Mobile Robot Mapping

Wolfram Burgard

With contributions by: Maren Bennewitz, Dieter Fox, Giorgio Grisetti, Slawomir Grzonka, Dirk Haehnel, Armin Hornung, Kristian Kersting, Rainer Kuemmerle, Oscar Martinez Mozos, Patrick Pfaff, Christian Plagemann, Axel Rottmann, Cyrill Stachniss, Hauke Strasdat, Rudolph Triebel, Sebastian Thrun …

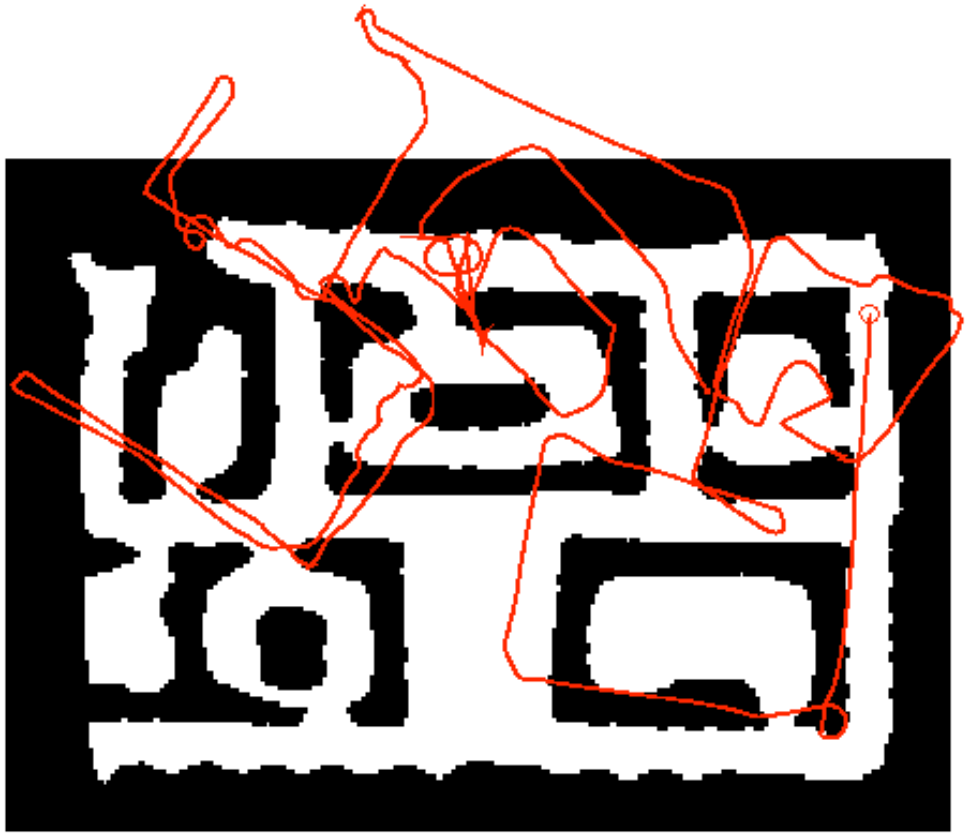
Why Mapping?

- 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.

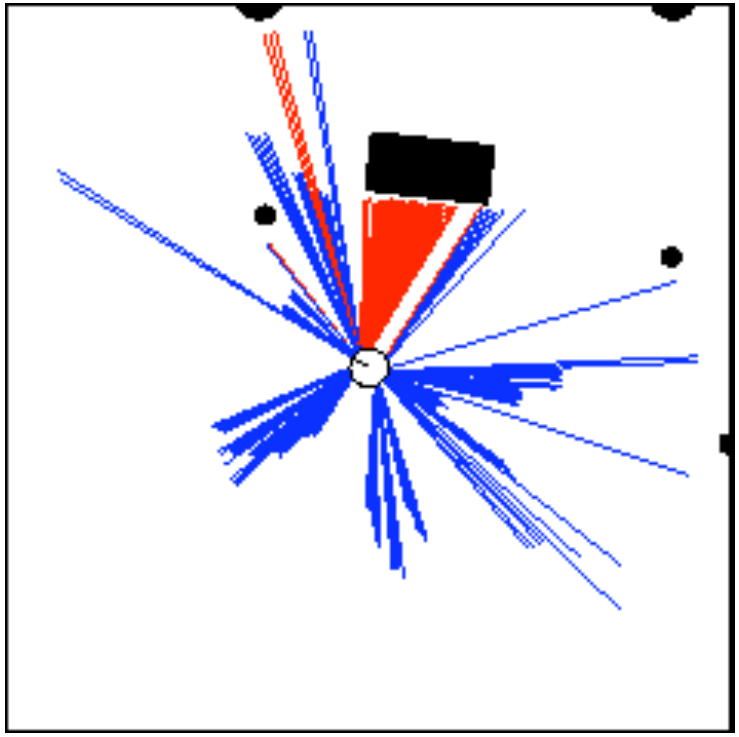
Topics Covered

- Probabilistic reasoning
- Occupancy grid mapping
- Reflection maps
- 3d mapping
- SLAM
- Learning for improving SLAM
- Interpretation
- …

