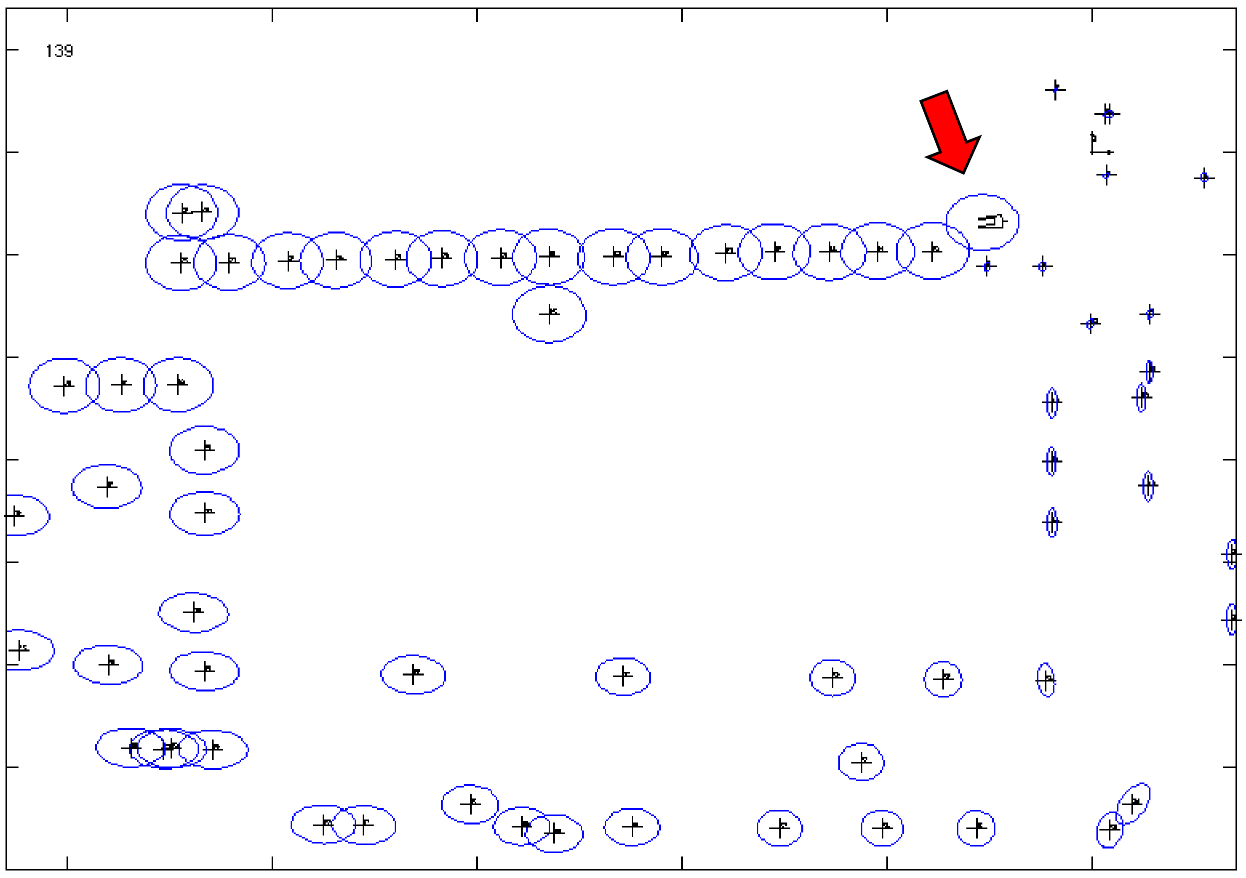
- Loop closure is the problem of **recognizing an already mapped area**, typically after a long exploration path (the robot "closes a loop")

- Structually identical to data association, but
  - high levels of ambiguity
  - possibly useless validation gates
  - environment symmetries

- Uncertainties **collapse** after a loop closure (whether the closure was correct or not)

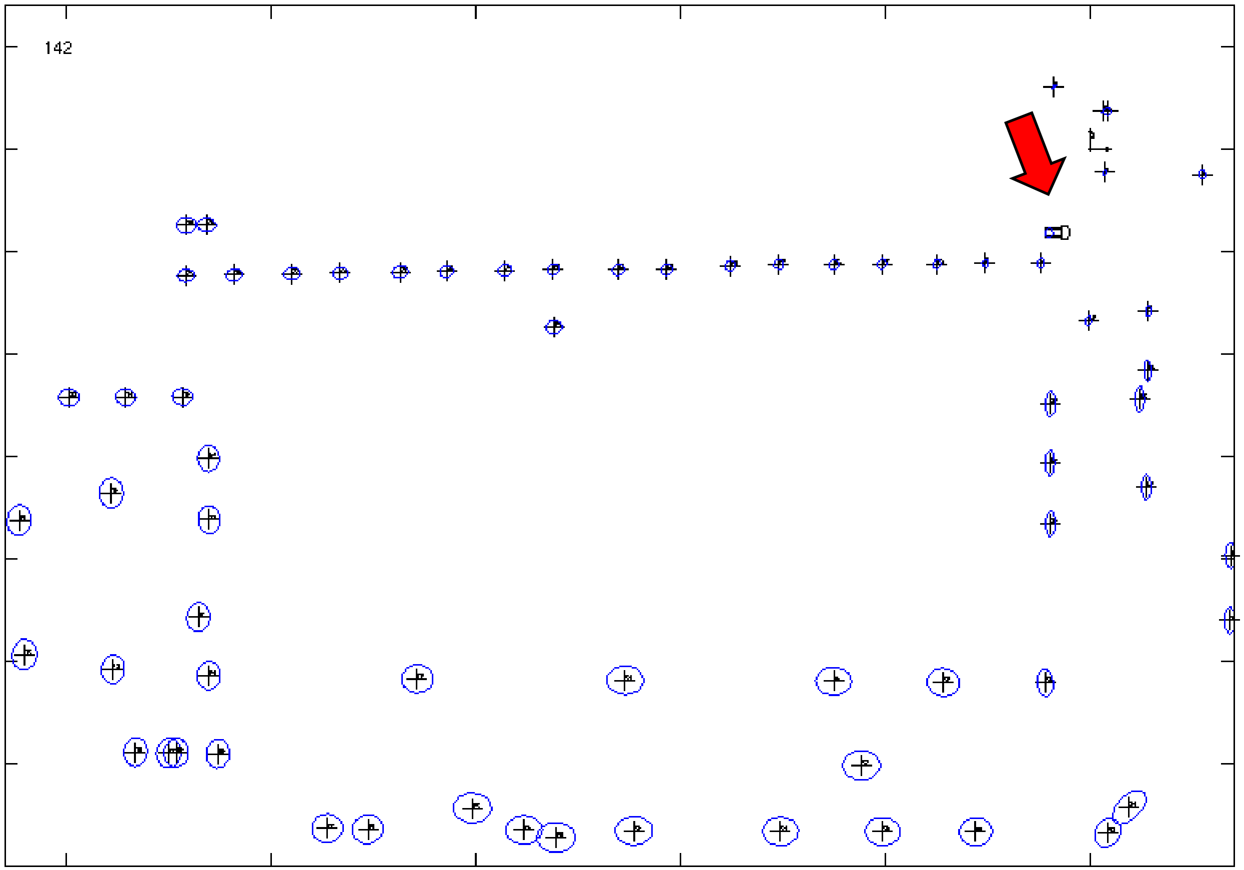
SLAM: Loop Closure

- Before loop closure



SLAM: Loop Closure

- After loop closure



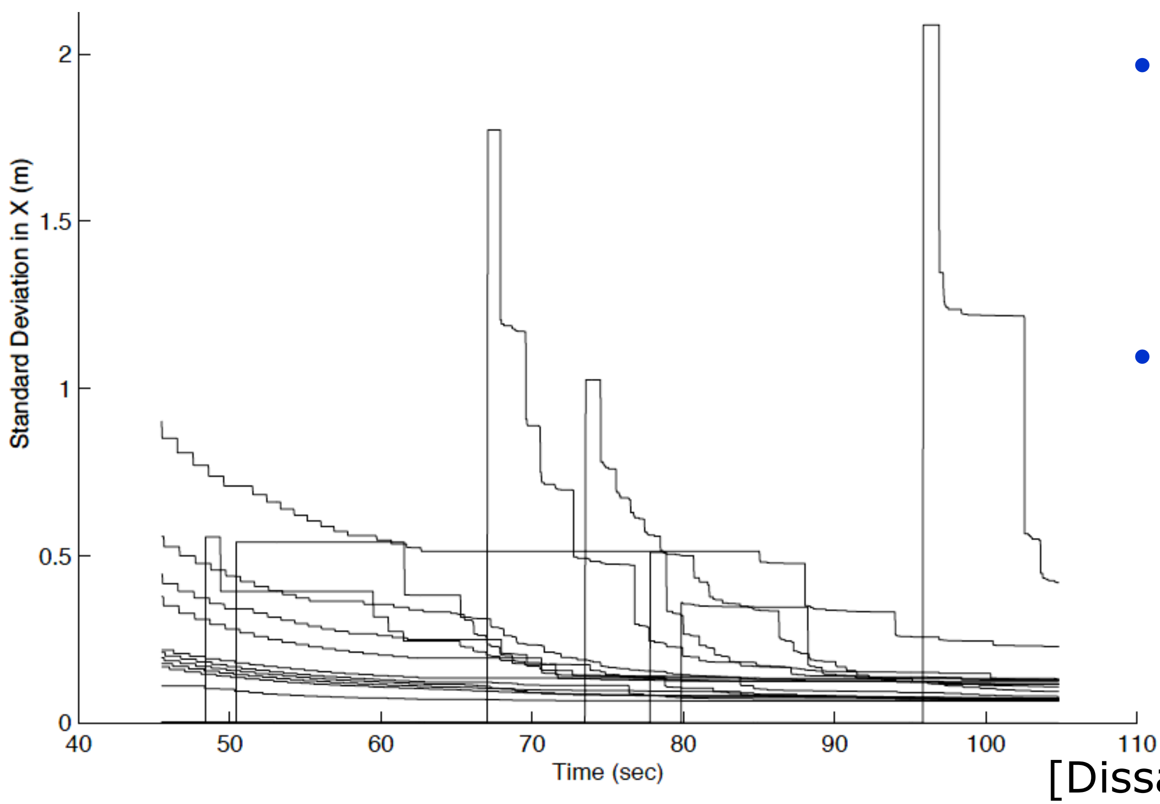
SLAM: Loop Closure

- By revisiting already mapped areas, uncertain-ties in robot and landmark estimates can be **reduced**
- This can be exploited to **"optimally" explore** an environment for the sake of better (e.g. more accurate) maps
- Exploration: the problem of ***where* to acquire new information** (e.g. depth-first vs. breadth first)

→ See separate chapter on exploration

KF-SLAM Properties (Linear Case)

- The **determinant** of any sub-matrix of the map covariance matrix **decreases monotonically** as successive observations are made

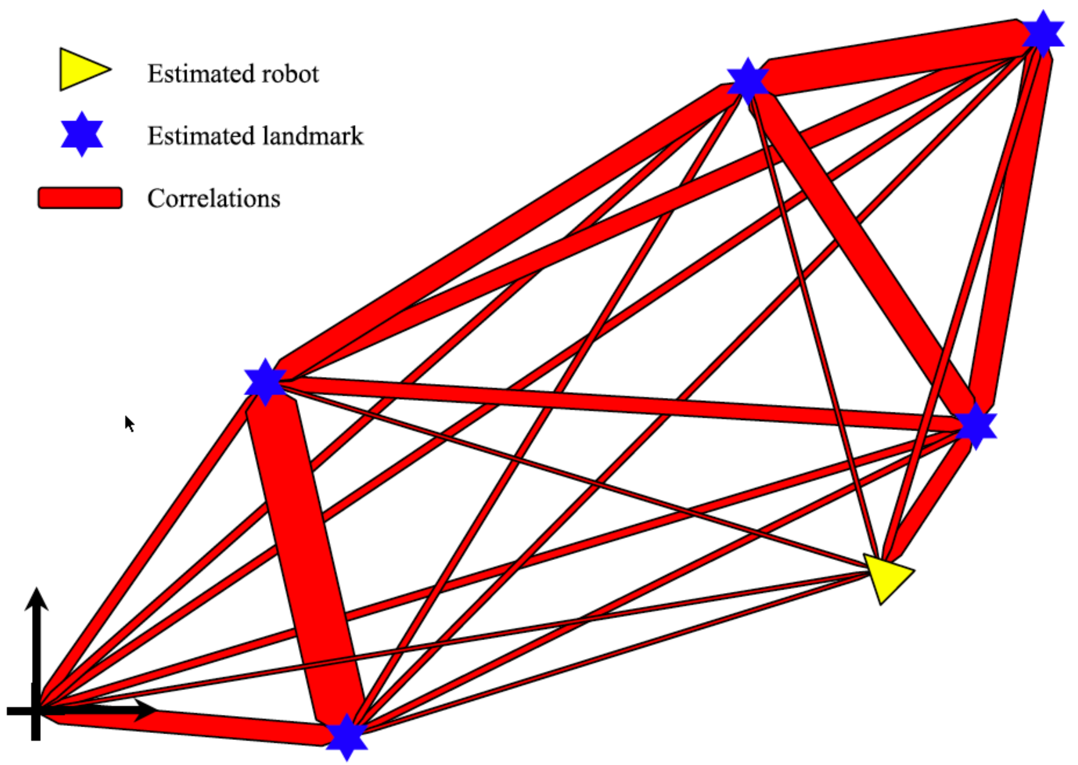


- When a new land-mark is initialized, its **uncertainty** is **maximal**
- Landmark uncer-tainty **decreases monotonically** with each new observation

[Dissanayake et al., 2001]

KF-SLAM Properties (Linear Case)

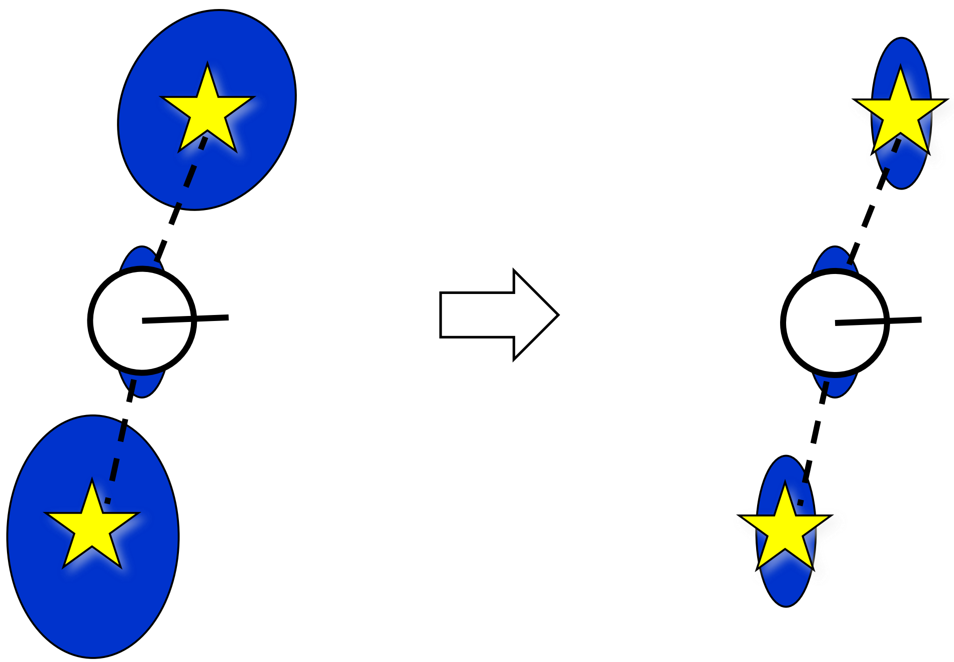
- In the limit, the landmark estimates become **fully correlated**



[Dissanayake et al., 2001]

KF-SLAM Properties (Linear Case)

- In the limit, the **covariance** associated with any single landmark location estimate is determined only by the **initial covariance in the vehicle location estimate**.

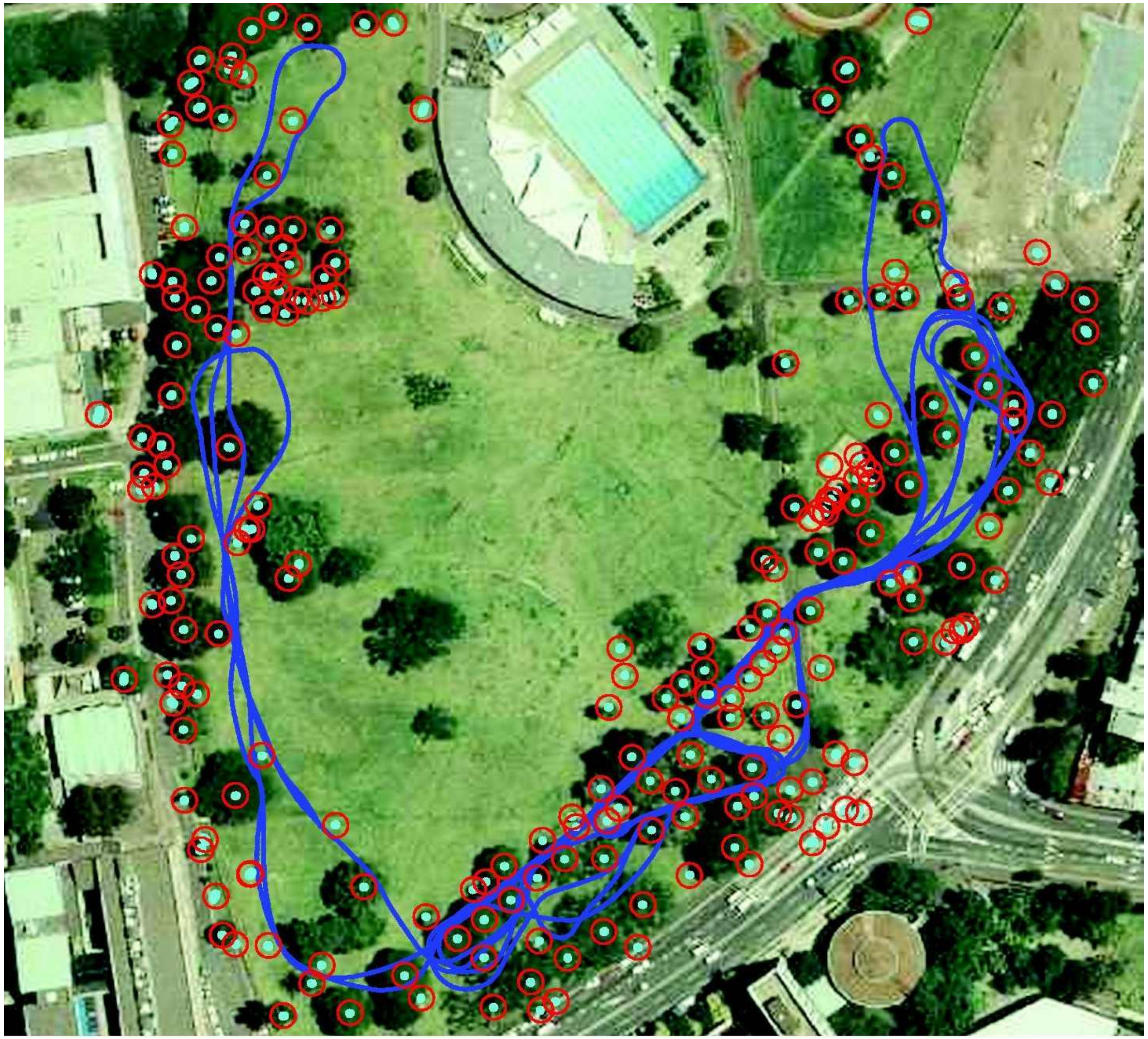


[Dissanayake et al., 2001]