Nature of Data



Odometry Data



Range Data

Probabilistic Robotics

- perception = state estimation

- action = utility optimization

Probabilistic Robotics

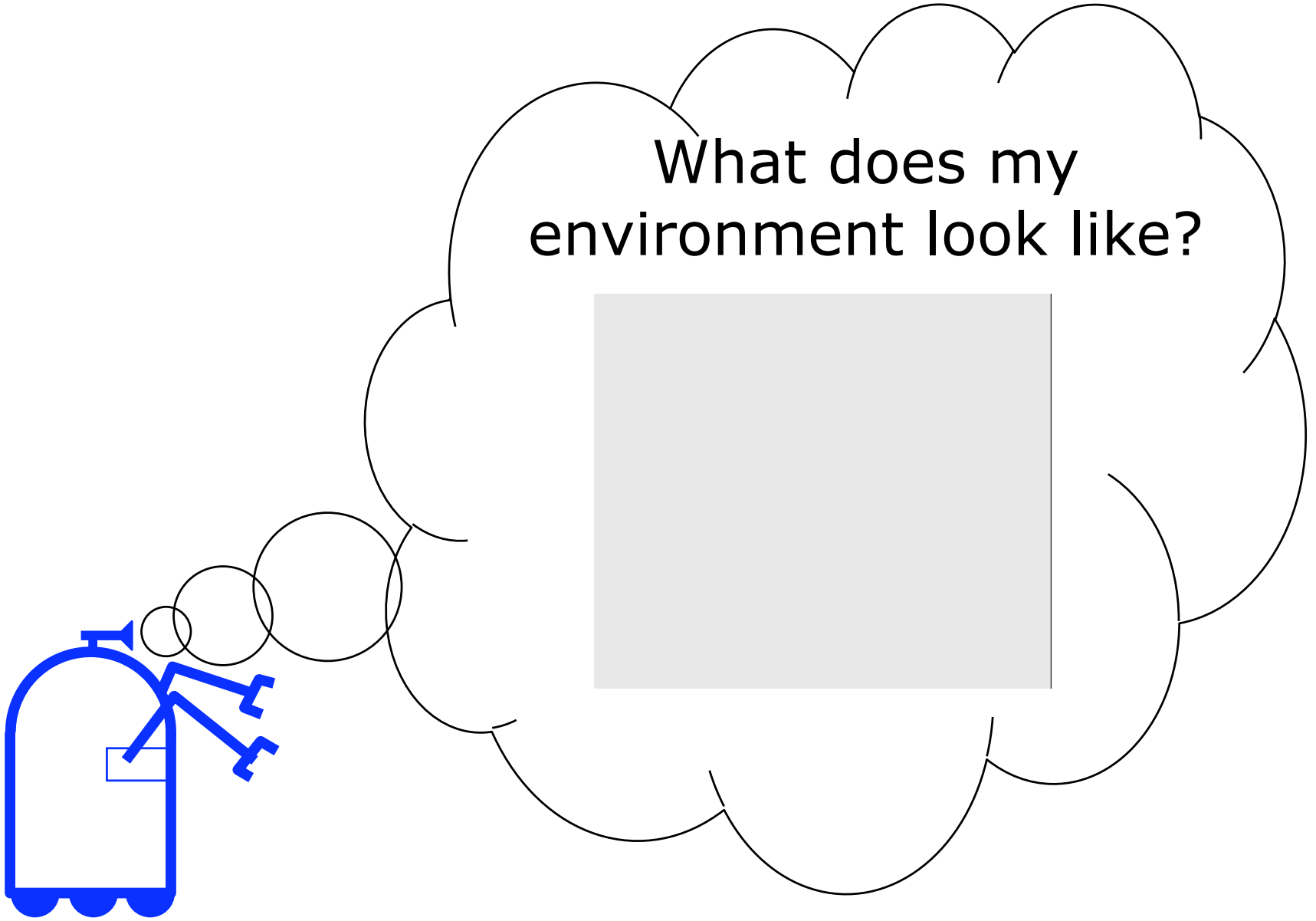
- perception = state estimation

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

- action = utility optimization

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')]$$

The Robot Mapping Problem



The Robot Mapping Problem

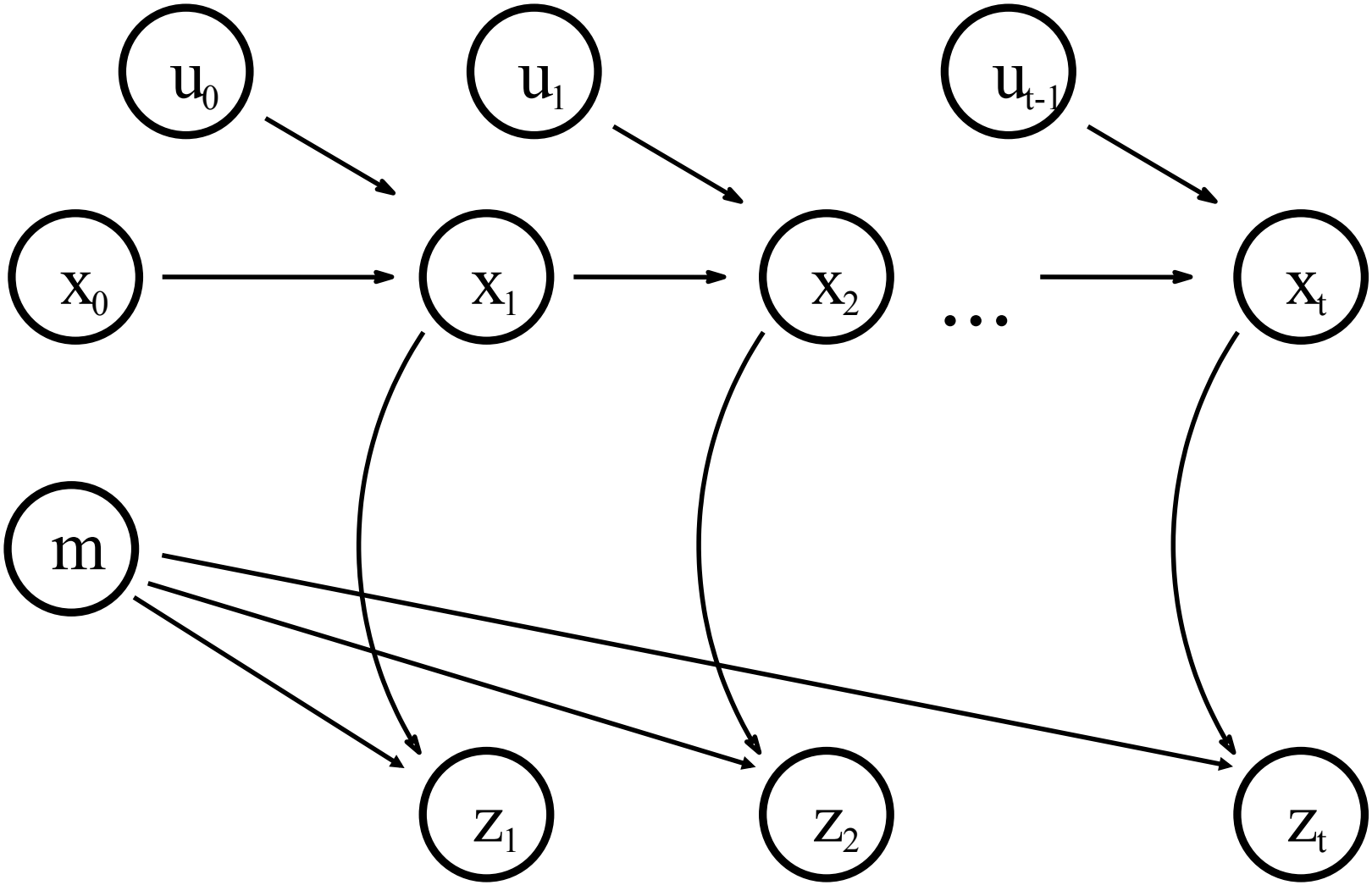
Formally, mapping involves, given the sensor data,

$$d = \{z_{1,\ldots,t}, u_{0,\ldots,t-1}\}$$

to calculate the most likely map

$$m^* = \operatorname*{argmax}_m P(m \mid z_{1,\ldots,t}, u_{0,\ldots,t-1})$$

A Graphical Model for SLAM

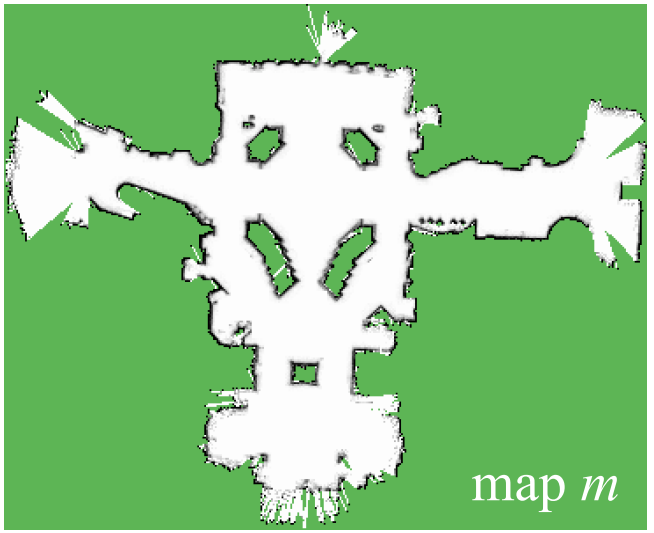


Probabilistic Formulation of SLAM

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

n=axb dimensions

three dimensions



Several Aspects

- Mapping with known poses
- Localization
- Simultaneous localization and mapping (SLAM)
- Simplifications
- Dynamic environments
- Learning and SLAM

Bayes Filters

z = observation
u = action
x = state

$$Bel(x_t) = P(x_t \mid z_{1,\ldots,t}, u_{0,\ldots,t-1})$$

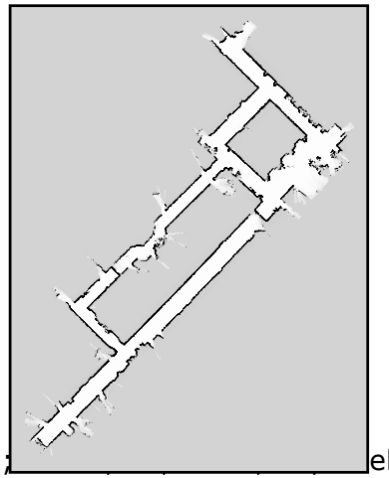
SLAM: Mapping as a Chicken and Egg Problem?