EKF SLAM Example: Victoria Park

Syndey, Australia

Victoria Park: Landmarks



[courtesy by E. Nebot]

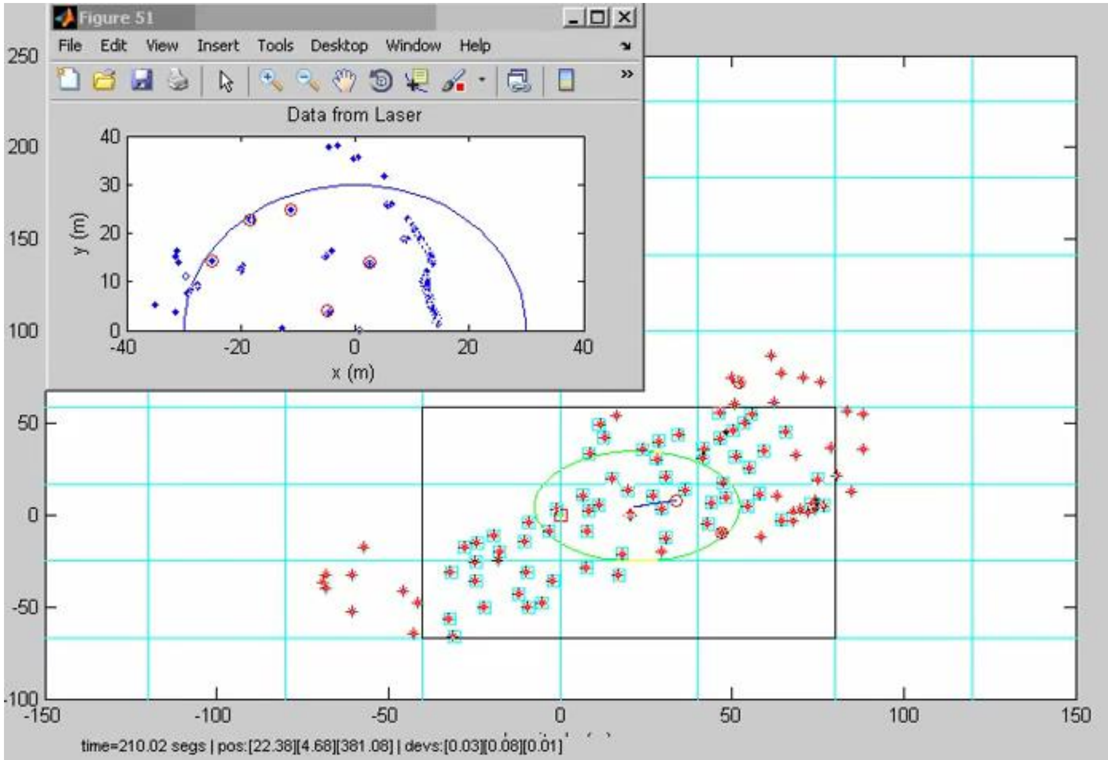
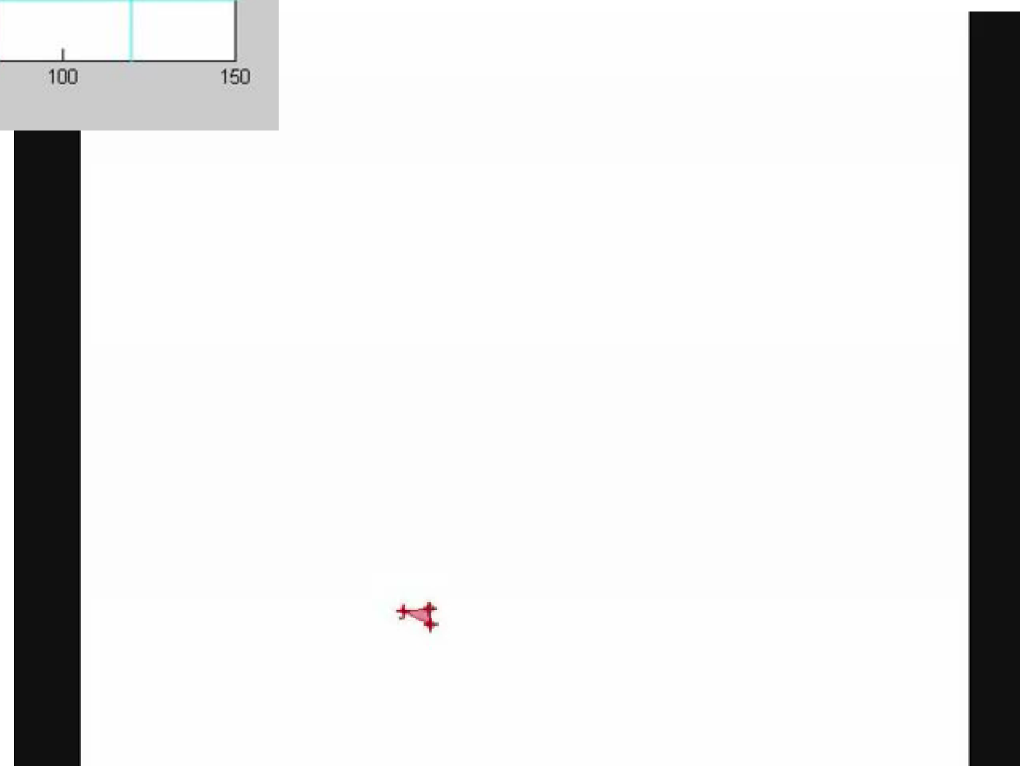
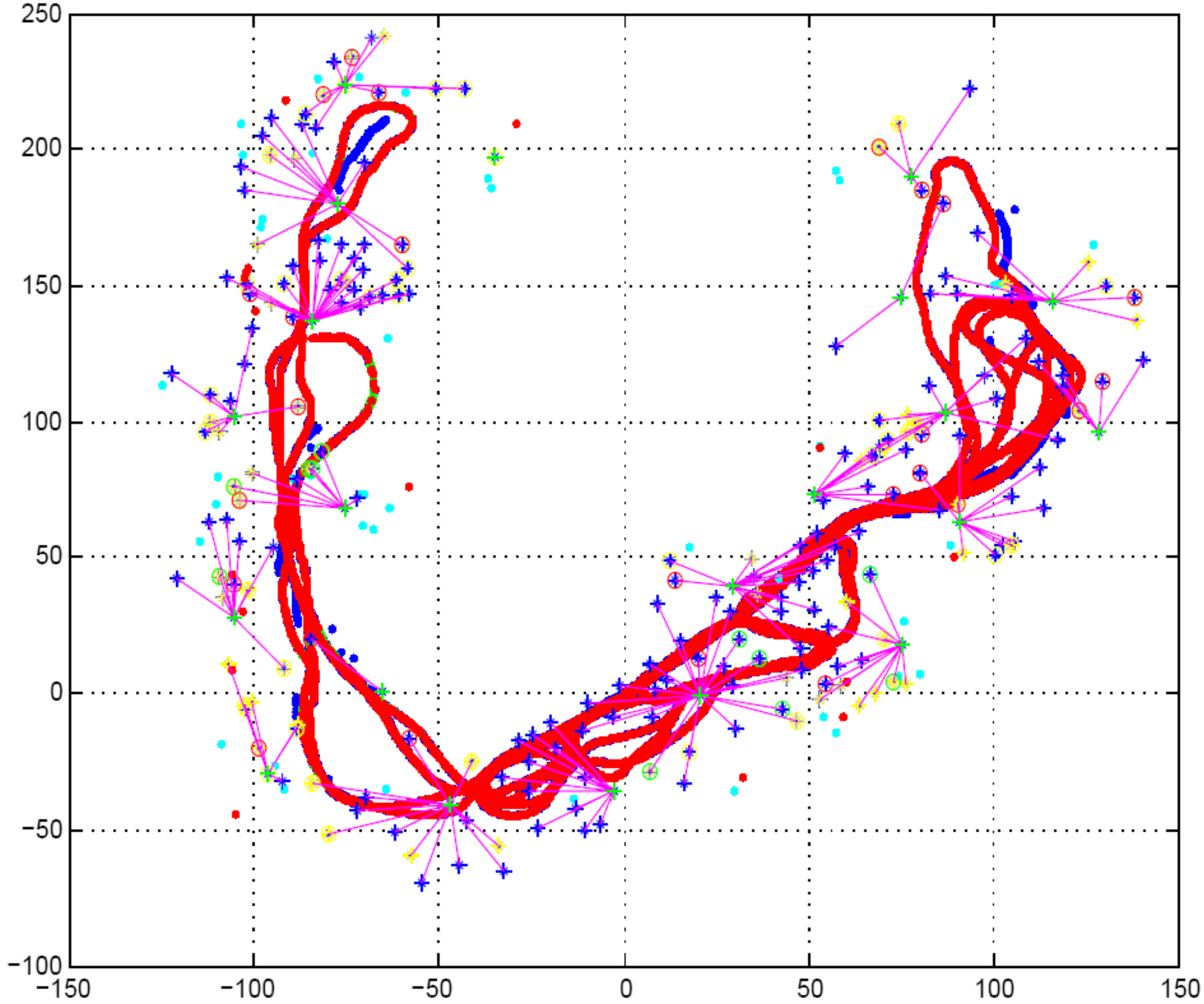


Figure 51
File Edit View Insert Tools Desktop Window Help
Data from Laser
y (m)
x (m)
time=210.02 segs | pos:[22.38][4.68][381.08] | devs:[0.03][0.08][0.01]

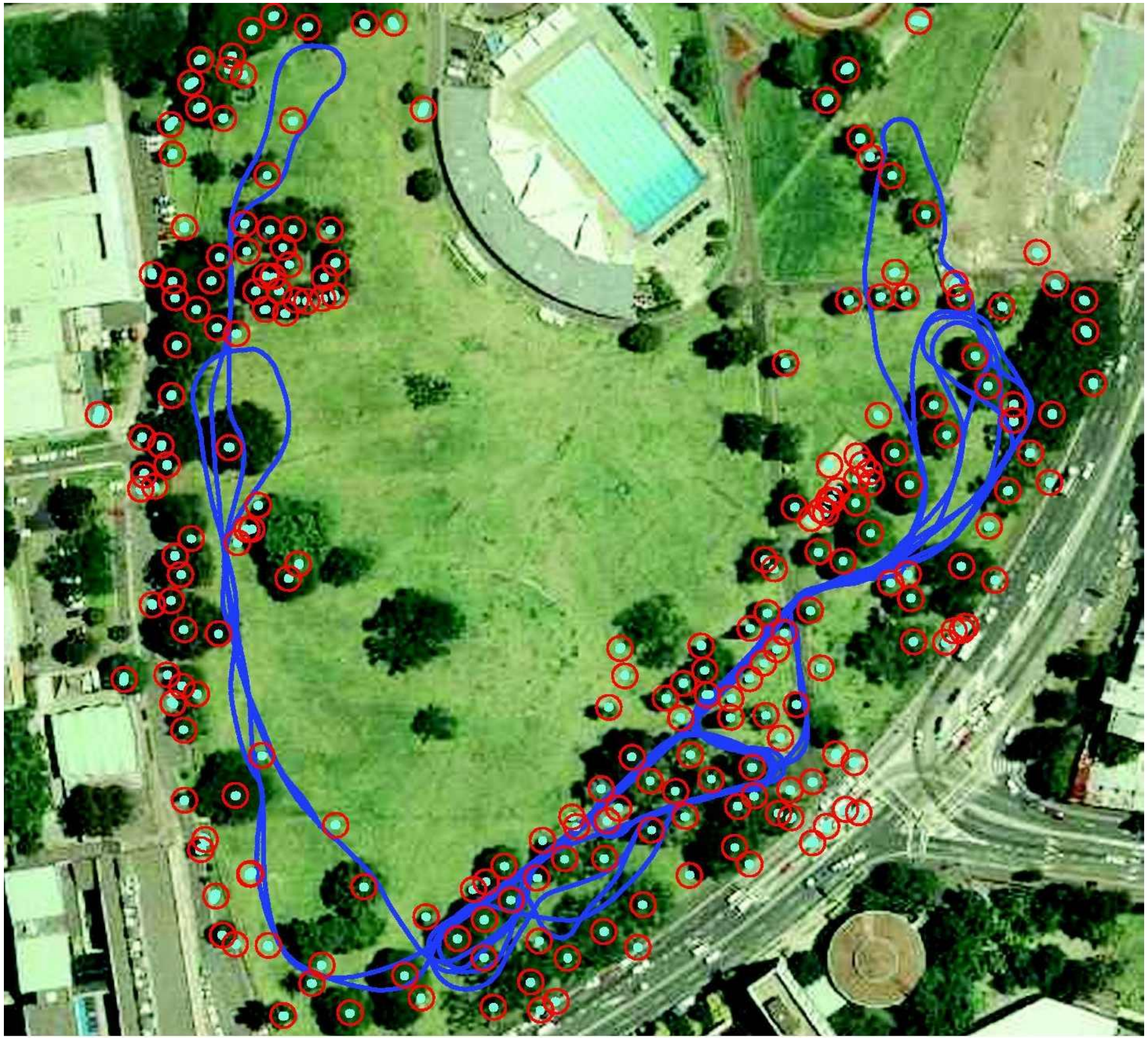


Victoria Park: Estimated Trajectory



[courtesy by E. Nebot]

Victoria Park: Landmarks

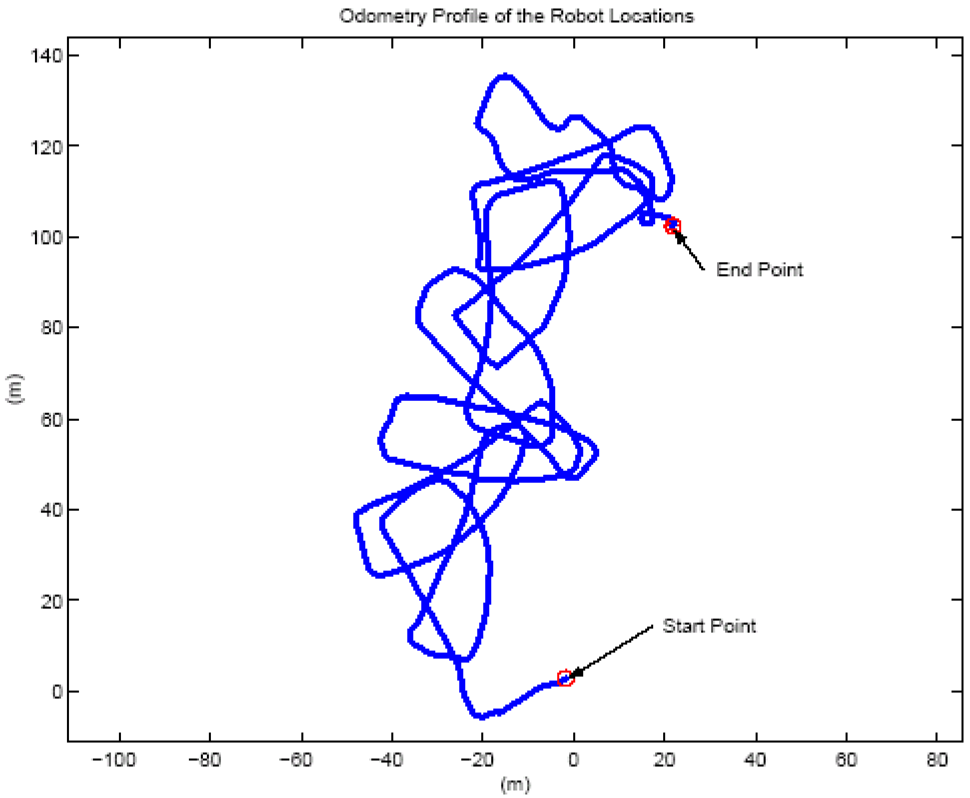


[courtesy by E. Nebot]

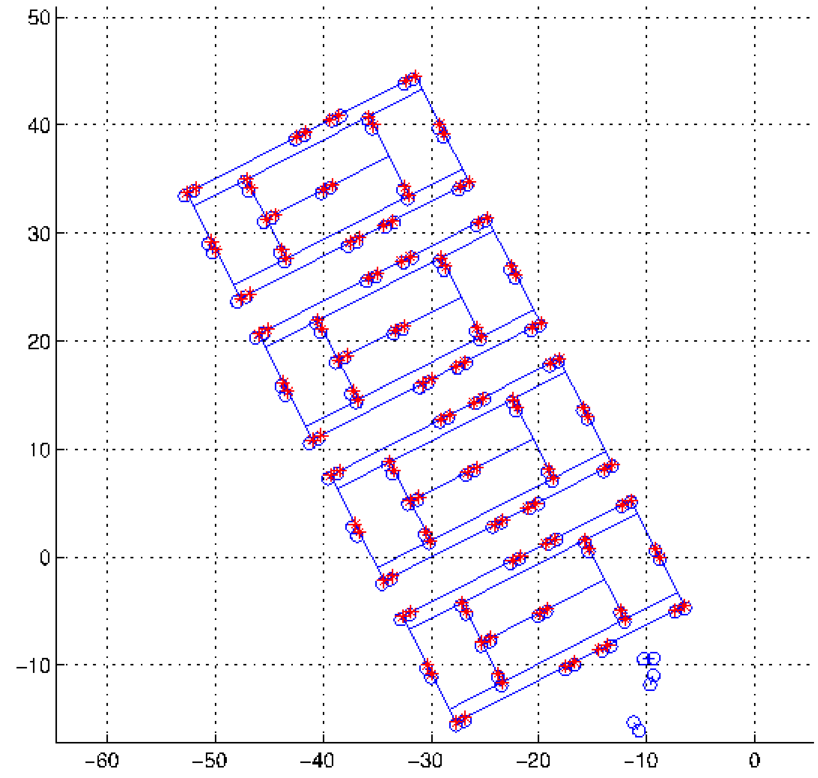
EKF SLAM Example: Tennis Court

[courtesy by J. Leonard]

EKF SLAM Example: Tennis Court



odometry

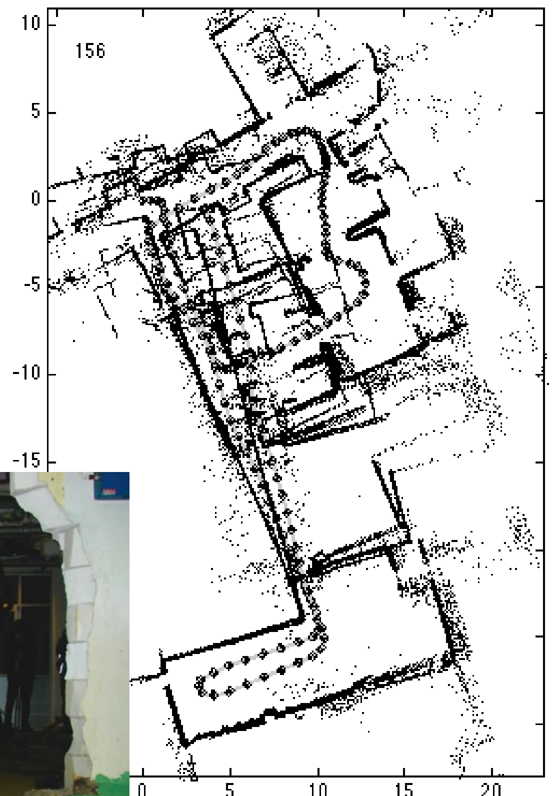


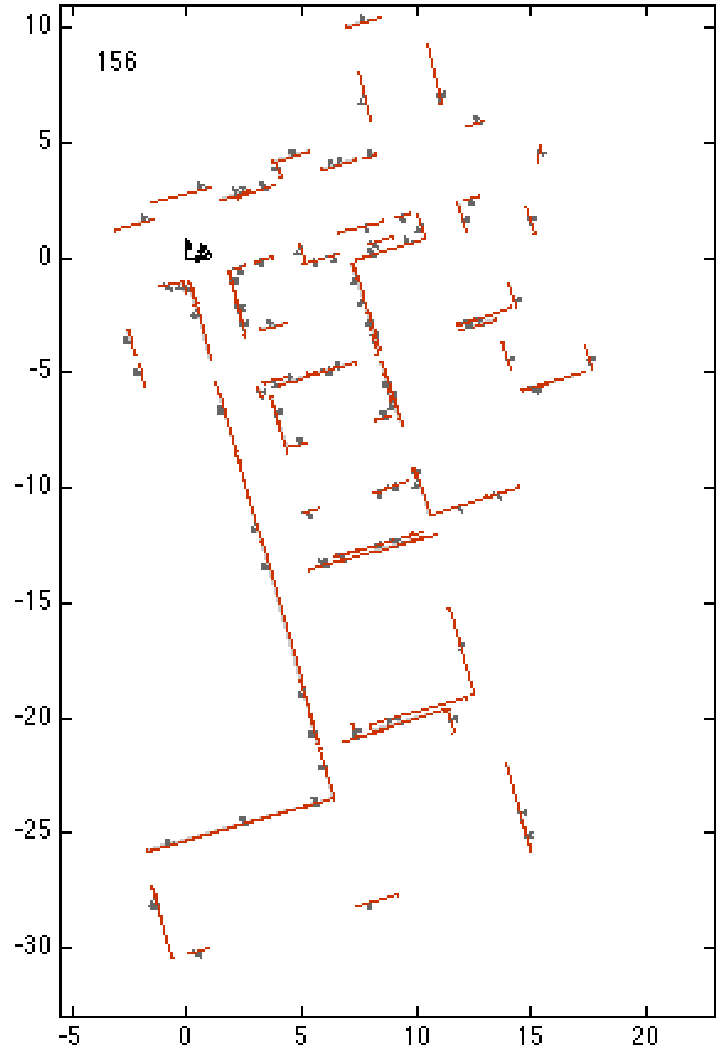
estimated trajectory

[courtesy by John Leonard]

EKF SLAM Example: Line Features

- KTH Bakery Data Set





[Wulf et al., ICRA 04]

EKF-SLAM: Complexity

- **Cost per step**: quadratic in n, the number of landmarks: $O(n^2)$
- **Total cost** to build a **map** with n landmarks: $O(n^3)$
- **Memory**: $O(n^2)$

Problem: becomes computationally intractable for large maps!

➔ Approaches exist that make EKF-SLAM amortized $O(n)$ / $O(n^2)$ / $O(n^2)$
D&C SLAM *[Paz et al., 2006]*

SLAM Techniques

- EKF SLAM
- FastSLAM
- Graphical SLAM
- Topological SLAM
  (mainly place recognition)
- Scan Matching / Visual Odometry
  (only locally consistent maps)
- Approximations for SLAM: Local submaps, Sparse extended information filters, Sparse links, Thin junction tree filters, etc.

Introduction to Mobile Robotics

SLAM – Grid-based FastSLAM

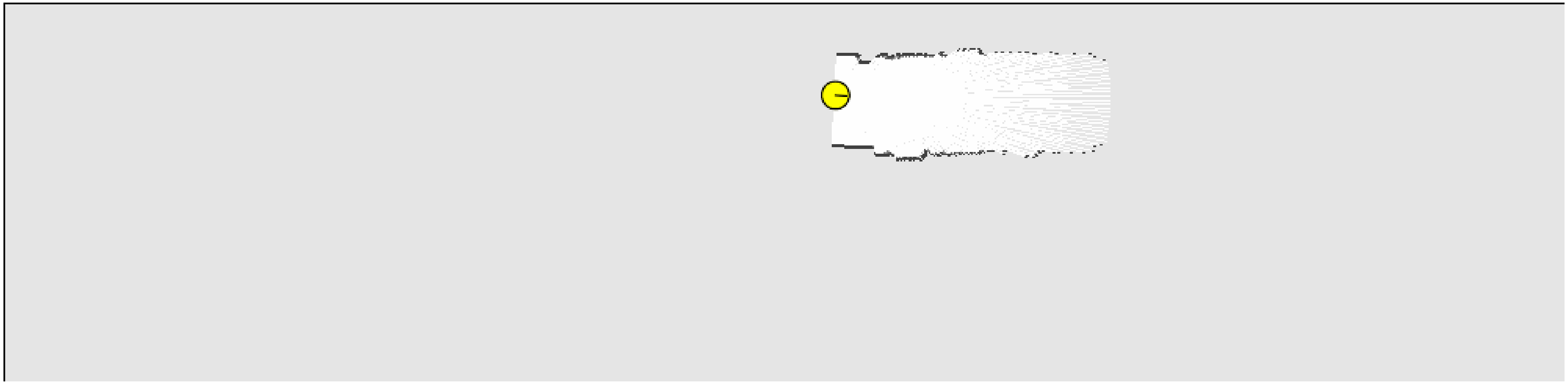
Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

Grid-based SLAM

- Can we solve the SLAM problem if no pre-defined landmarks are available?
- Can we use the ideas of FastSLAM to build grid maps?
- As with landmarks, the map depends on the poses of the robot during data acquisition
- If the poses are known, grid-based mapping is easy ("mapping with known poses")

Mapping with Known Poses



- Mapping with known poses using laser range data

Rao-Blackwellization

poses map observations & movements

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

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

Factorization first introduced by Murphy in 1999

Rao-Blackwellization

poses map observations & movements

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

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

SLAM posterior

Robot path posterior

Mapping with known poses

Factorization first introduced by Murphy in 1999

Rao-Blackwellization

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

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

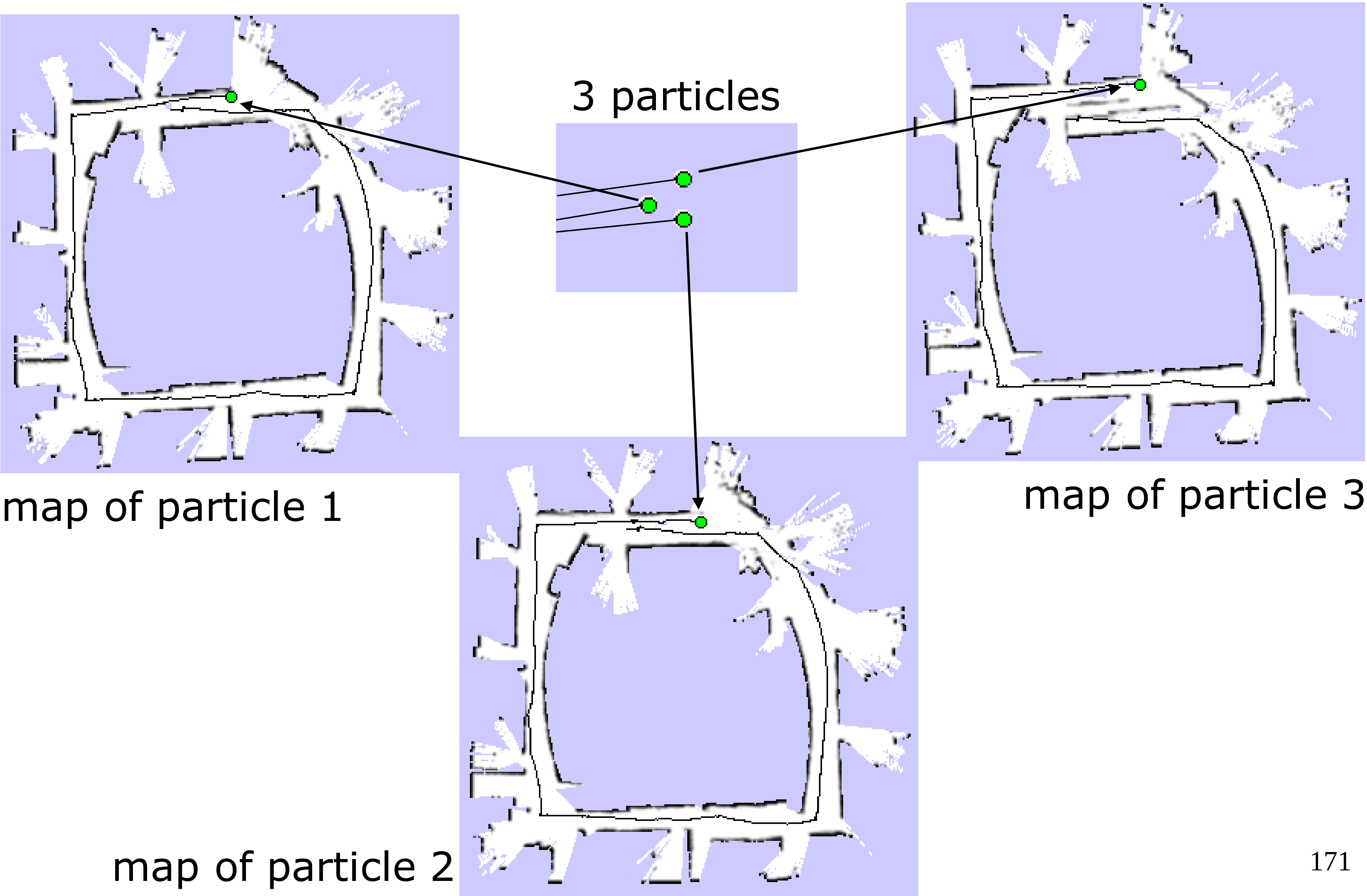
This is localization, use MCL

Use the pose estimate from the MCL and apply mapping with known poses

Rao-Blackwellized Mapping

- Each particle represents a possible trajectory of the robot
- Each particle
  - maintains its own map and
  - updates it upon "mapping with known poses"
- Each particle survives with a probability proportional to the likelihood of the observations relative to its own map

Particle Filter Example



Problem

- Each map is quite big in case of grid maps
- Since each particle maintains its own map
- Therefore, one needs to keep the number of particles small

- **Solution**:
  Compute better proposal distributions!
- **Idea**:
  Improve the pose estimate **before** applying the particle filter

Pose Correction Using Scan Matching

Maximize the likelihood of the i-th pose and map relative to the (i-1)-th pose and map

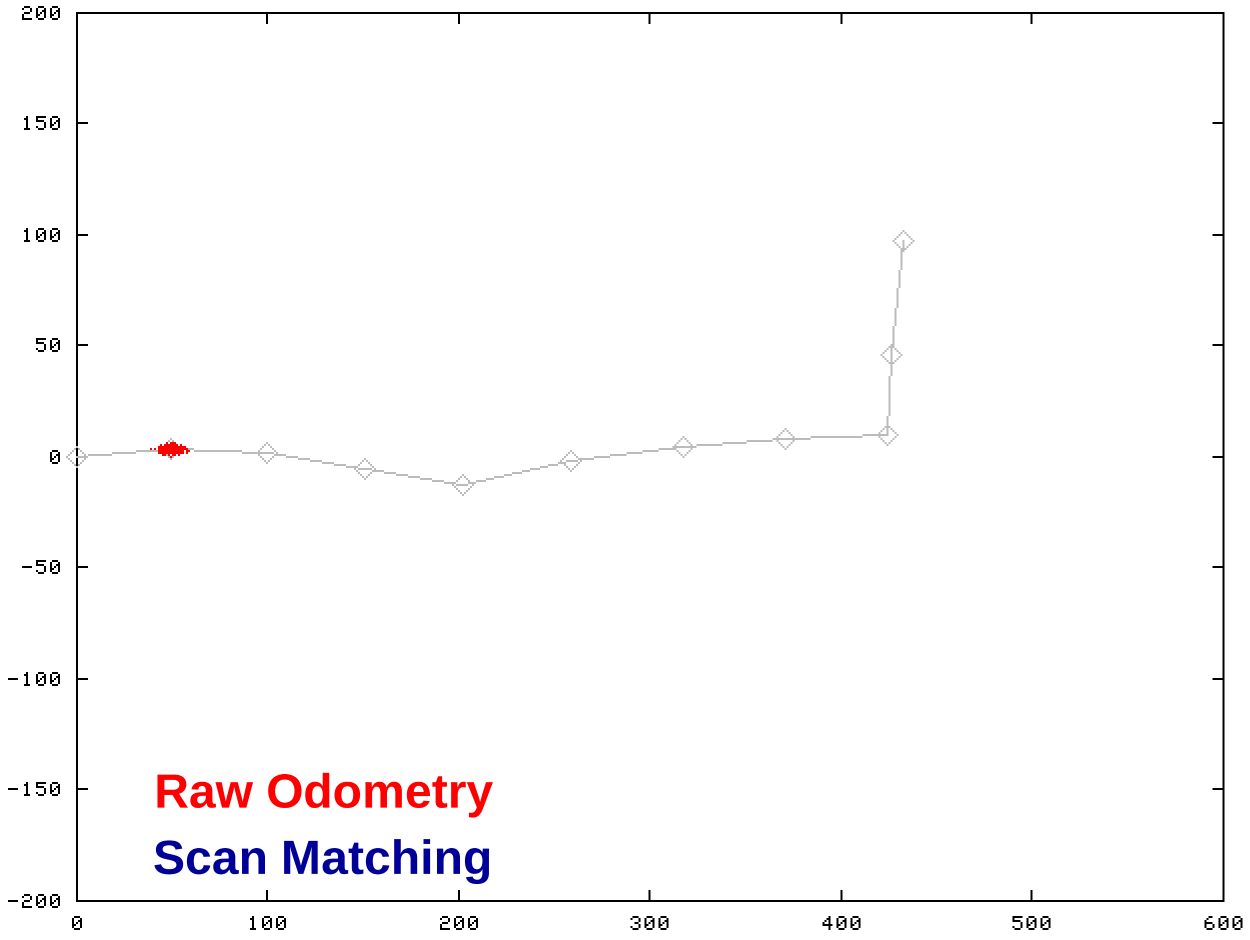
$$\hat{x}_t = \underset{x_t}{\arg\max} \left\{ p(z_t \mid x_t, \hat{m}_{t-1}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current measurement

robot motion

map constructed so far

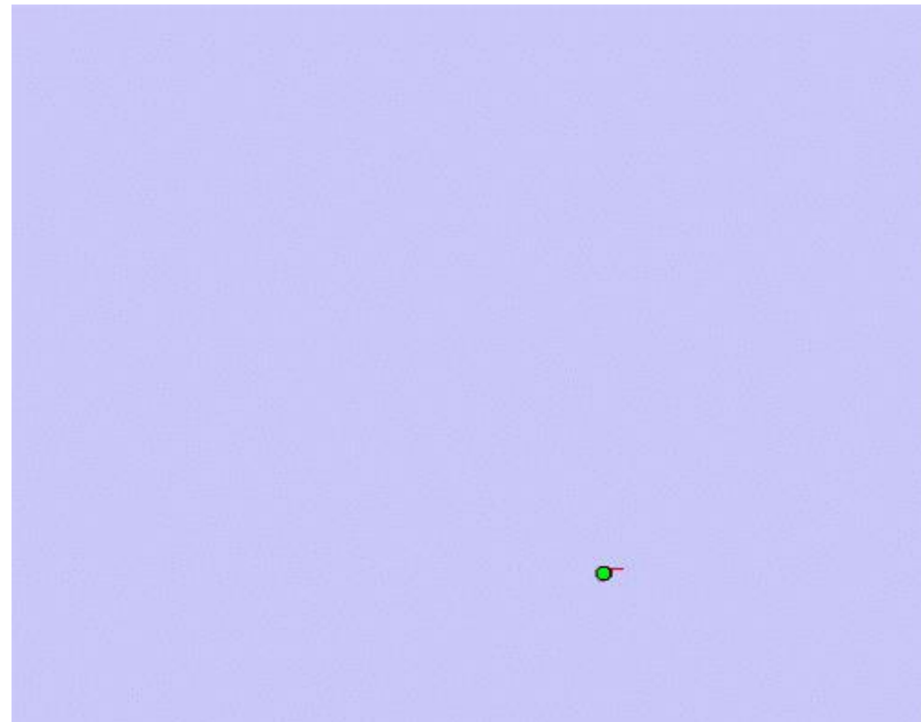
Motion Model for Scan Matching



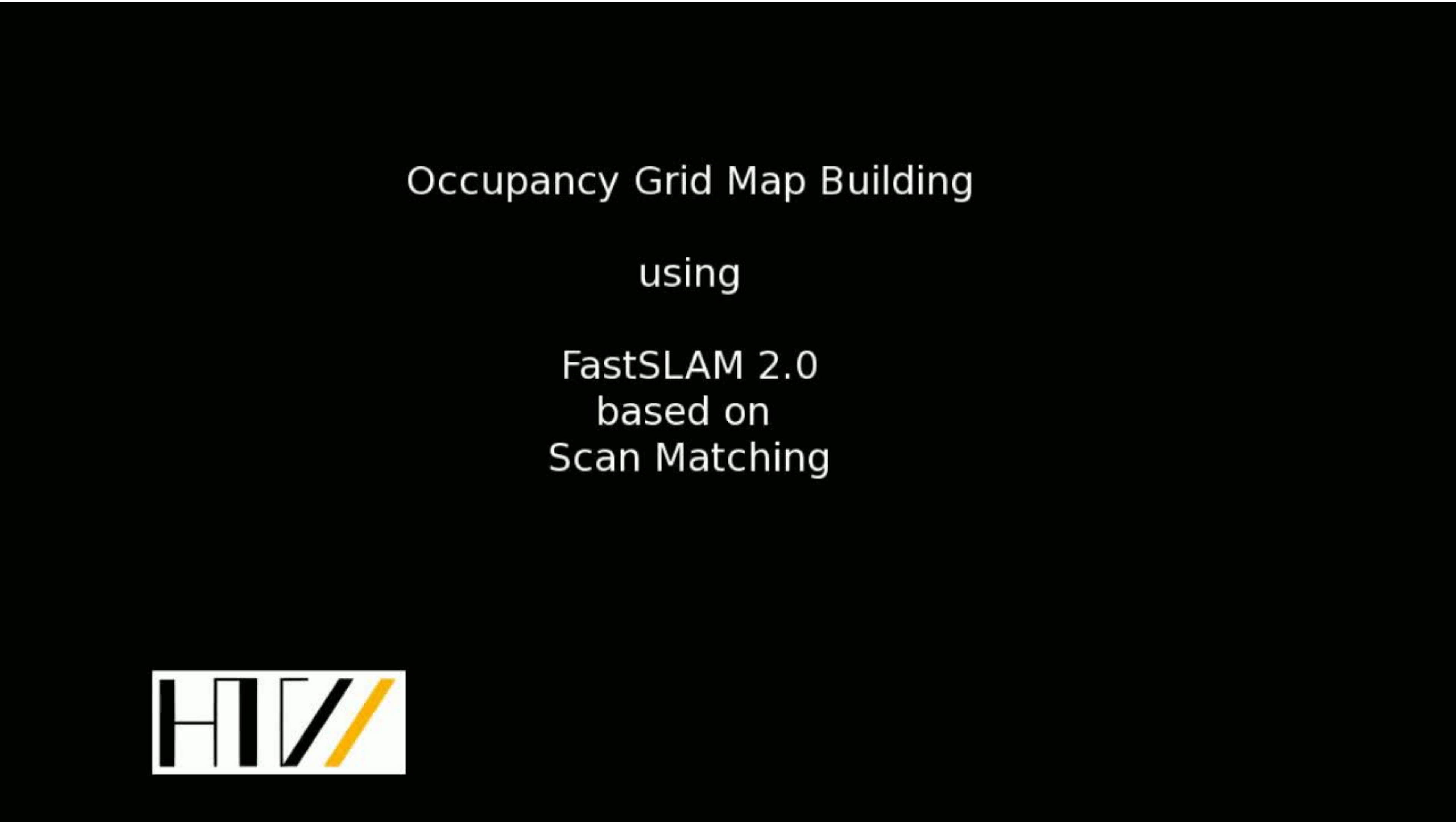
FastSLAM with Improved Odometry

- Scan-matching provides a **locally consistent** pose correction
- Pre-correct short odometry sequences using scan-matching and use them as input to FastSLAM
- Fewer particles are needed, since the error in the input in smaller