- It appeared easy to estimate the pose of a robot 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.

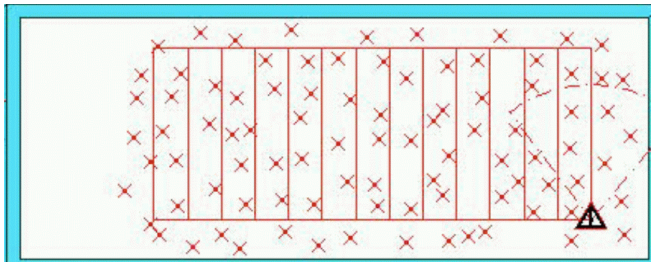
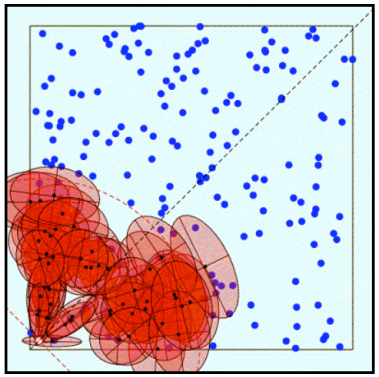
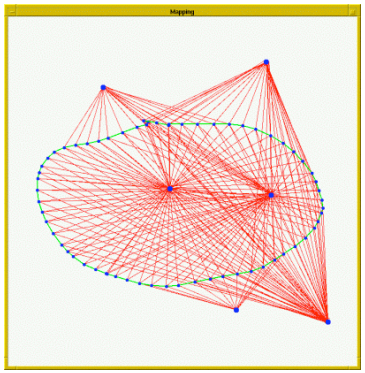
Types of SLAM-Problems

- Grid maps or scans


mann, 98: T

Gutmann, 00;

ehnel, 01;…]

- Landmark-based

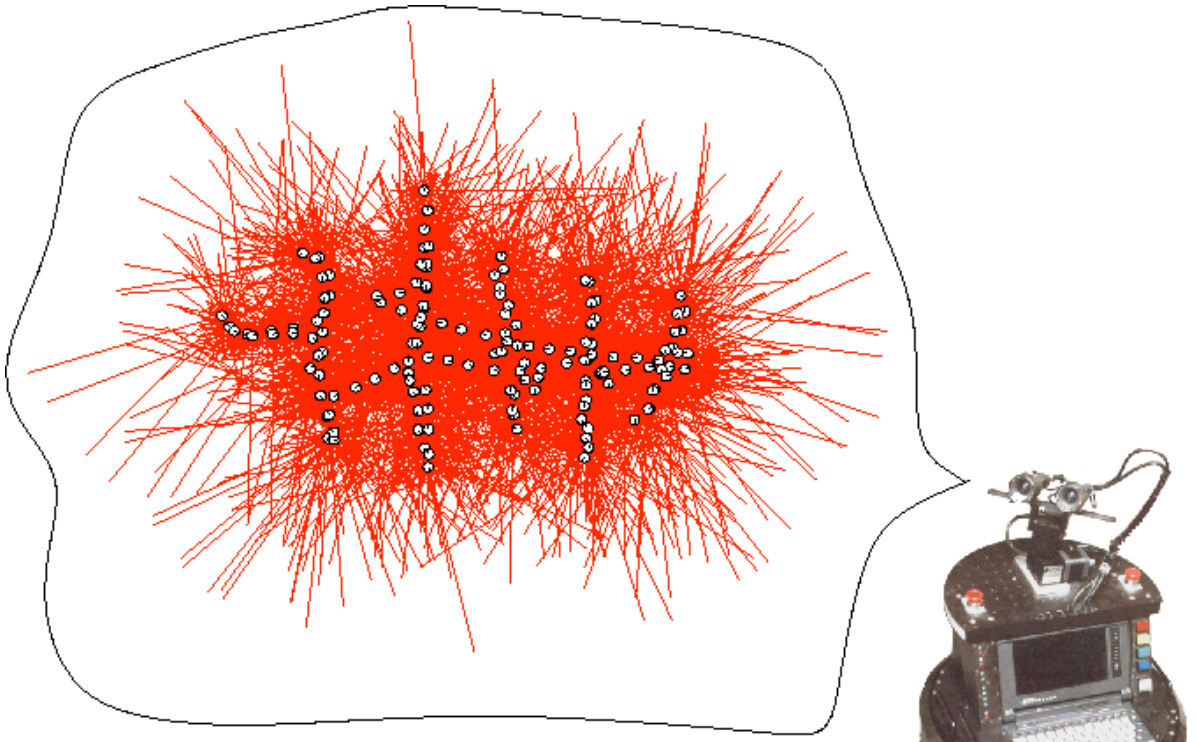
[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

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.

Mapping with Known Poses



What does the environment look like?

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 $m_t^{[x,y]}$ is independent

$$Bel(m_t) = P(m_t \mid x_{1,\ldots,t}, z_{1,\ldots,t}) = \prod_{x,y} Bel(m_t^{[x,y]})$$

  - Robot positions are known!

Updating Occupancy Grid Maps

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

$$Bel(m_t^{[x,y]}) \;=\; \eta P(z_t \mid m_t^{[x,y]}) \int P(m_t^{[x,y]} \mid m_{t-1}^{[x,y]}) Bel(m_{t-1}^{[x,y]}) dm_{t-1}^{[x,y]}$$

- Additional assumption: Map is static.

$$Bel(m_t^{[x,y]}) \;=\; \eta P(z_t \mid m_{t-1}^{[x,y]}) Bel(m_{t-1}^{[x,y]})$$

Updating Occupancy Grid Maps

- Update the map cells using the inverse sensor model

$$Bel(m_t^{[x,y]}) \;=\; 1 - \left(1 + \frac{P(m_t^{[x,y]} \mid z_t, x_t)}{1 - P(m_t^{[x,y]} \mid z_t, x_t)} \frac{1 - P(m_t^{[x,y]})}{P(m_t^{[x,y]})} \frac{Bel(m_{t-1}^{[x,y]})}{1 - Bel(m_{t-1}^{[x,y]})}\right)^{-1}$$

- Or use the log-odds representation

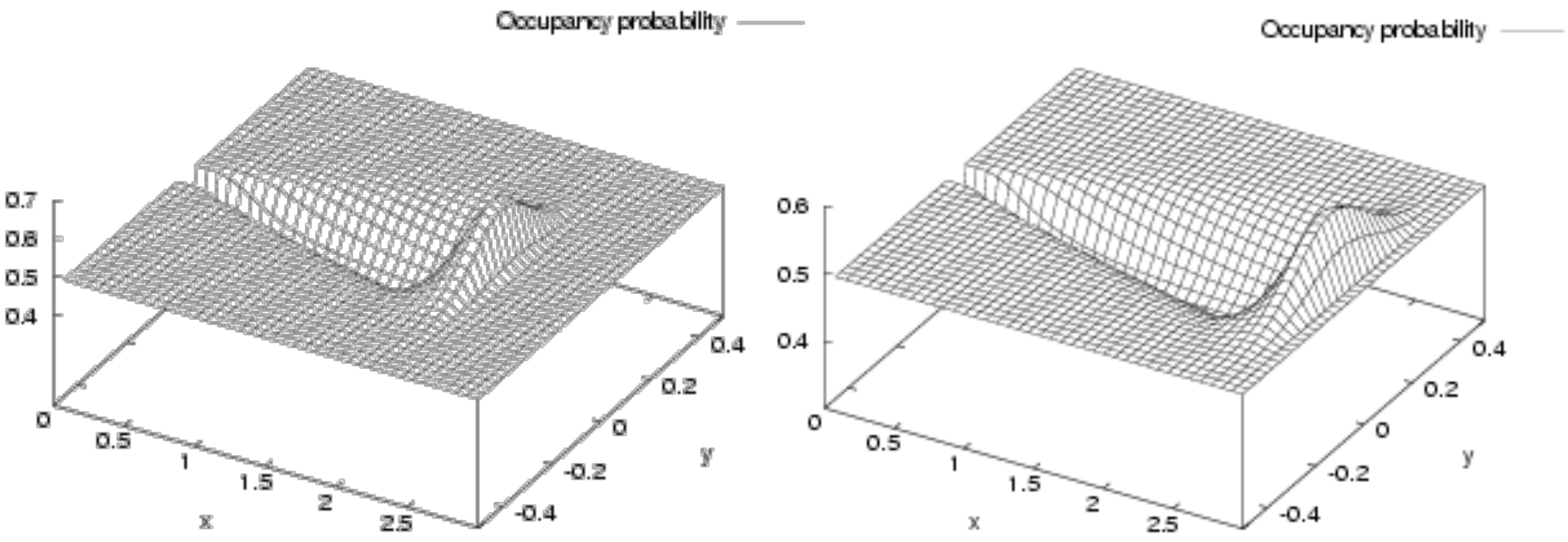
$$\bar{B}(m_t^{[x,y]}) \;=\; \log odds(m_t^{[x,y]} \mid z_t, x_t) - \log odds(m_t^{[x,y]}) + \bar{B}(m_{t-1}^{[x,y]})$$