[Haehnel et al., 2003]

FastSLAM with Scan-Matching



Conclusion (so far…)

- The presented approach is a highly efficient algorithm for SLAM combining ideas of scan matching and FastSLAM
- Scan matching is used to transform sequences of laser measurements into odometry measurements
- This version of grid-based FastSLAM can handle larger environments than before in “real time”

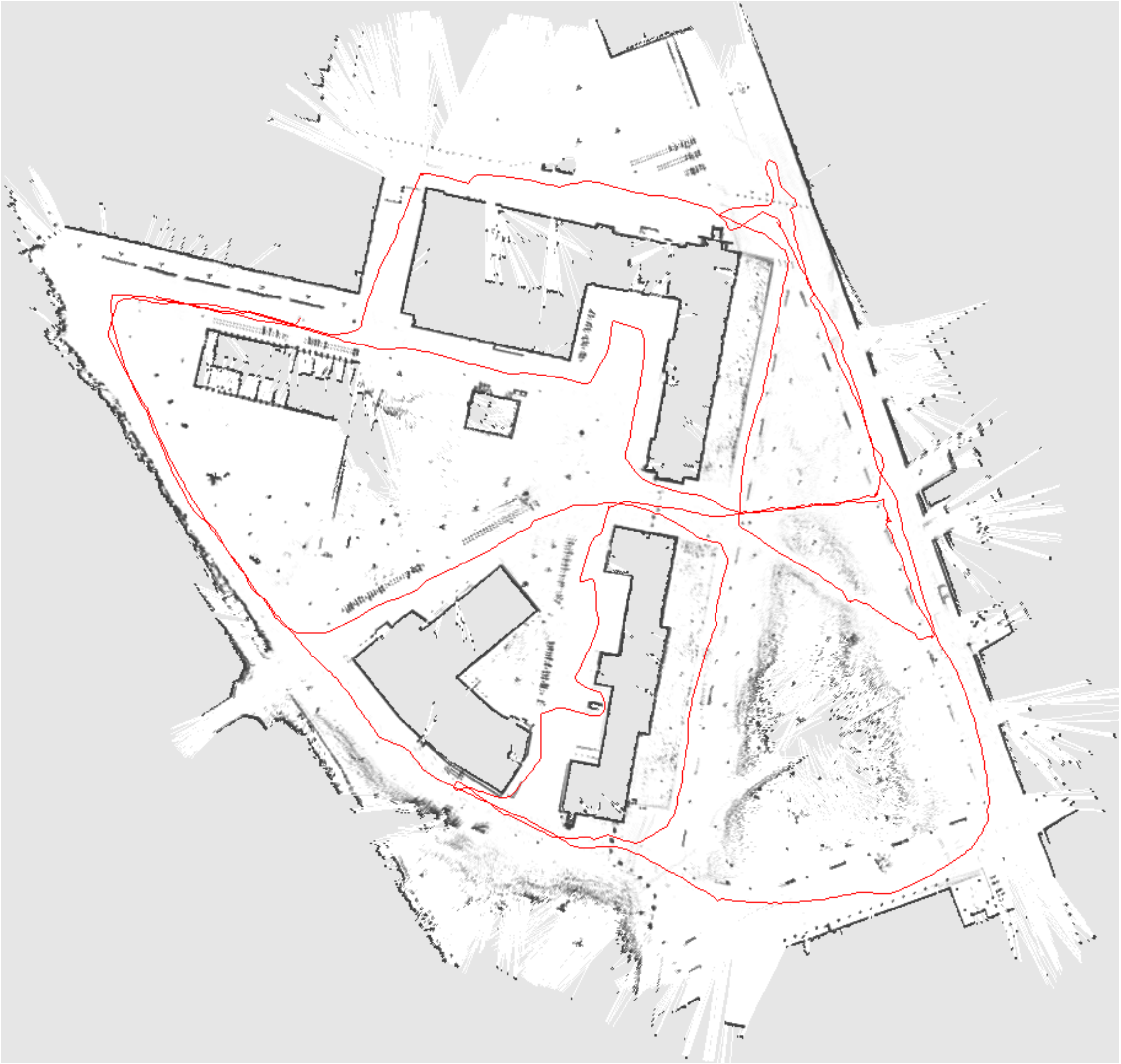
What’s Next?

- Further reduce the number of particles
- Improved proposals will lead to more accurate maps
- Use the properties of our sensor when drawing the next generation of particles

Intel Lab

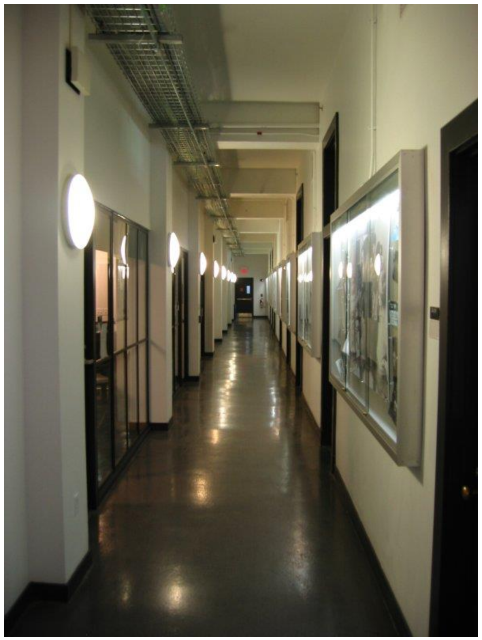
- **15 particles**
- four times faster than real-time P4, 2.8GHz
- 5cm resolution during scan matching
- 1cm resolution in final map

Outdoor Campus Map



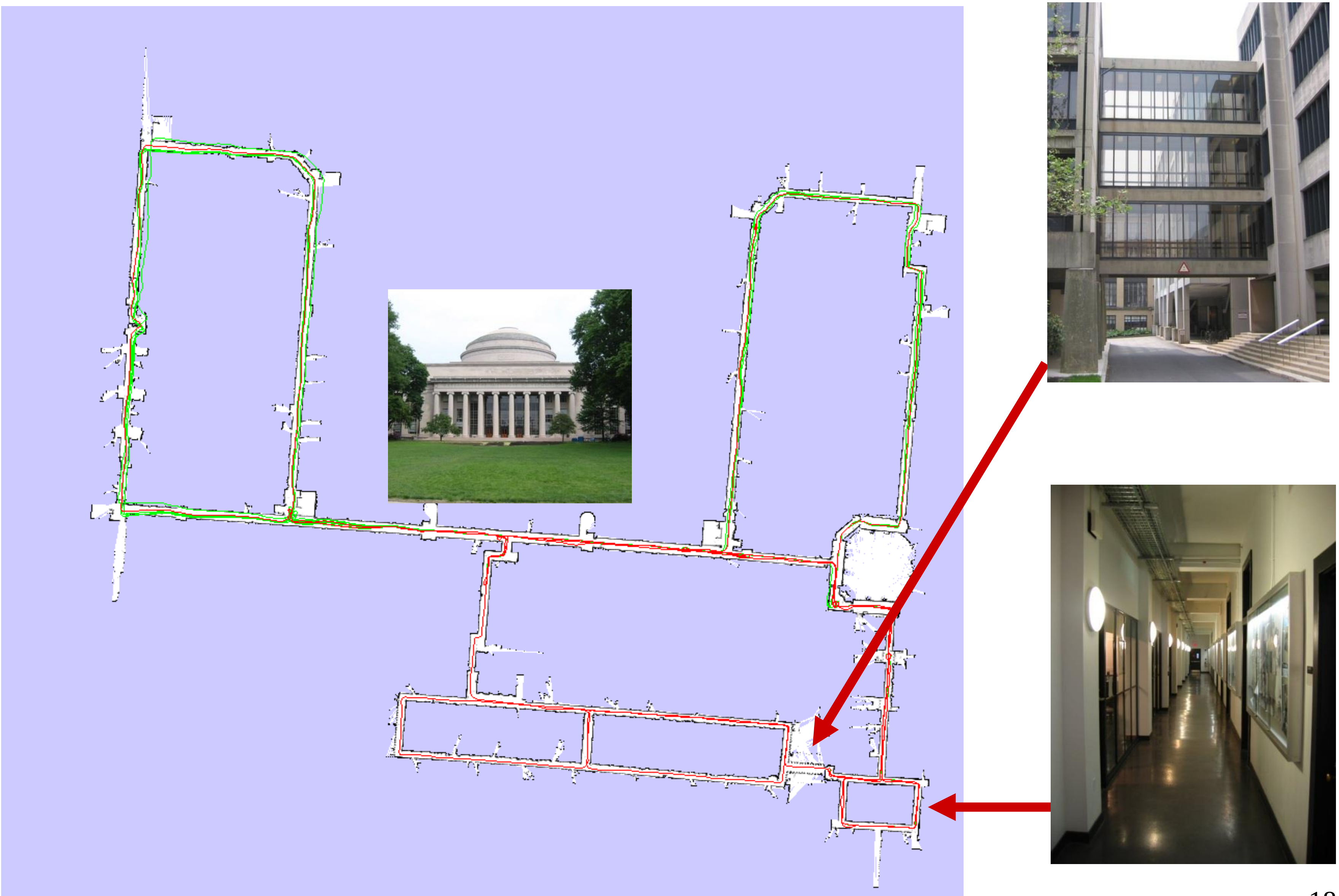
- **30 particles**
- 250x250m^2
- 1.75 km (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map

MIT Killian Court

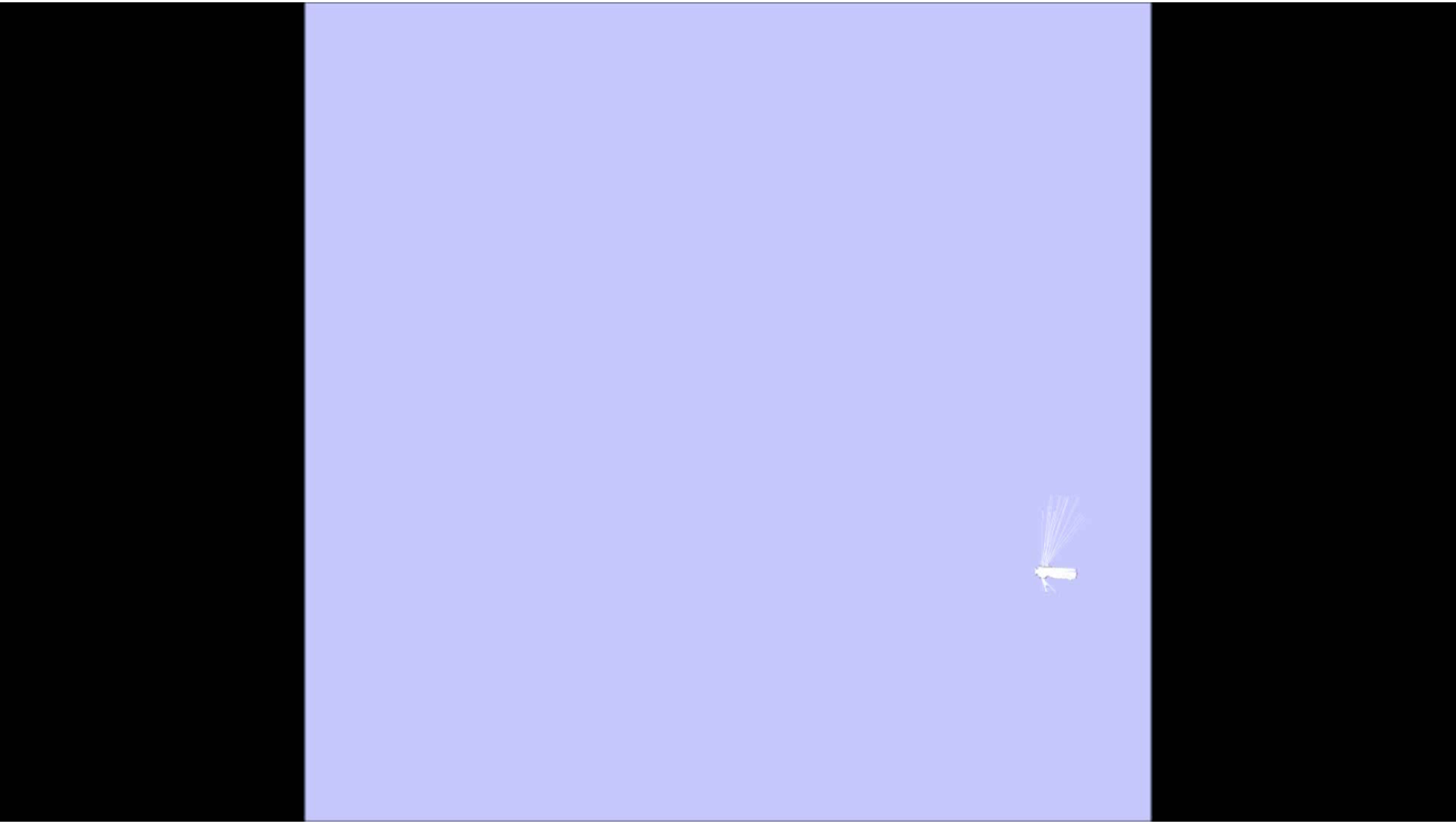


- The **"infinite-corridor-dataset"** at MIT

MIT Killian Court



MIT Killian Court - Video



Conclusion

- The ideas of FastSLAM can also be applied in the context of grid maps
- Utilizing accurate sensor observation leads to good proposals and highly efficient filters
- It is similar to scan-matching on a per-particle base
- The number of necessary particles and re-sampling steps can seriously be reduced
- Improved versions of grid-based FastSLAM can handle larger environments than naïve implementations in “real time” since they need one order of magnitude fewer samples

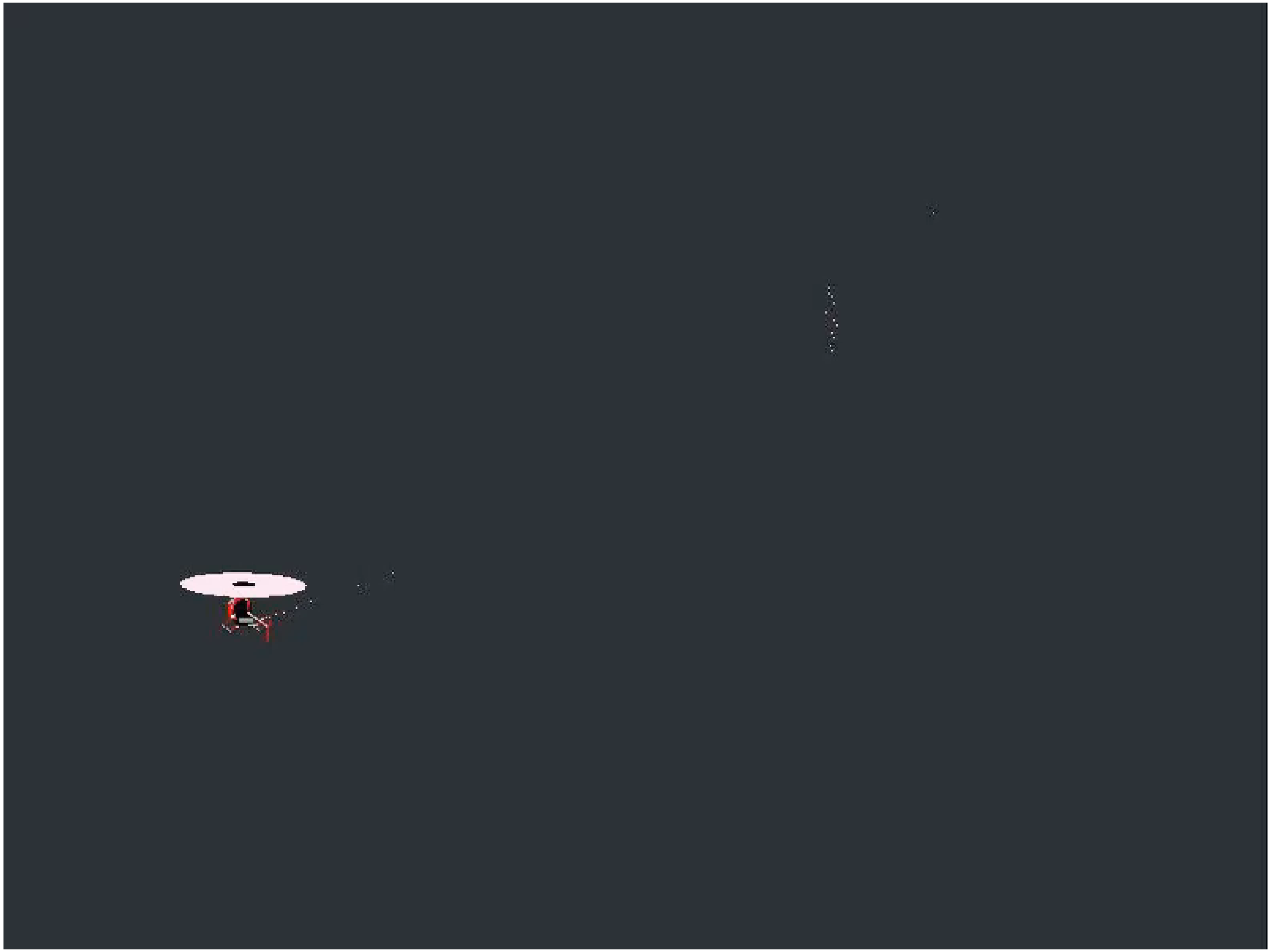
More Details on FastSLAM

- M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping, *AAAI02*
  (The classic FastSLAM paper with landmarks)

- D. Haehnel, W. Burgard, D. Fox, and S. Thrun. An efcient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, IROS03
  (FastSLAM on grid-maps using scan-matched input)

- G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling, ICRA05
  (Proposal using laser observation, adaptive resampling)

- A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks, IJCAI03
  (A representation to handle big particle sets)



Sept 2011

WARWICK MOBILE ROBOTICS

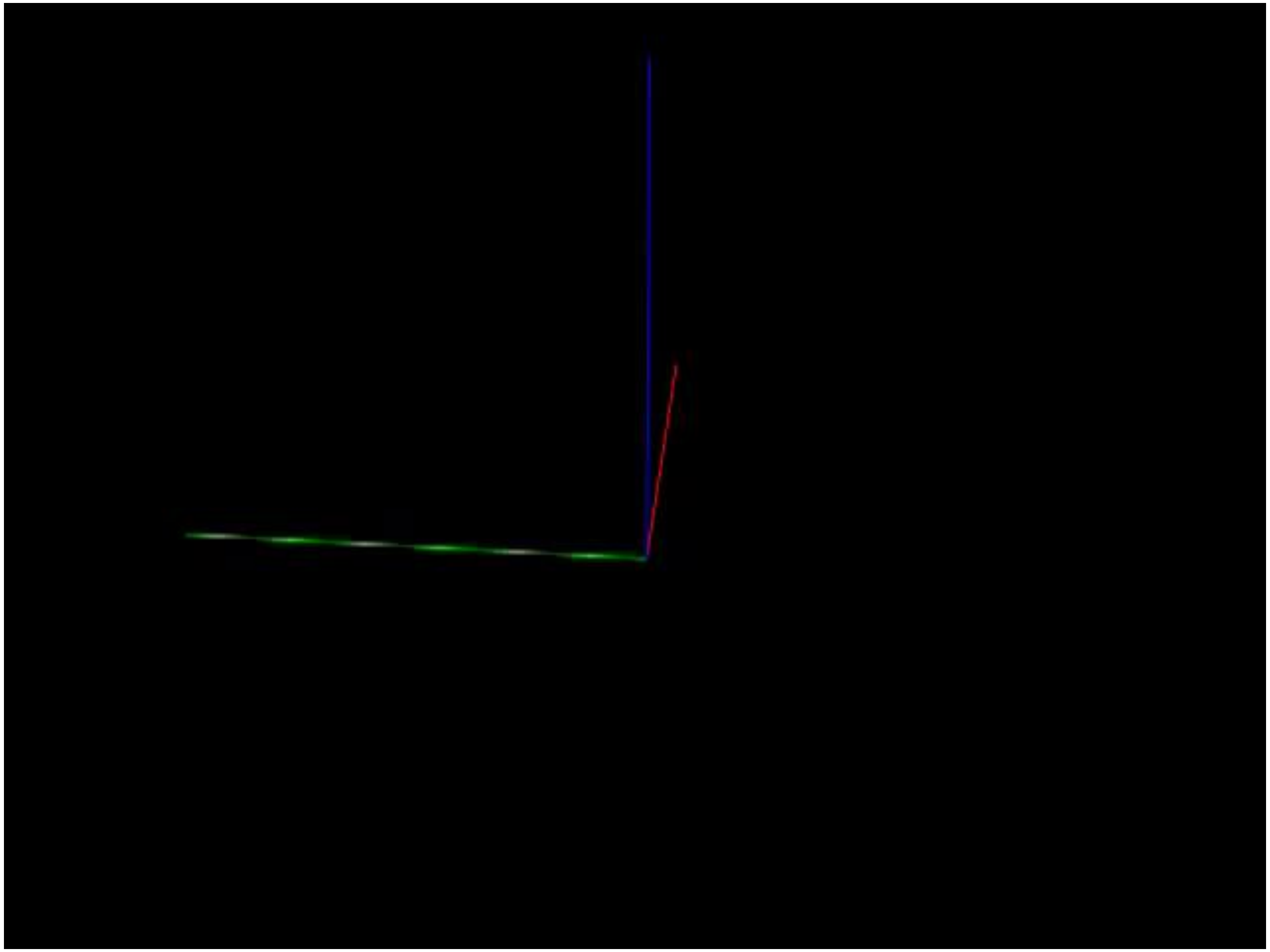
WMR

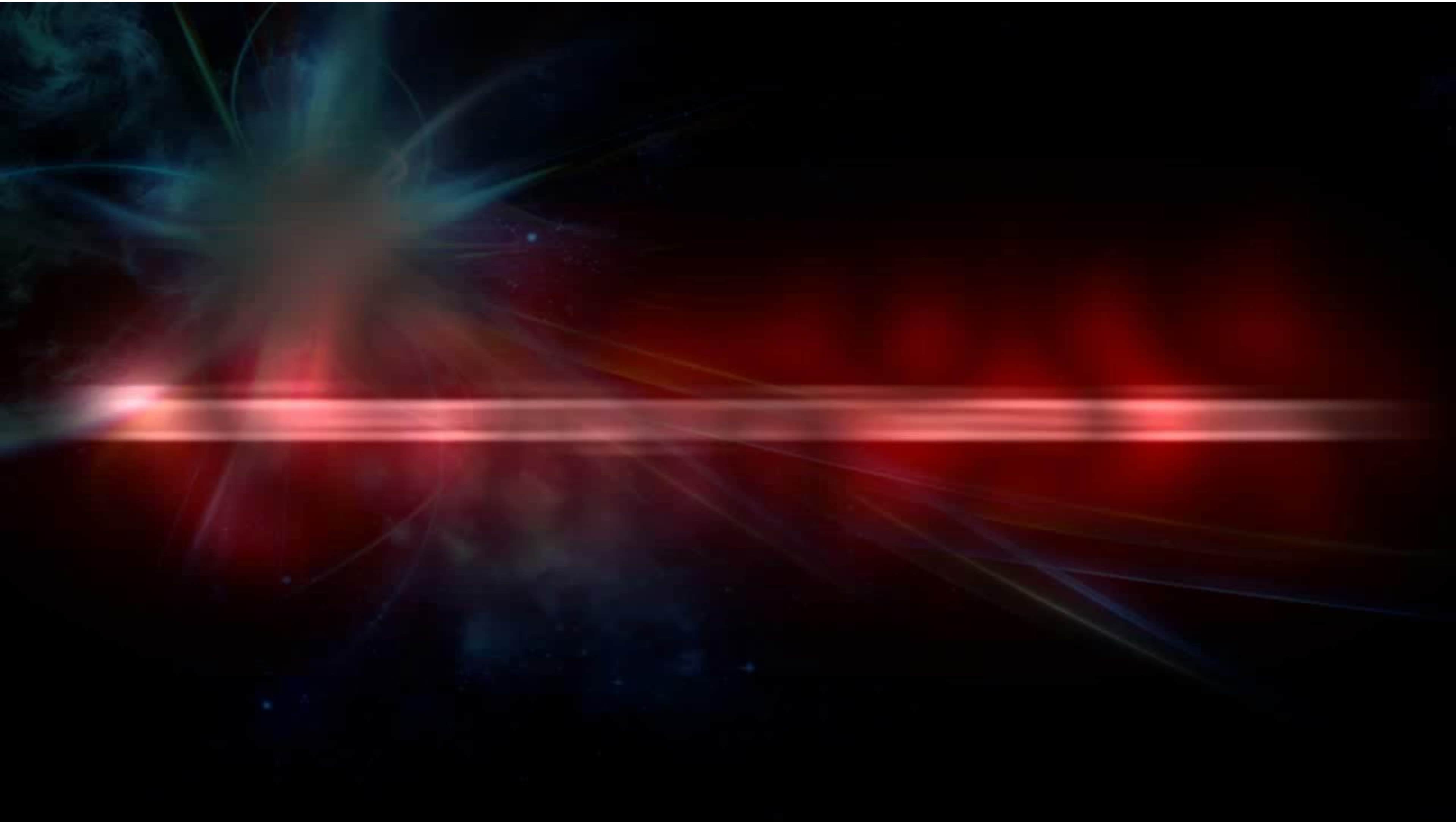
Indoor Aerial Inspection Vehicle

LiDAR Mapping Demo #2

THE UNIVERSITY OF WARWICK

Sellafield Ltd EPSRC WMG Innovative Solutions WARWICK IMRC Innovative Manufacturing Research Centre





Introduction to Mobile Robotics

Robot Motion Planning

Wolfram Burgard, Cyrill Stachniss,
Maren Bennewitz, Kai Arras

Slides by Kai Arras Last update July 2011

With material from S. LaValle, JC. Latombe, H. Choset et al., W. Burgard

Robot Motion Planning

J.-C. Latombe (1991):

"…eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world."

Goals

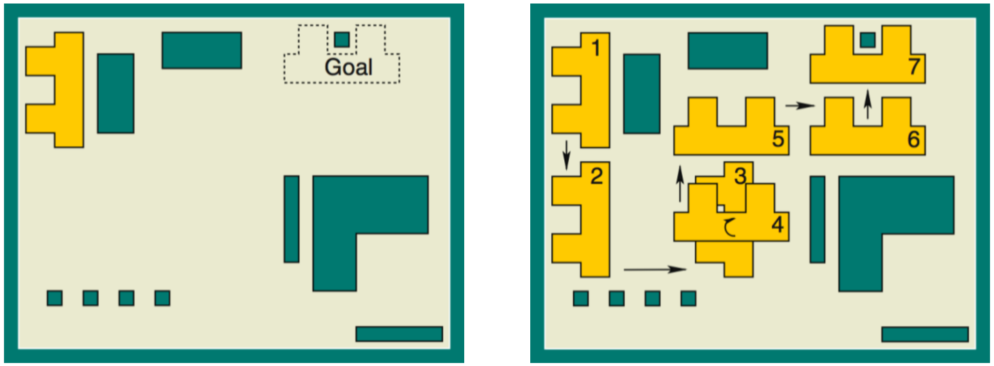
- Collision-free trajectories
- Robot should reach the goal location as fast as possible

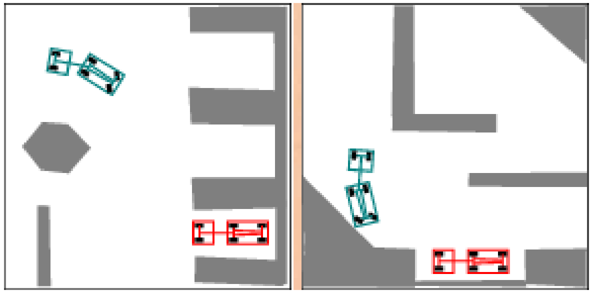
Problem Formulation

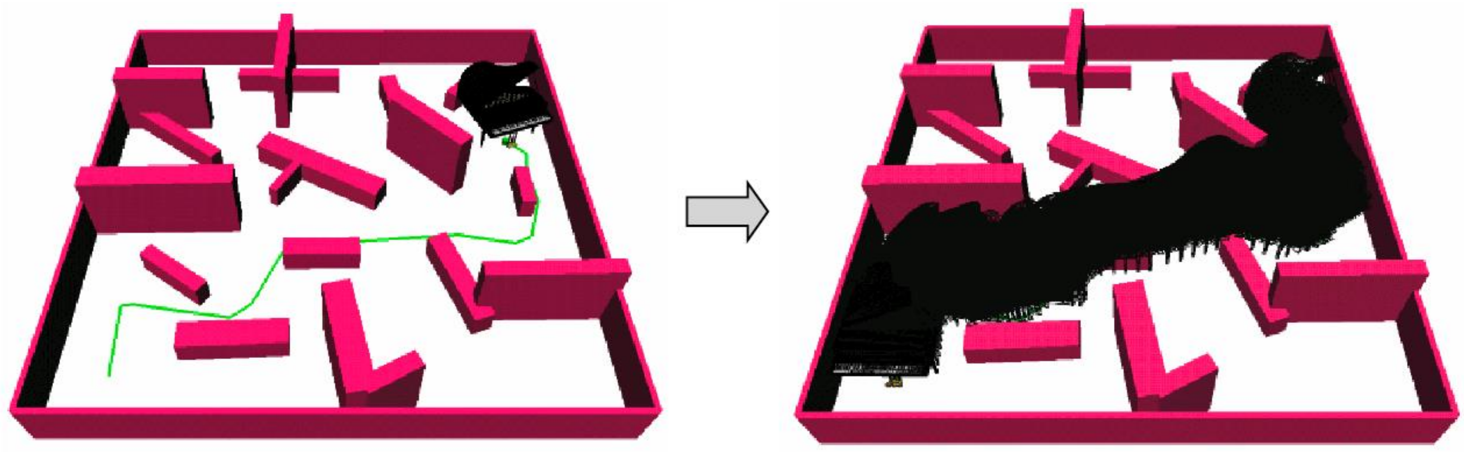
- The **problem of motion planning** can be stated as follows. Given:
  - A **start** pose of the robot
  - A desired **goal** pose
  - A geometric description of the **robot**
  - A geometric description of the **world**

- Find a path that moves the robot gradually from **start** to **goal** while **never touching** any obstacle

Problem Formulation

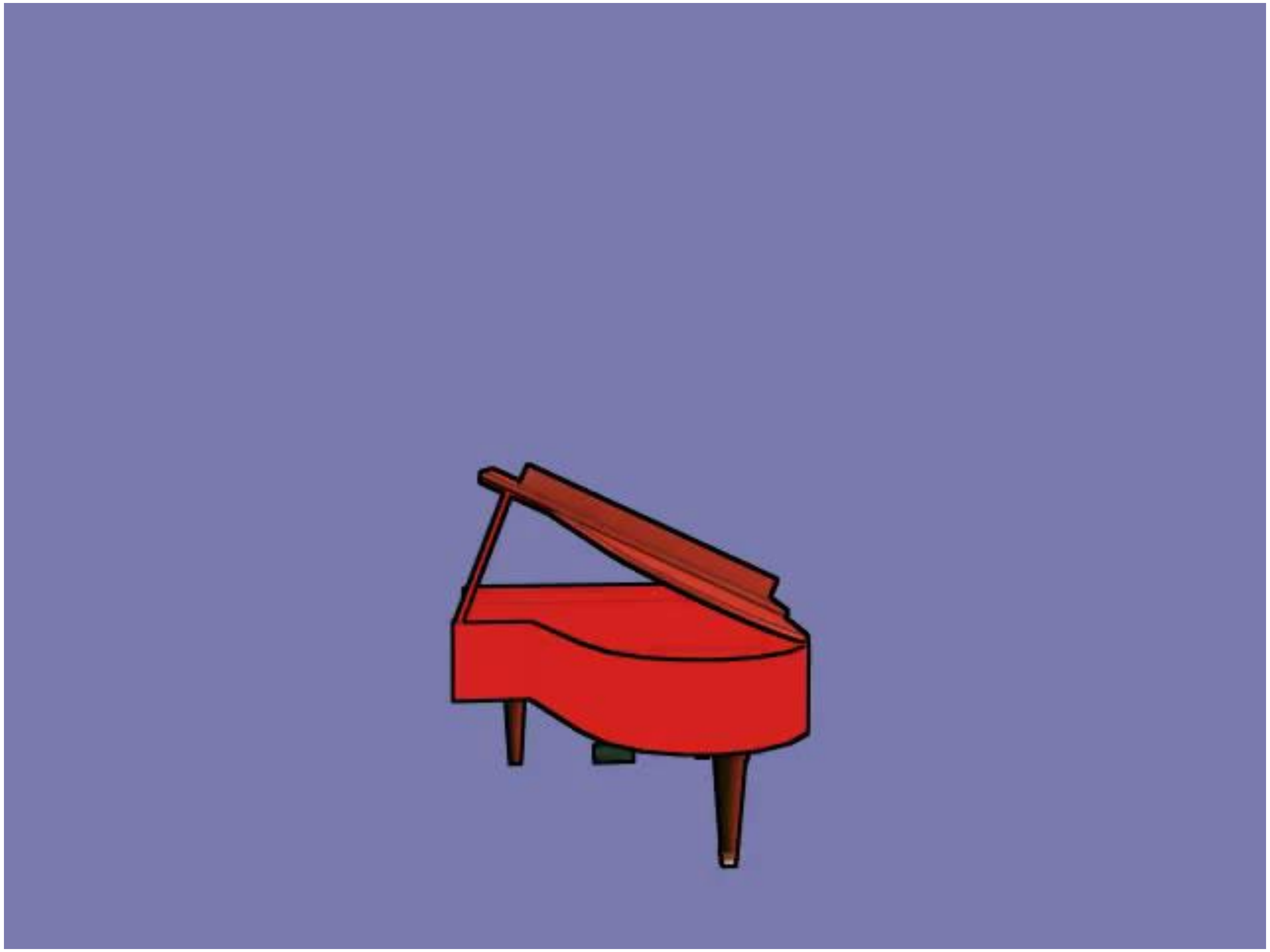






Motion planning is sometimes also called **piano mover's problem**

Piano mover's problem



Configuration Space

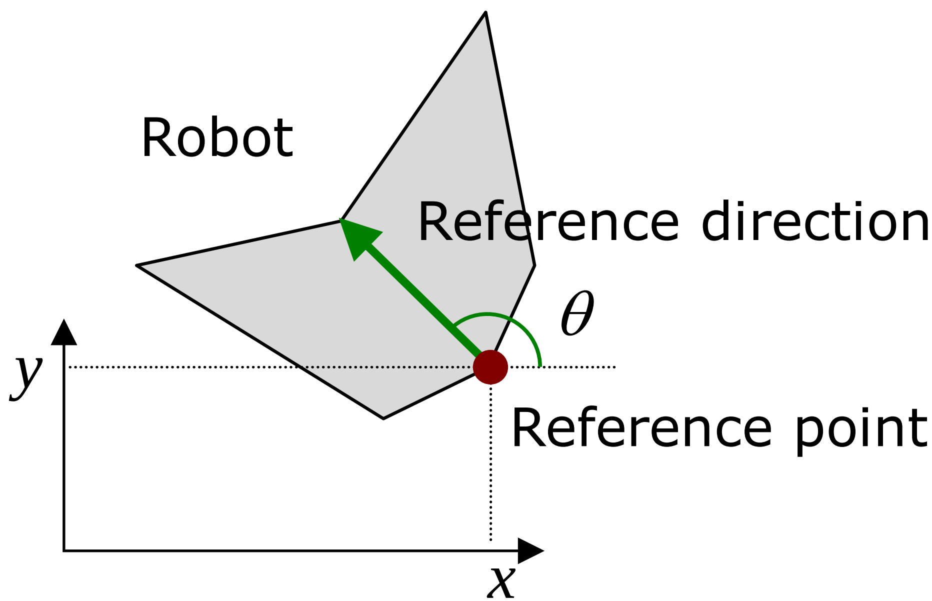
- Although the motion planning problem is defined in the regular world, it lives in another space: the **configuration space**

- A robot configuration q is a specification of the positions of all robot points relative to a fixed coordinate system

- Usually a configuration is expressed as a **vector of positions** and **orientations**