- with

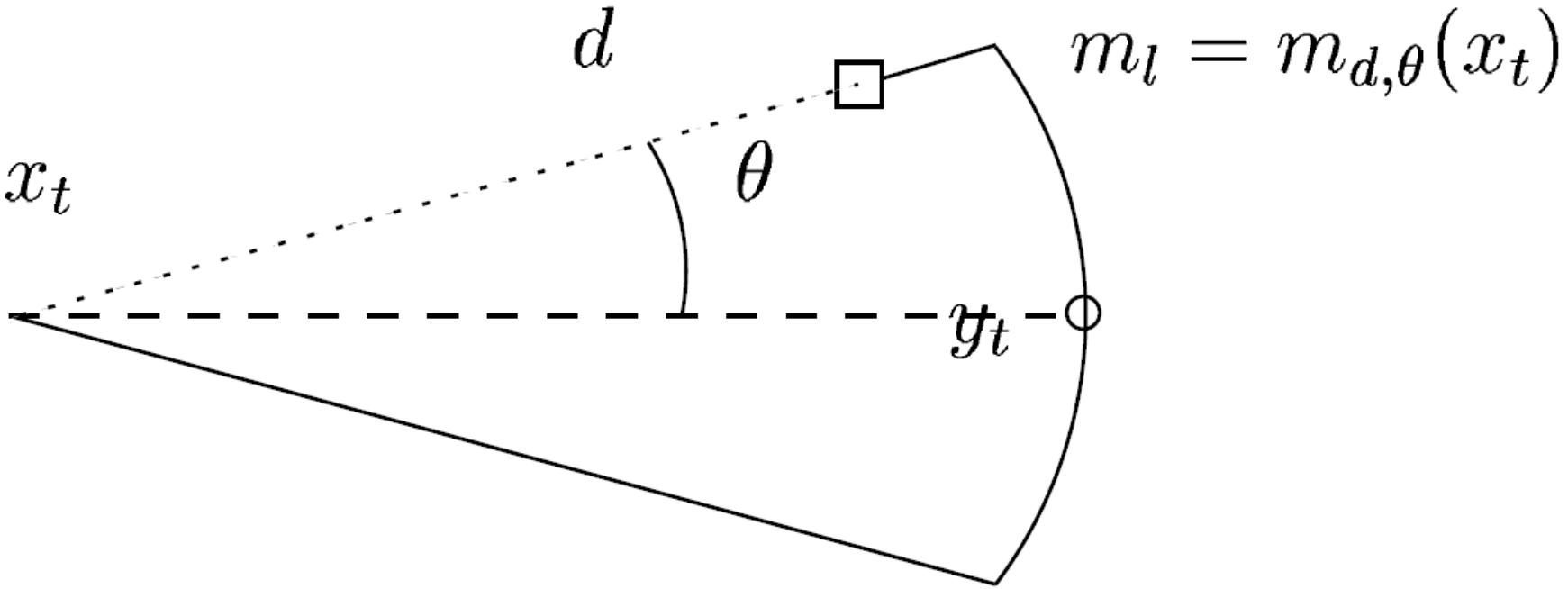
$$\bar{B}(m_t^{[x,y]}) \;:=\; \log odds(m_t^{[x,y]}) \qquad\qquad odds(x) \;:=\; \frac{P(x)}{1 - P(x)}$$

An Inverse Sensor Model for Occupancy Grid Maps

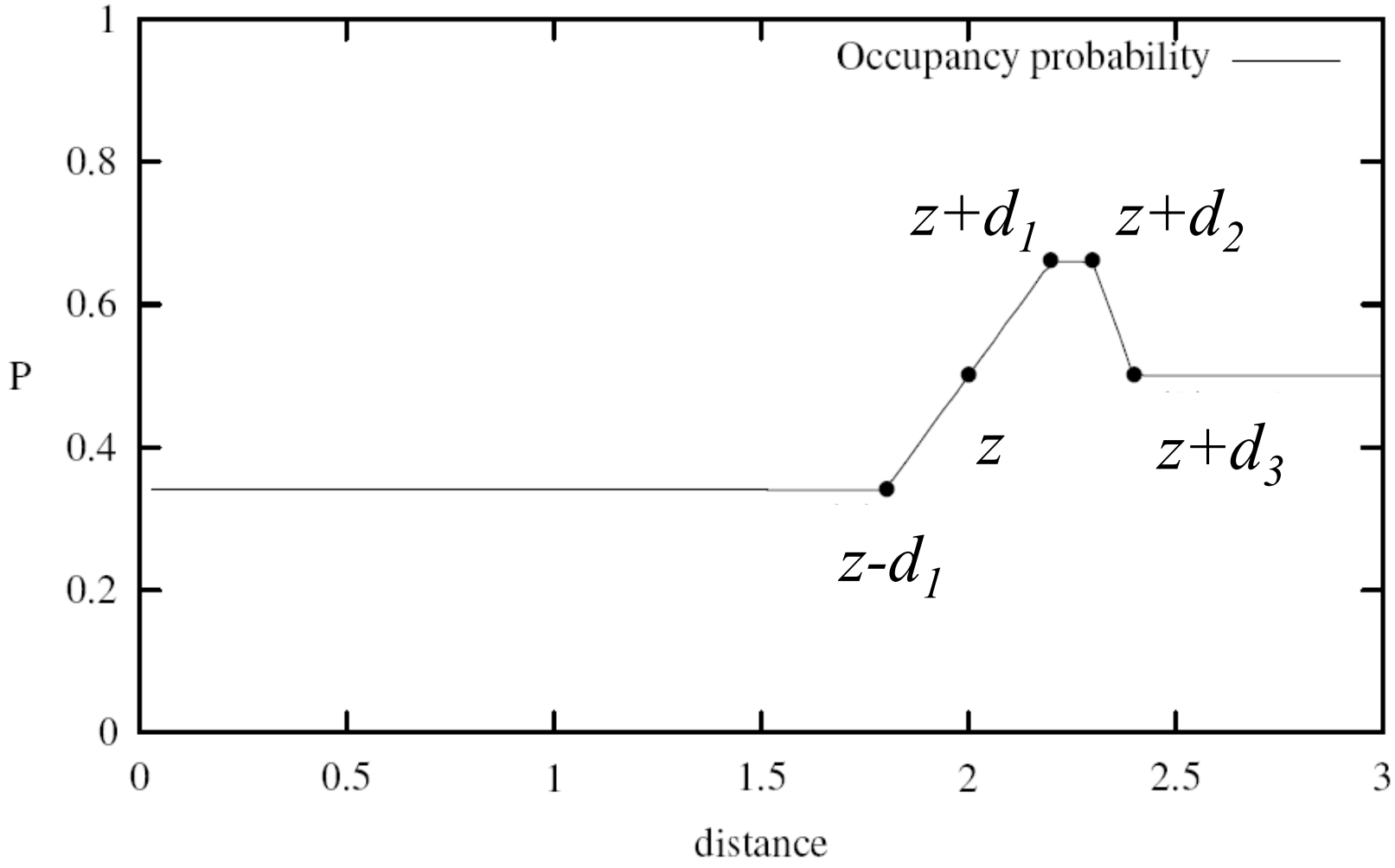
Combination of a linear function and a Gaussian:



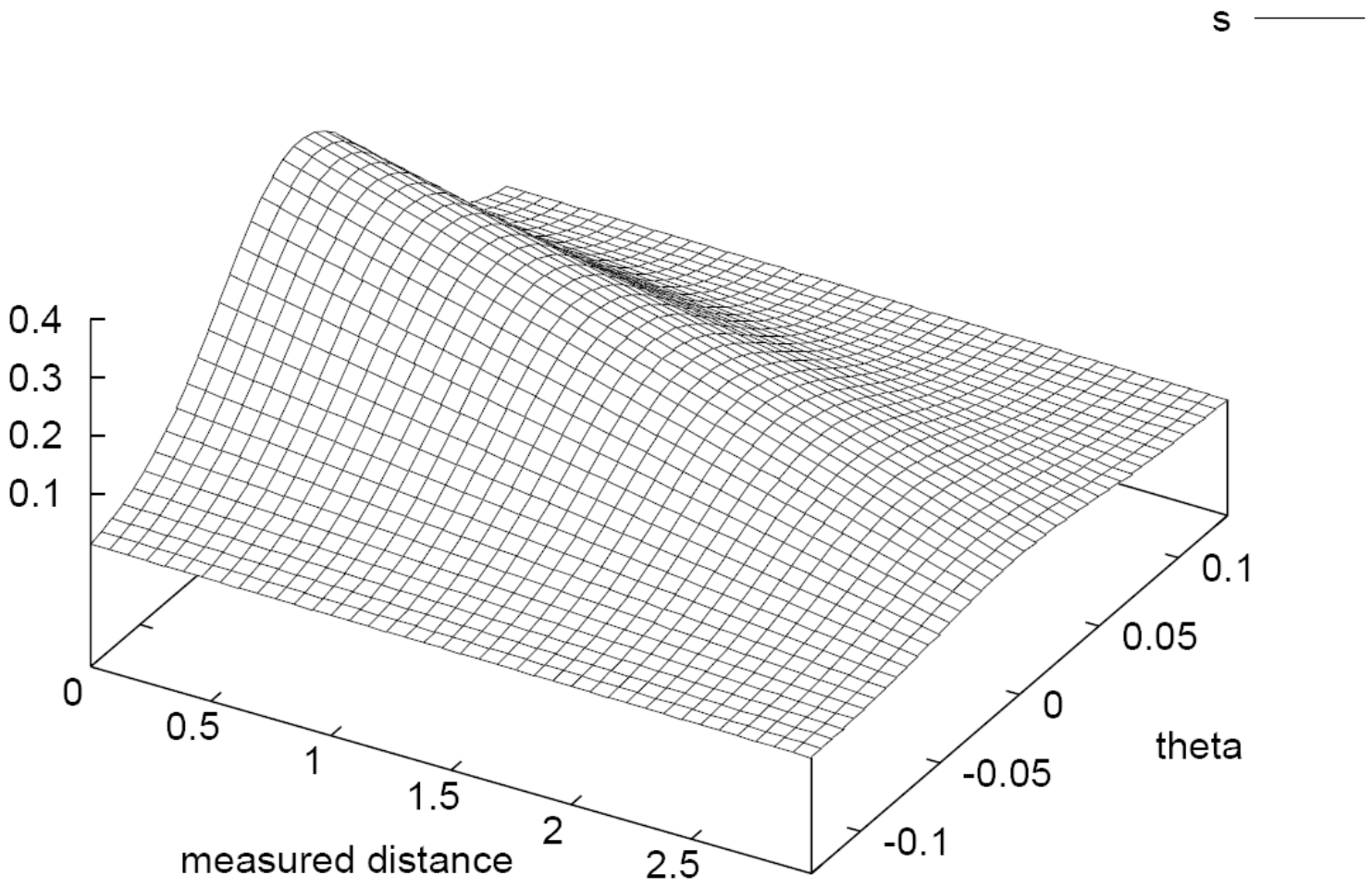
Key Parameters of the Model



Occupancy Value Depending on the Measured Distance



Deviation from the Prior Belief
(the sphere of influence of the sensors)