Configuration Space

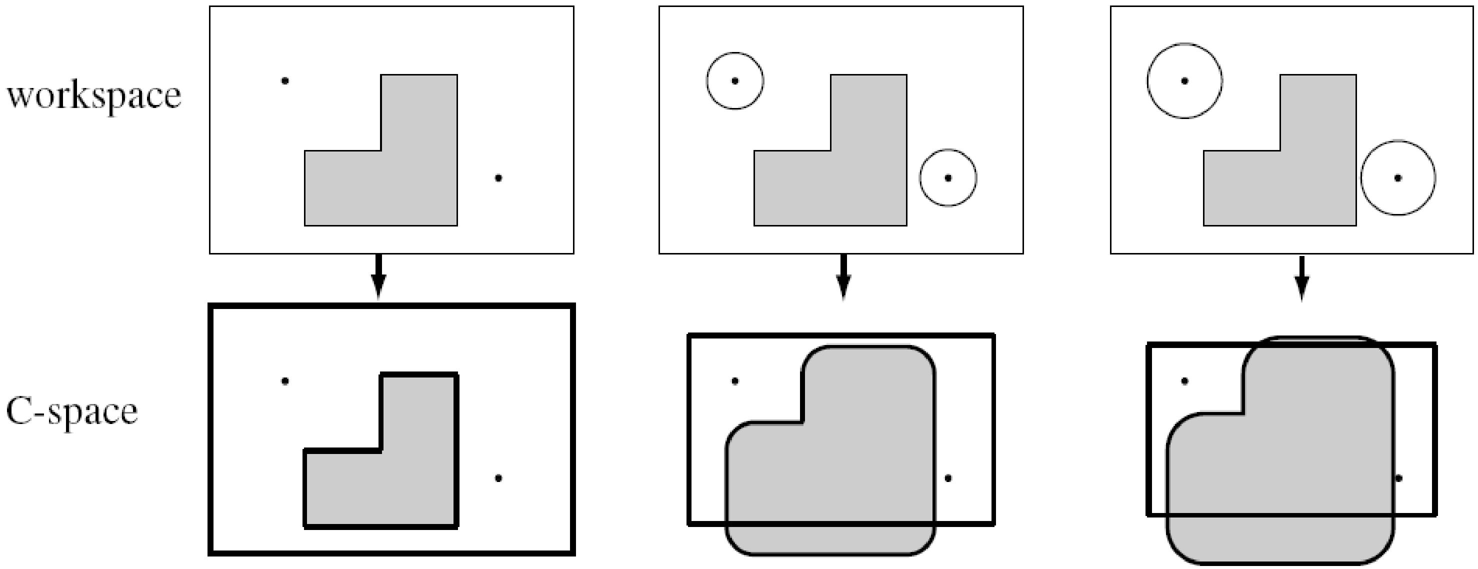
Rigid-body robot example



- 3-parameter representation: $q = (x,y,\theta)$
- In 3D, q would be of the form $(x,y,z,\alpha,\beta,\gamma)$

Configuration Space

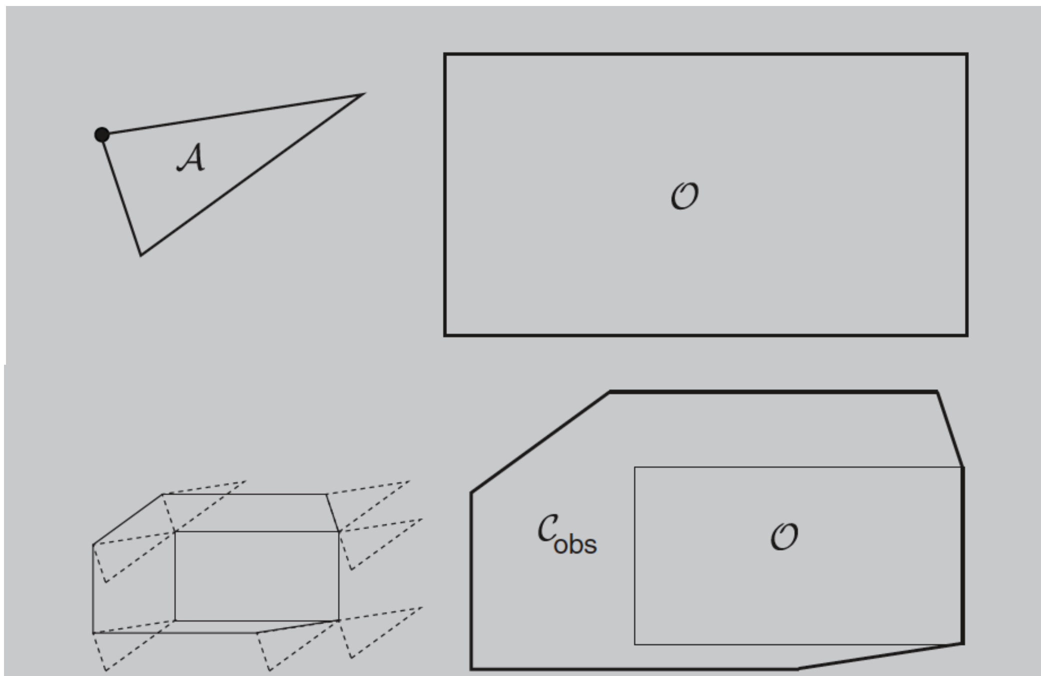
- Example: circular robot



- C-space is obtained by sliding the robot along the edge of the obstacle regions "blowing them up" by the robot radius

Configuration Space

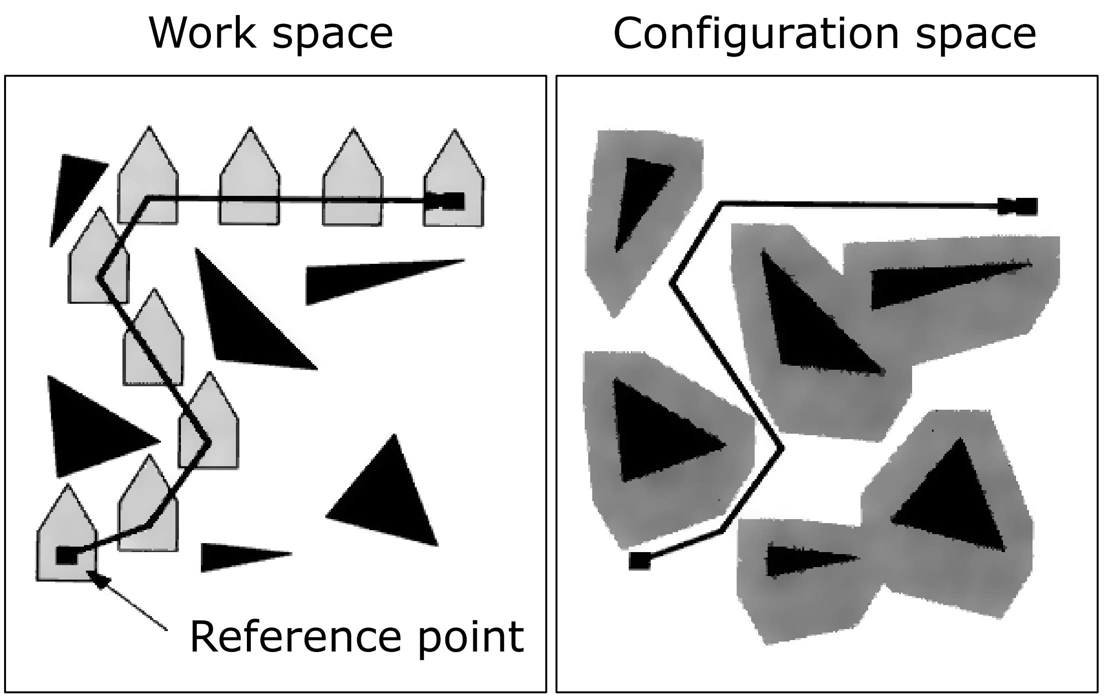
- Example: polygonal robot, translation only



- C-space is obtained by sliding the robot along the edge of the obstacle regions

Configuration Space

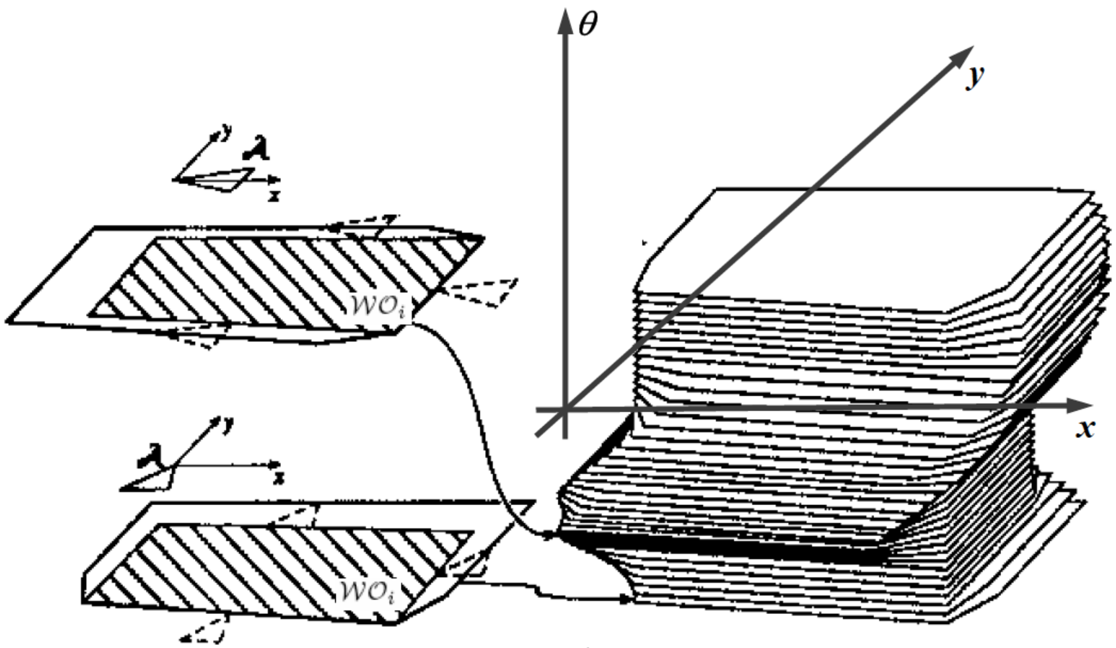
- Example: polygonal robot, translation only



- C-space is obtained by sliding the robot along the edge of the obstacle regions

Configuration Space

- Example: polygonal robot, trans+**rotation**



- C-space is obtained by sliding the robot along the edge of the obstacle regions in all orientations

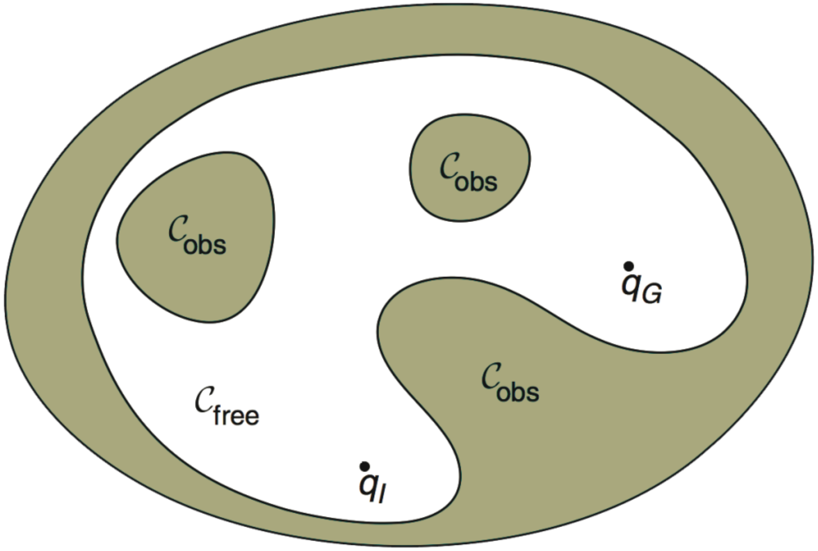
Configuration Space

Free space and **obstacle region**

- With $\mathcal{W} = \mathbb{R}^m$ being the work space, $\mathcal{O} \in \mathcal{W}$ the set of obstacles, $\mathcal{A}(q)$ the robot in configuration $q \in \mathcal{C}$

$$
\begin{aligned}
\mathcal{C}_{free} &= \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset\} \\
\mathcal{C}_{obs} &= \mathcal{C} / \mathcal{C}_{free}
\end{aligned}
$$

- We further define
  q_I : start configuration
  q_G : goal configuration



Configuration Space

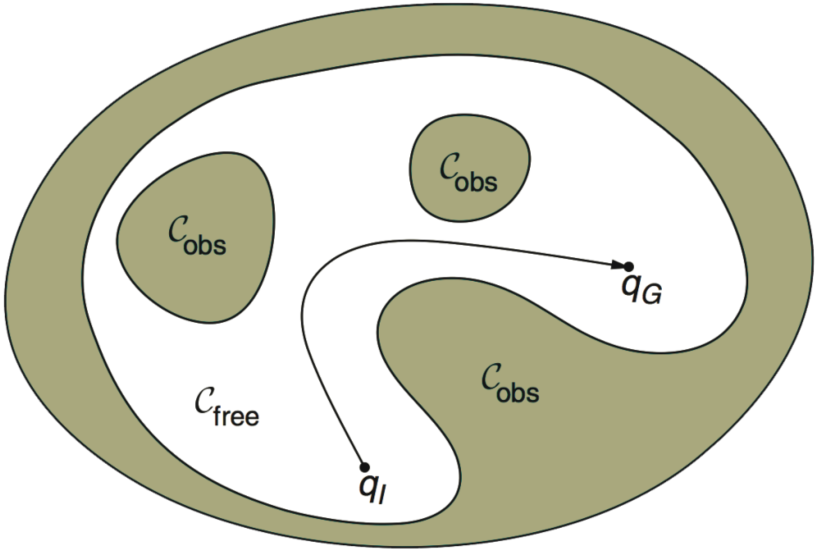
Then, motion planning amounts to

- Finding a continuous path

  $$\tau : [0,1] \rightarrow \mathcal{C}_{free}$$

  with $\tau(0) = q_I\,,\ \tau(1) = q_G$

- Given this setting, we can do planning with the robot being a **point in C-space!**



C-Space Discretizations

- Continuous terrain needs to be **discretized** for path planning
- There are **two general approaches** to discretize C-spaces:
  - **Combinatorial planning**

    Characterizes C_{free} explicitely by capturing the connectivity of C_{free} into a graph and finds solutions using search

  - **Sampling-based planning**

    Uses collision-detection to probe and incrementally search the C-space for solution

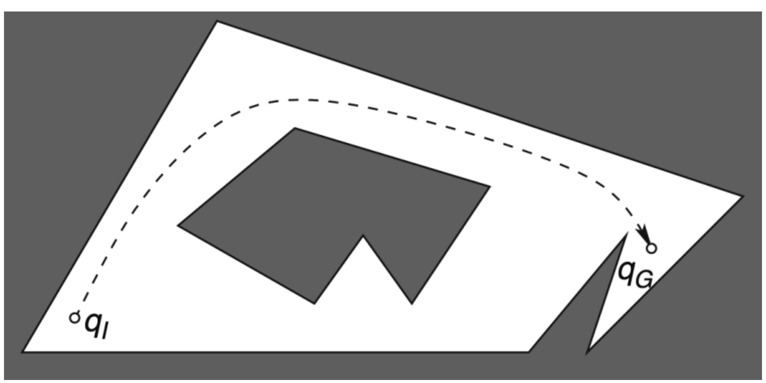
Combinatorial Planning

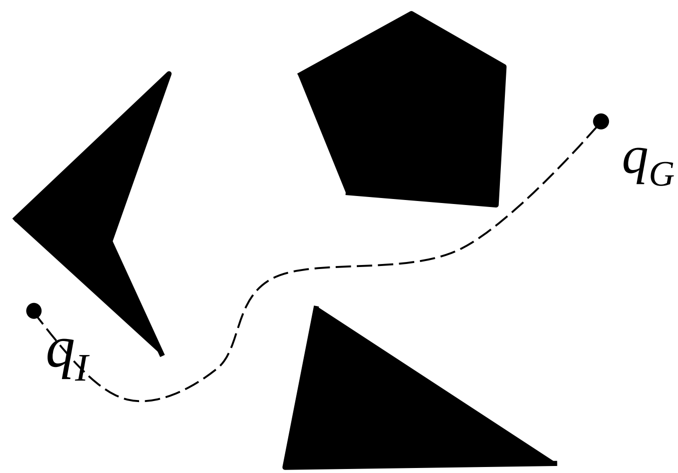
- We will look at four **combinatorial planning techniques**
  - Visibility graphs
  - Voronoi diagrams
  - Exact cell decomposition
  - Approximate cell decomposition

- They all produce a **road map**
  - A **road map** is a **graph in $\boldsymbol{C_{free}}$** in which each vertex is a configuration in C_{free} and each edge is a collision-free path through C_{free}

Combinatorial Planning

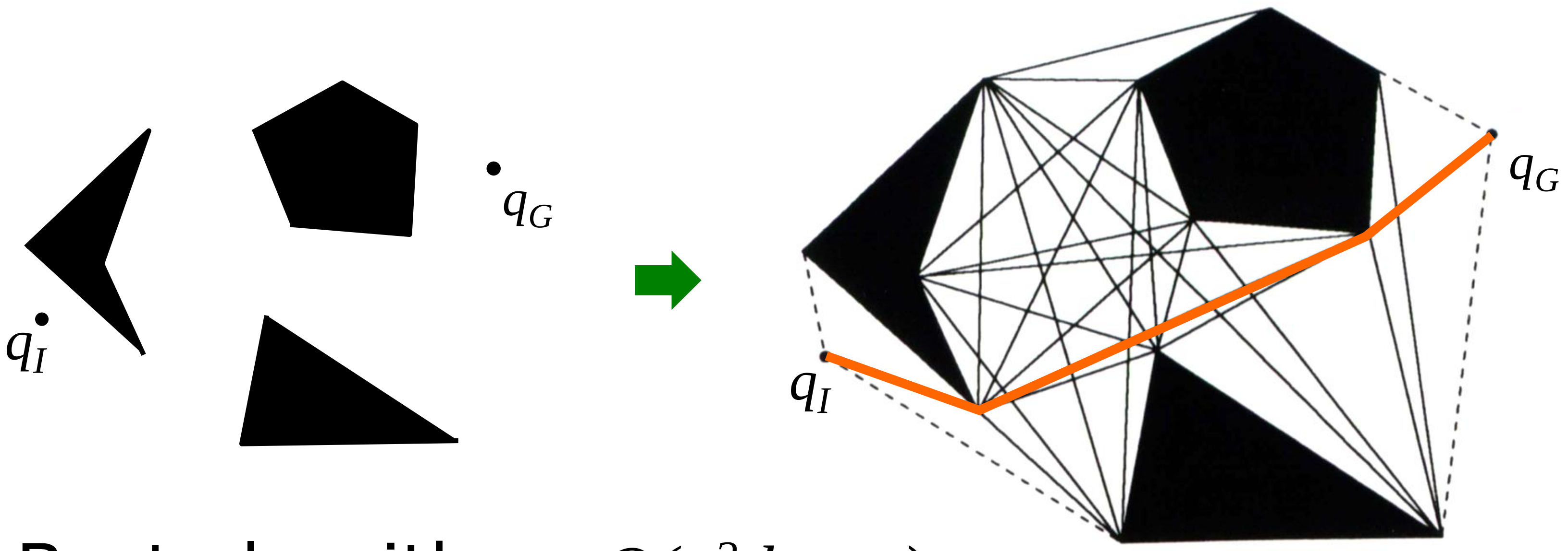
- Without loss of generality, we will consider a problem in $\mathcal{W} = \mathbb{R}^2$ with a **point robot** that cannot rotate. In this case: $\mathcal{C} = \mathbb{R}^2$
- We further assume a **polygonal** world





Visibility Graphs

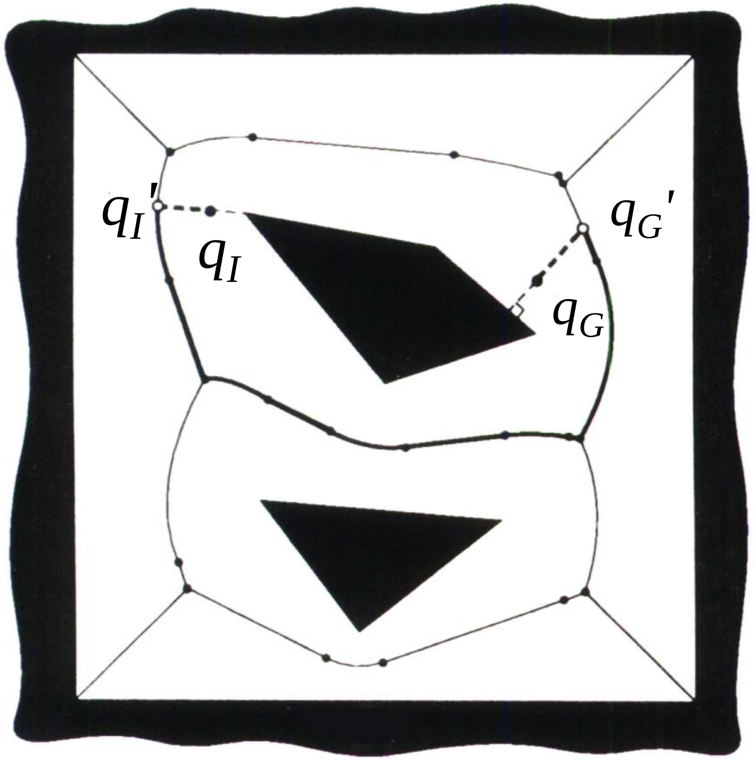
- **Idea:** construct a path as a polygonal line connecting q_I and q_G through vertices of C_{obs}
- Existence proof for such paths, **optimality**
- One of the earliest path planning methods



- Best algorithm: $O(n^2 \log n)$

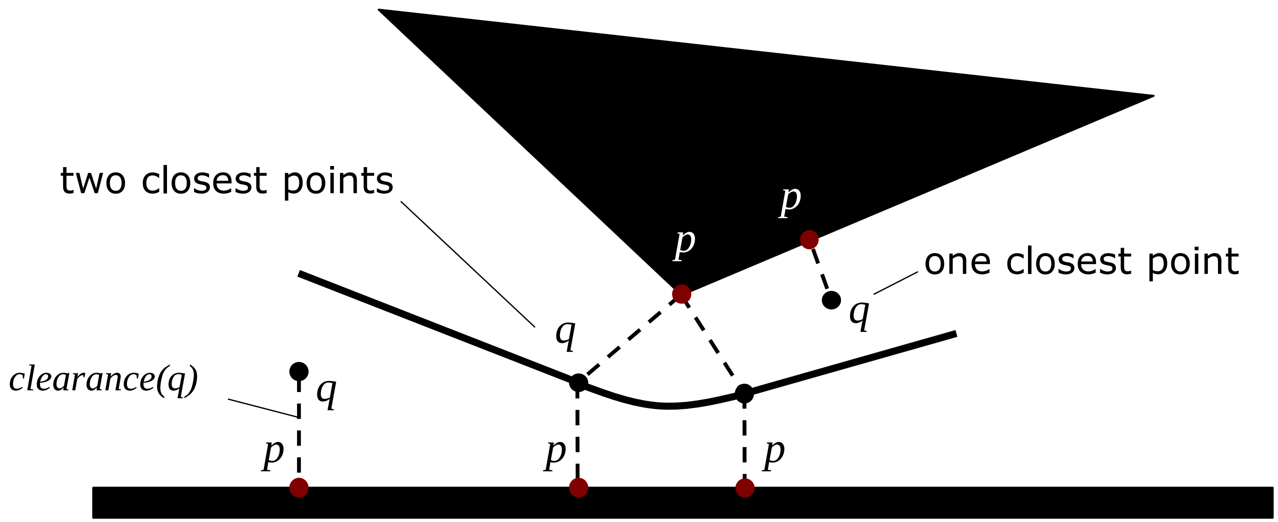
Generalized Voronoi Diagram

- **Defined** to be the set of points q whose cardinality of the set of boundary points of C_{obs} with the same distance to q is greater than 1
- Let us decipher this definition...
- **Informally:** the place with the same **maximal clearance** from all nearest obstacles



Generalized Voronoi Diagram

- **Geometrically:**



- For a polygonal C_{obs}, the Voronoi diagram consists of (n) lines and parabolic segments
- Naive algorithm: $O(n^4)$, best: $O(n \log n)$

Voronoi Diagram

- Voronoi diagrams have been well studied for (reactive) **mobile robot** path planning
- Fast methods exist to compute and update the diagram in real-time for low-dim. C's
  - **Pros:** maximize clear-ance is a good idea for an uncertain robot
  - **Cons:** unnatural at-traction to open space, suboptimal paths

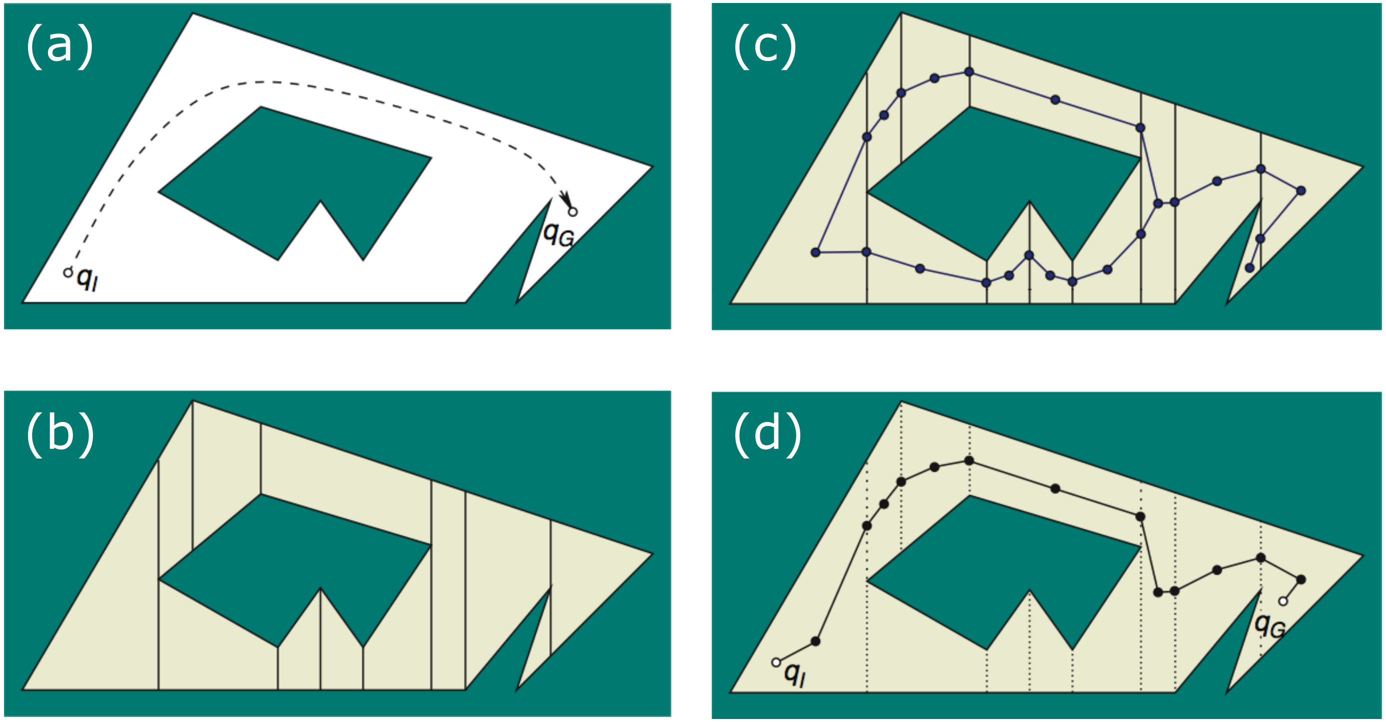
- Needs extensions

Exact Cell Decomposition

- **Idea:** decompose C_{free} into non-overlapping cells, construct connectivity graph to represent adjacencies, then search
- A popular implementation of this idea:
  1. Decompose C_{free} into **trapezoids** with vertical side segments by shooting rays upward and downward from each polygon vertex
  2. Place one **vertex** in the interior of every **trapezoid**, pick e.g. the centroid
  3. Place one **vertex** in every vertical **segment**
  4. Connect the vertices

Exact Cell Decomposition

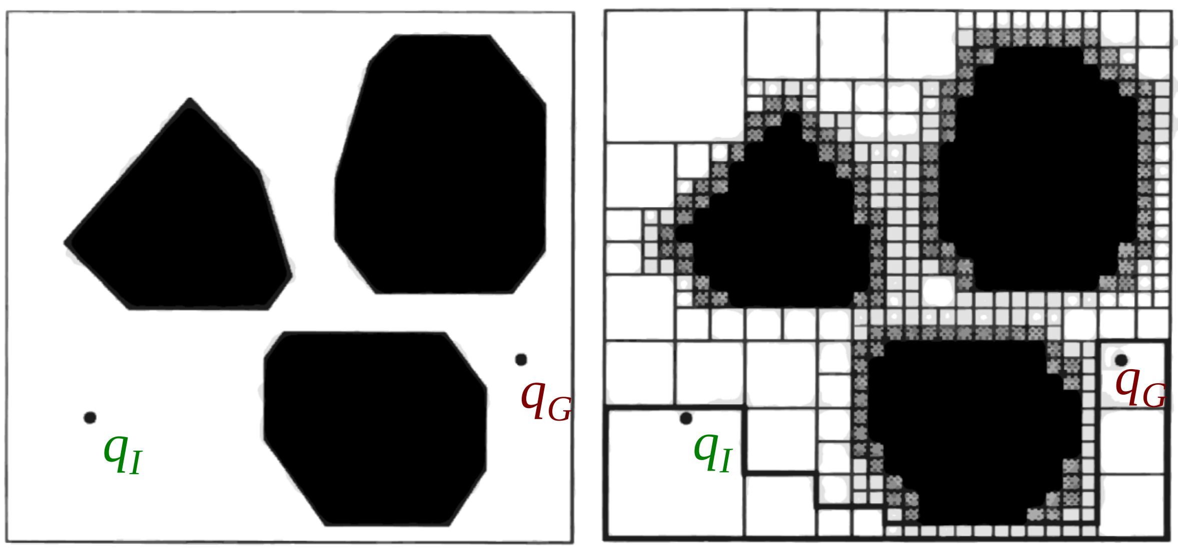
- Trapezoidal decomposition ($\mathcal{C} = \mathbb{R}^3$ max)



- Best known algorithm: $O(n \log n)$ where n is the number of vertices of C_{obs}

Approximate Cell Decomposition

- Exact decomposition methods can be involved and inefficient for complex problems
- Approximate decomposition uses cells with the **same simple predefined shape**



Quadtree decomposition

Approximate Cell Decomposition

- Exact decomposition methods can be involved and inefficient for complex problems
- Approximate decomposition uses cells with the **same simple predefined shape**
- **Pros:**
  - Iterating the **same** simple computations
  - Numerically more **stable**
  - **Simpler** to implement
  - Can be made **complete**

Combinatorial Planning

Wrap Up

- Combinatorial planning techniques are **elegant** and **complete** (they find a solution if it exists, report failure otherwise)
- But: become **quickly intractable** when C-space dimensionality increases (or n resp.)
- Combinatorial **explosion** in terms of **facets** to represent $\mathcal{A}$, $\mathcal{O}$, and $\mathcal{C}_{obs}$, especially when rotations bring in non-linearities and make C a nontrivial manifold

➡ Use **sampling-based planning**
Weaker guarantees but more efficient

Sampling-Based Planning

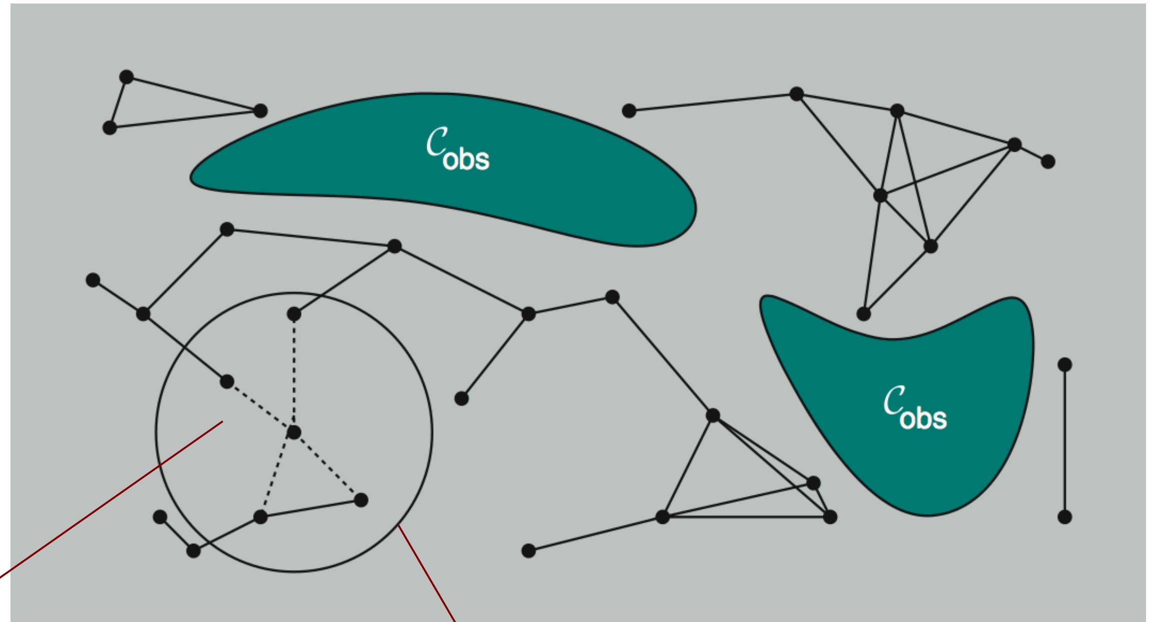
- Abandon the concept of explicitly characterizing C_{free} and C_{obs} and leave the algorithm **in the dark** when exploring C_{free}
- The only light is provided by a **collision-detection algorithm**, that probes C to see whether some configuration lies in C_{free}
- We will have a look at
  - **Probabilistic road maps** (PRM)
    [Kavraki et al., 92]
  - **Rapidly exploring random trees** (RRT)
    [Lavalle and Kuffner, 99]

Probabilistic Road Maps

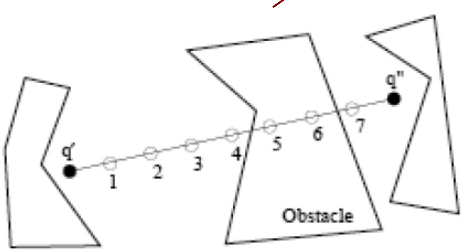
- **Idea:** Take random samples from C, declare them as vertices if in C_{free}, try to connect nearby vertices with local planner
- The **local planner** checks if line-of-sight is collision-free (powerful or simple methods)
- Options for *nearby:* **k-nearest neighbors** or all neighbors within **specified radius**
- Configurations and connections are added to graph until roadmap is **dense enough**

Probabilistic Road Maps

- Example



specified radius



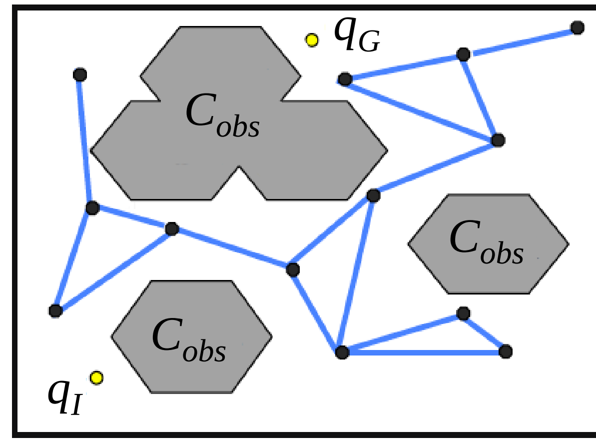
Example local planner

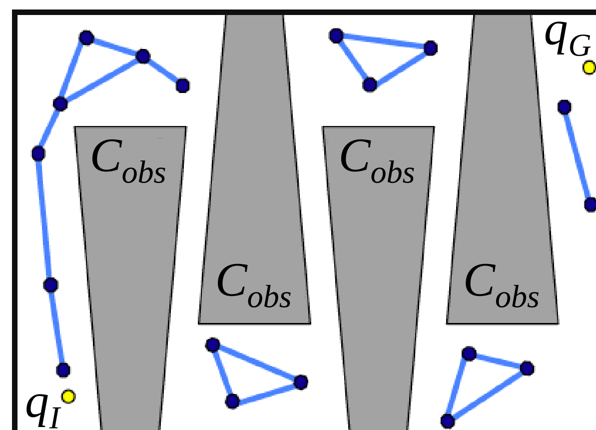
What means "nearby" on a manifold?
Defining a good metric on C is crucial

Probabilistic Road Maps

Good and bad news:

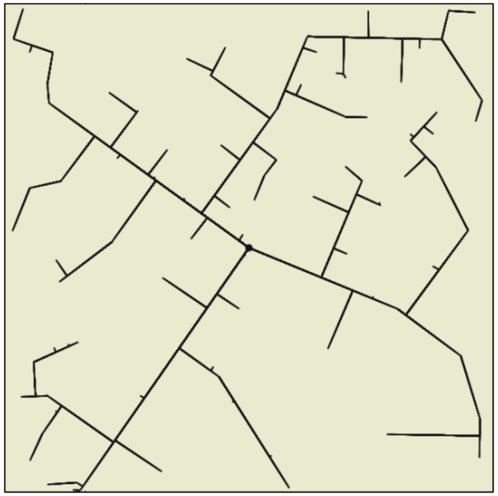
- **Pros:**
  - *Probabilistically complete*
  - Do not construct C-space
  - Apply easily to high-dim. C's
  - PRMs have solved previously unsolved problems
- **Cons:**
  - Do not work well for some problems, narrow passages
  - Not optimal, not complete





Rapidly Exploring Random Trees

- **Idea:** aggressively probe and explore the C-space by **expanding incrementally** from an initial configuration q_0
- The explored territory is marked by a **tree rooted at** $\boldsymbol{q_0}$



45 iterations

2345 iterations

From Road Maps to Paths

- All methods discussed so far **construct a road map** (without considering the query pair q_I and q_G)

- Once the investment is made, the **same road map** can be reused for **all** queries (provided world and robot do not change)

  1. **Find** the cell/vertex that contain/is close to q_I and q_G (not needed for visibility graphs)
  2. **Connect** q_I and q_G to the road map
  3. **Search** the road map for a path from q_I to q_G

Sampling-Based Planning

Wrap Up

- Sampling-based planners are **more efficient** in most **practical problems** but offer weaker guarantees
- They are **probabilistically complete**: the probability tends to 1 that a solution is found if one exists (otherwise it may still run forever)
- Performance degrades in problems with **narrow passages**. Subject of active research
- Widely used. Problems with high-dimensional and complex C-spaces are still computationally hard

Potential Field Methods

- All techniques discussed so far aim at capturing the connectivity of C_{free} into a graph

- **Potential Field methods** follow a different idea:

  The robot, represented as a point in C, is modeled as a **particle** under the influence of a **artificial potential field** $\mathbf{U}$

  $\mathbf{U}$ superimposes

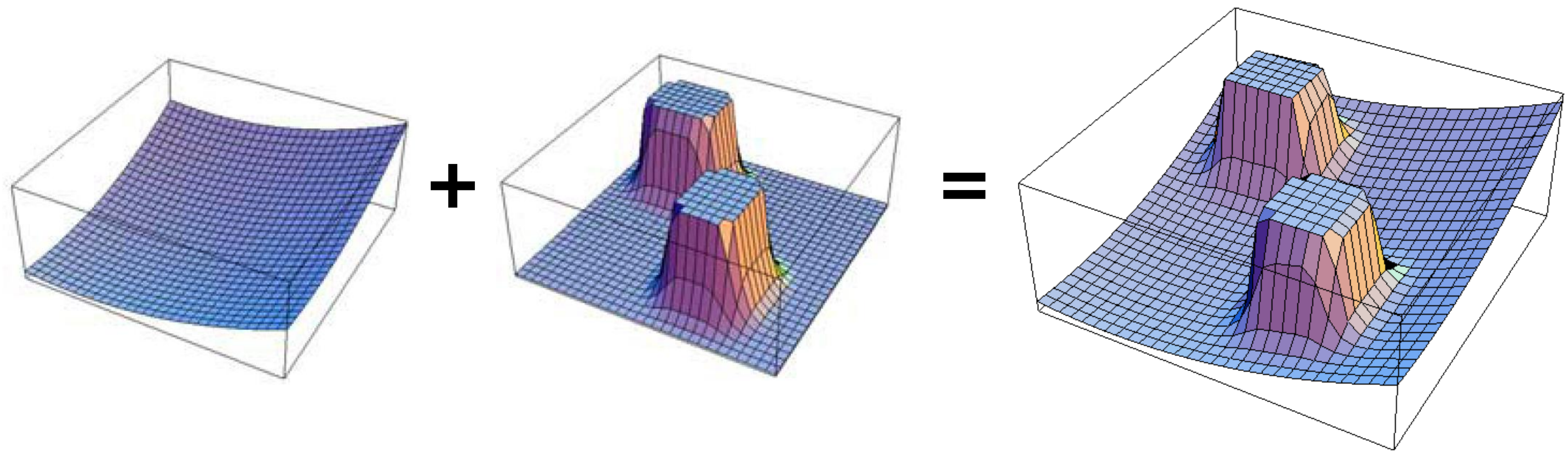
  - **Repulsive forces** from obstacles
  - **Attractive force** from goal

Potential Field Methods

- Potential function

$$\mathbf{U}(q) = \mathbf{U}_{att}(q) + \mathbf{U}_{rep}(q)$$

$$\vec{F}(q) = -\vec{\nabla}\mathbf{U}(q)$$



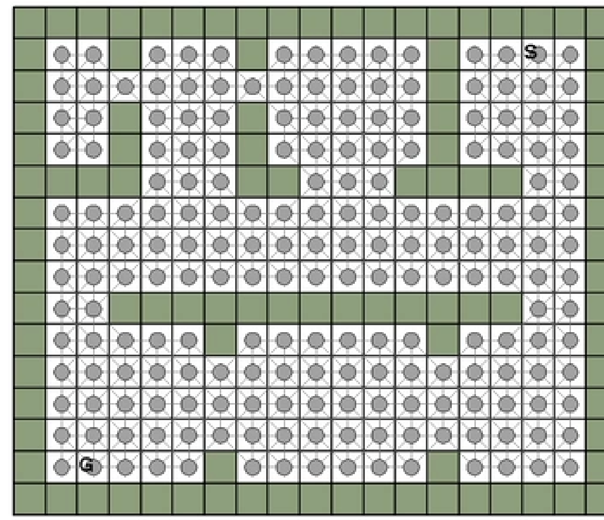
- Simply perform **gradient descent**
- C-pace typically discretized in a grid

Potential Field Methods

- Main problems: robot gets stuck in **local minima**
- Way out: Construct local-minima-free **navigation function** ("NF1"), then do gradient descent (e.g. bushfire from goal)
- The gradient of the potential function defines a **vector field** (similar to a policy) that can be used as **feedback control strategy**, relevant for an uncertain robot
- However, potential fields need to represent C_{free} **explicitely**. This can be too costly.

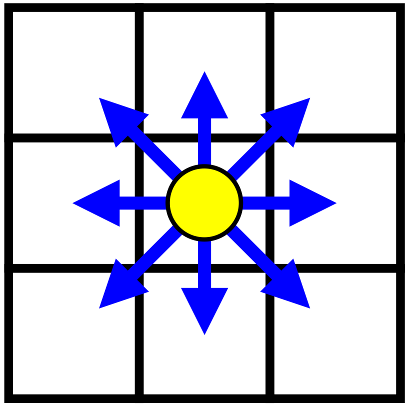
Robot Motion Planning

- Given a road map, let's do **search**!



A* Search

- **A*** is one of the most widely-known informed search algorithms with many applications in robotics
- *Where are we?*
A* is an instance of an **informed algorithm** for the general problem of **search**
- In robotics: planning on a 2D occupancy grid map is a common approach



Search

The problem of **search:** finding a sequence of actions (a *path*) that leads to desirable states (a *goal*)

- **Uninformed search:** besides the problem definition, no further information about the domain ("blind search")
- The only thing one can do is to expand nodes differently
- Example algorithms: breadth-first, uniform-cost, depth-first, bidirectional, etc.

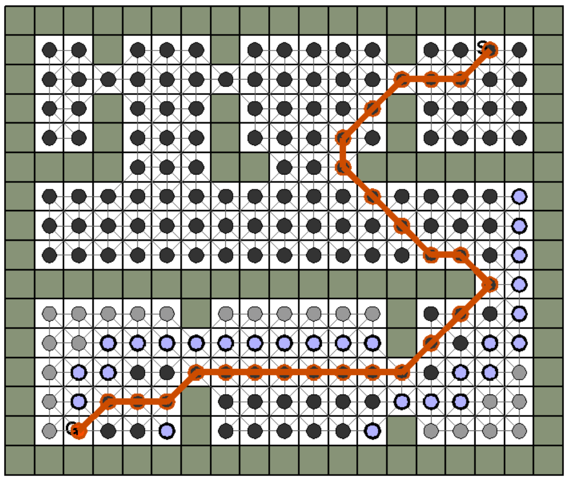
Search

The problem of **search:** finding a sequence of actions (a *path*) that leads to desirable states (a *goal*)

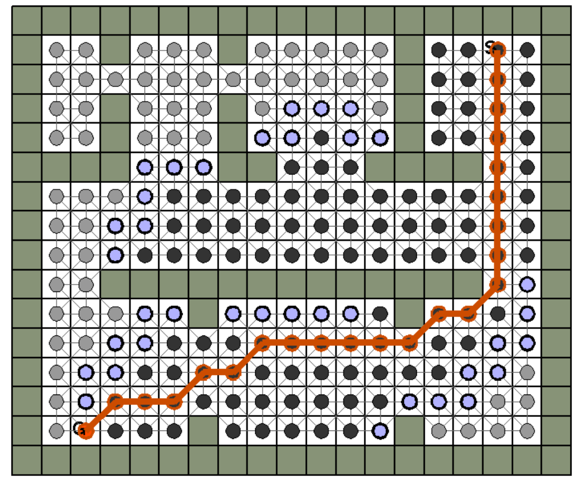
- **Informed search:** further information about the domain through **heuristics**
- Capability to say that a node is "more promising" than another node
- Example algorithms: greedy best-first search, **A***, many variants of A*, D*, etc.

Any-Angle A* Examples

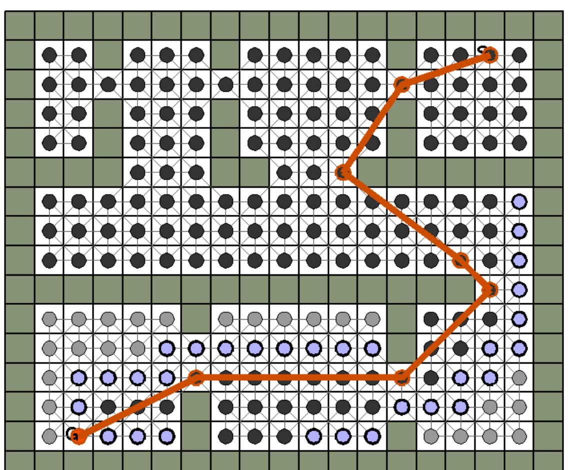
- A* vs. Theta* (*len*: path length, *nhead* = # heading changes)



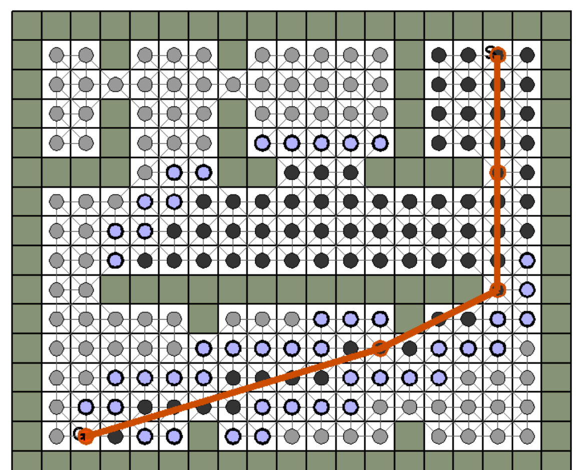
len: 30.0
nhead: 11



len: 24.1
nhead: 9



len: 28.9
nhead: 5



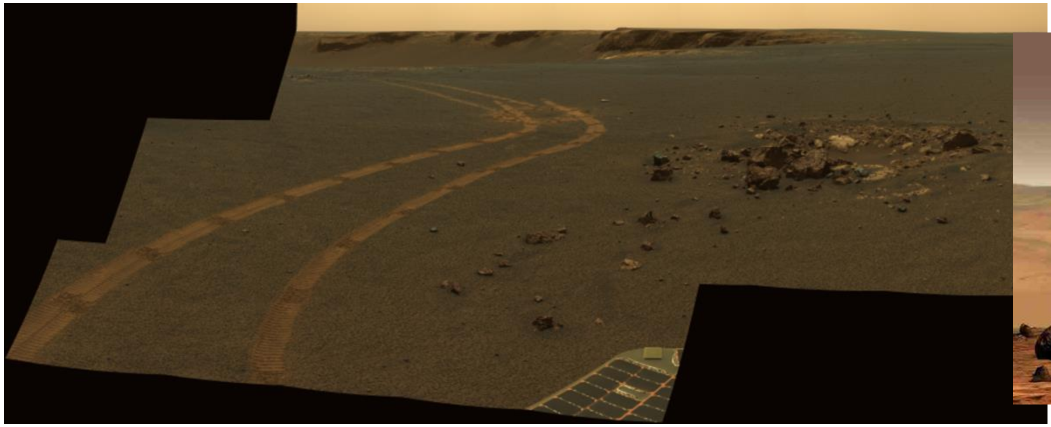
len: 22.9
nhead: 2

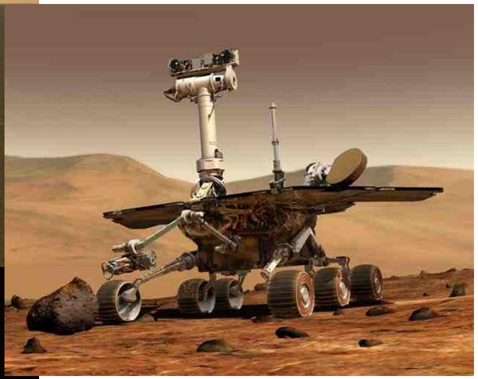
D* Search

- **Problem:** In unknown, partially known or dynamic environments, the planned path may be blocked and we need to **replan**
- Can this be done efficiently, avoiding to replan the **entire path?**
- **Idea:** Incrementally repair path keeping its modifications local around robot pose
- Several approaches implement this idea:
  - **D*** (Dynamic A*) *[Stentz, ICRA'94, IJCAI'95]*
  - **D* Lite** *[Koenig and Likhachev, AAAI'02]*
  - **Field D*** *[Ferguson and Stentz, JFR'06]*

D* Family

- **D* Lite** produces the same paths than D* but is **simpler** and more **efficient**
- D*/D* Lite are **widely used**
- **Field D*** was running on Mars rovers Spirit and Opportunity (retrofitted in yr 3)





Tracks left by a drive executed with Field D*

Still in Dynamic Environments...

- Do we really need to replan the entire path for **each obstacle** on the way?
- What if the robot has to react **quickly** to unforeseen, fast moving obstacles?
  - Even D* Lite can be too slow in such a situation
- Accounting for the **robot shape** (it's not a point)
- Accounting for **kinematic** and **dynamic** vehicle **constraints**, e.g.
  - Decceleration limits,
  - Steering angle limits, etc.

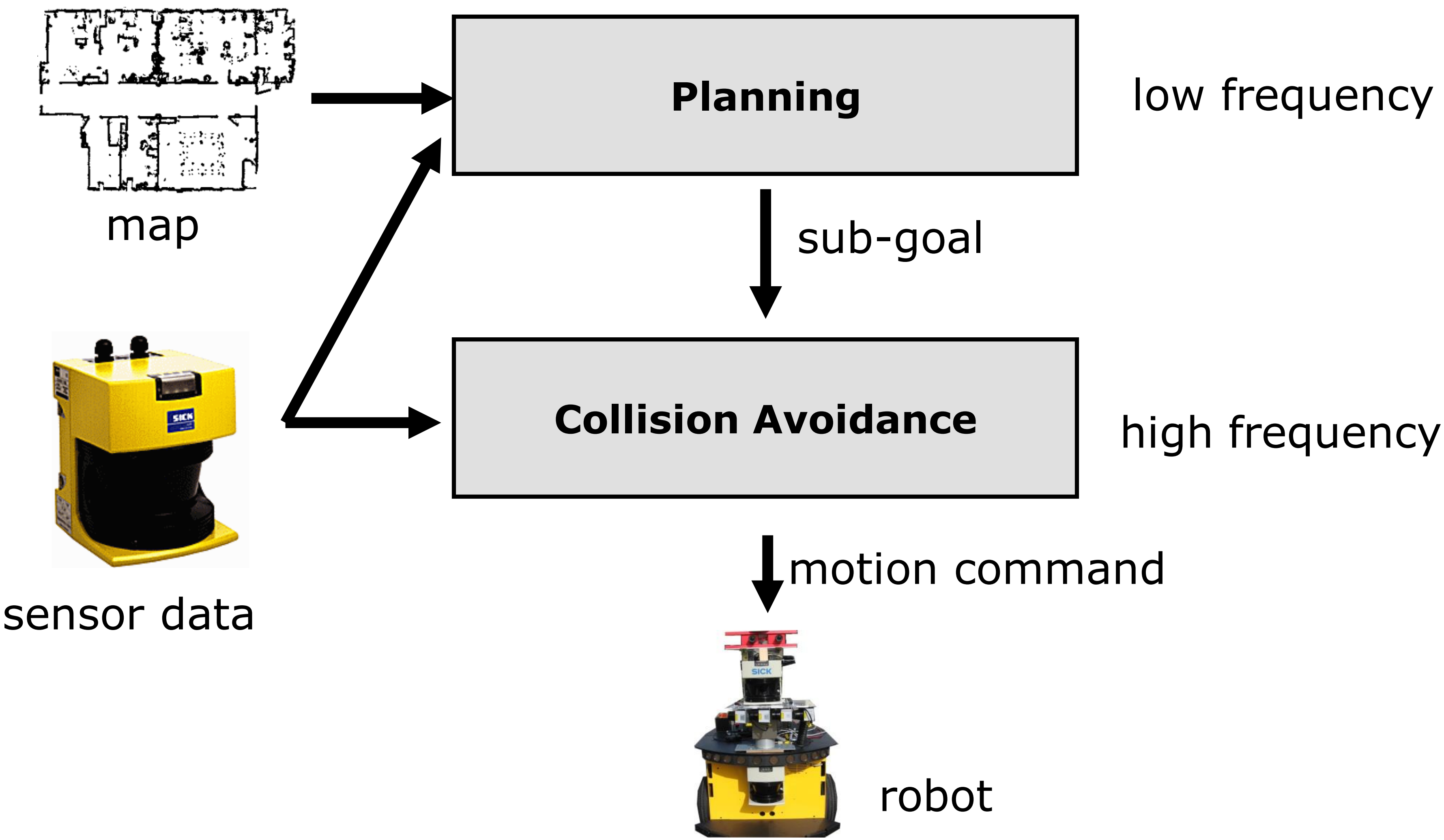
Collision Avoidance

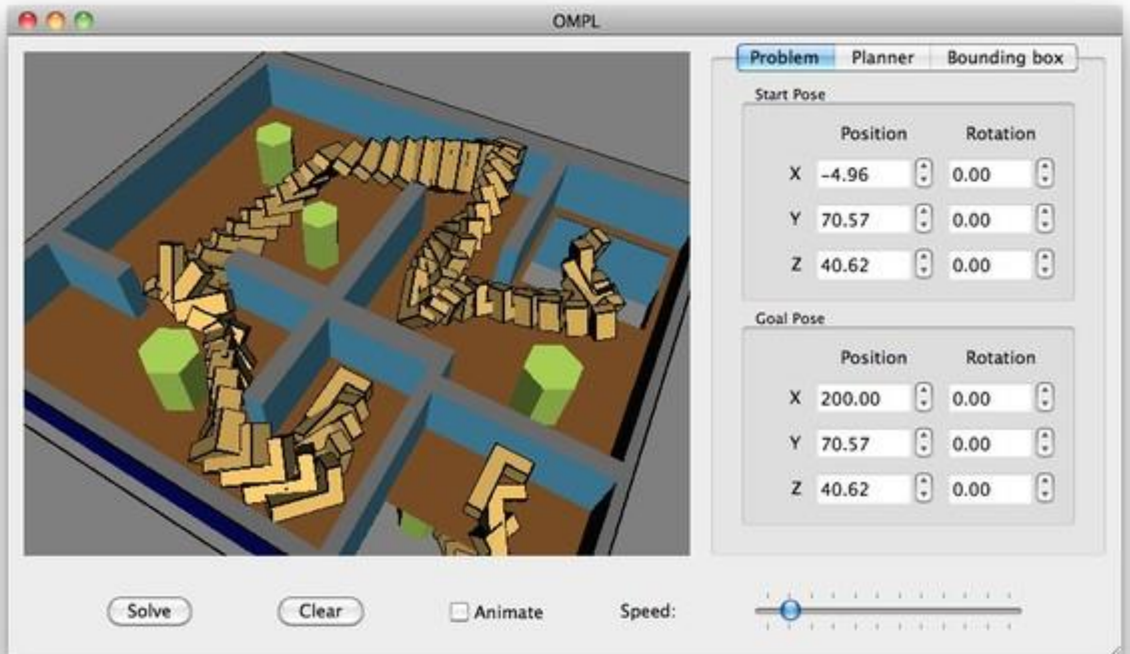
- This can be handled by techniques called **collision avoidance** (obstacle avoidance)
- A well researched subject, different **approaches** exist:
  - Dynamic Window Approaches
    [Simmons, 96], [Fox et al., 97], [Brock & Khatib, 99]
  - Nearness Diagram Navigation
    [Minguez et al., 2001, 2002]
  - Vector-Field-Histogram+
    [Ulrich & Borenstein, 98]
  - Extended Potential Fields
    [Khatib & Chatila, 95]

Collision Avoidance

- Integration into general motion planning?
- It is common to subdivide the problem into a global and local planning task:
  - An approximate **global planner** computes paths ignoring the kinematic and dynamic vehicle constraints
  - An accurate **local planner** accounts for the constraints and generates (sets of) feasible local trajectories ("collision avoidance")
- What do we loose? What do we win?

Two-layered Architecture




OMPL
Problem
Planner
Bounding box
Start Pose
Position
Rotation
X -4.96 0.00
Y 70.57 0.00
Z 40.62 0.00
Goal Pose
Position
Rotation
X 200.00 0.00
Y 70.57 0.00
Z 40.62 0.00
Solve
Clear
Animate
Speed:

ompl