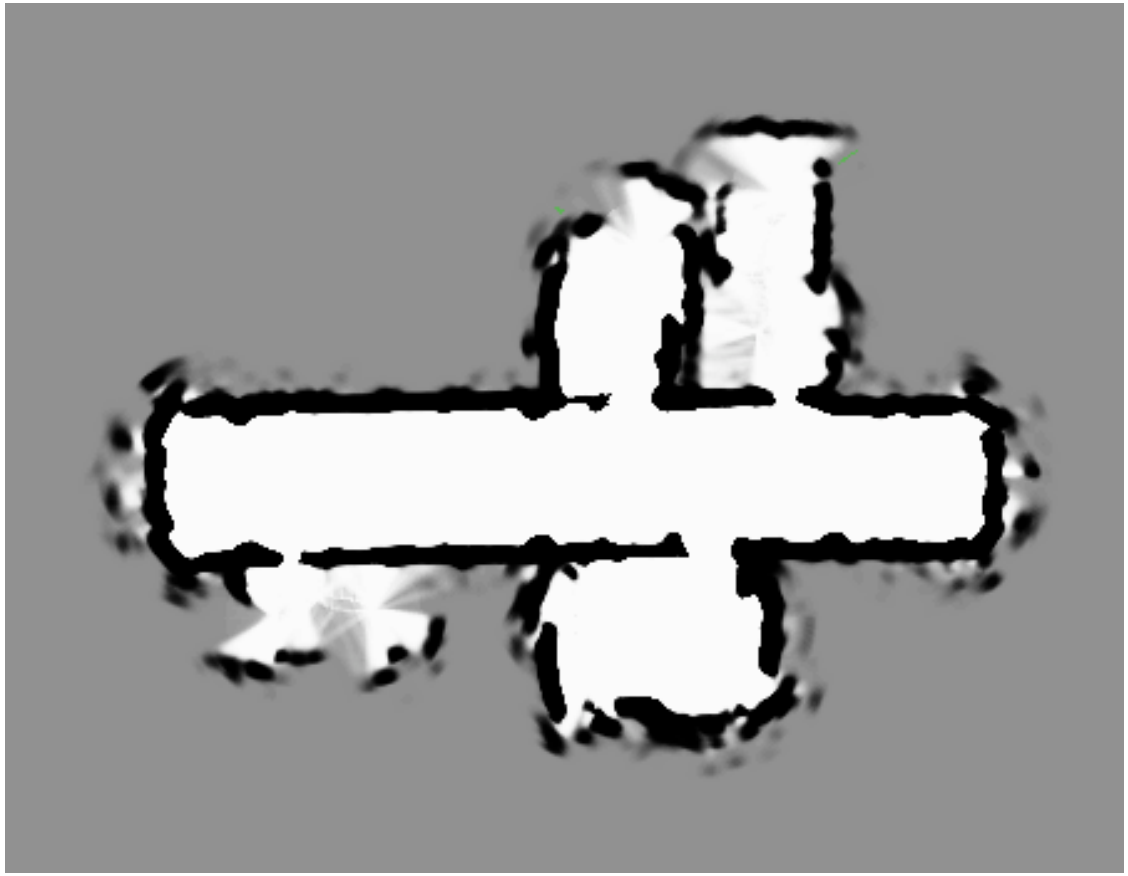


Calculating the Occupancy Probability Based on Single Observations

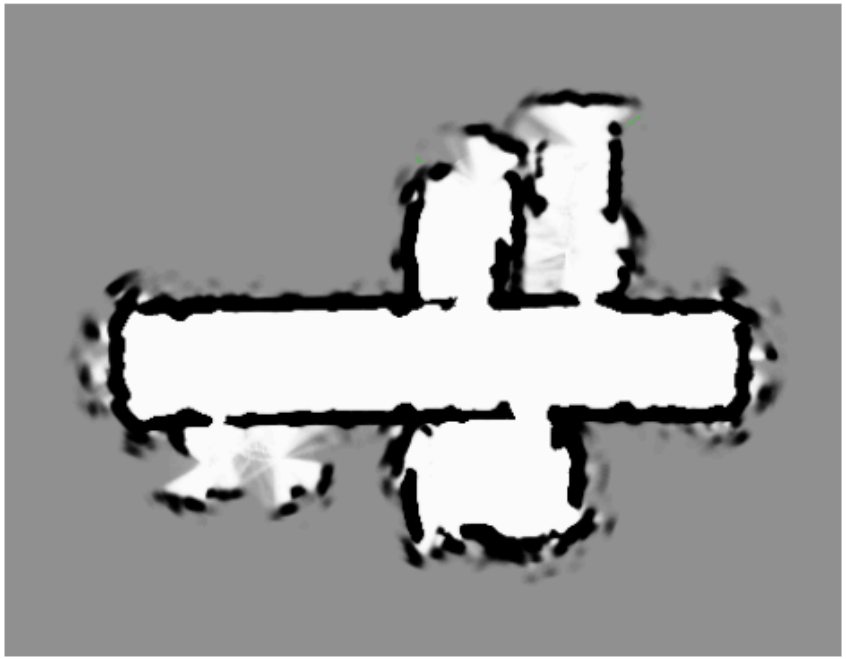
$$P(m_{d,\theta}(x(k)) \mid y(k), x(k)) = P(m_{d,\theta}(x(k)))$$
$$+ \begin{cases} -s(y(k),\theta) & d < y(k) - d_1 \\ -s(y(k),\theta) + \frac{s(y(k),\theta)}{d_1}\ (d - y(k) + d_1) & d < y(k) + d_1 \\ s(y(k),\theta) & d < y(k) + d_2 \\ s(y(k),\theta) - \frac{s(y(k),\theta)}{d_3 - d_2}\ (d - y(k) - d_2) & d < y(k) + d_3 \\ 0 & \text{otherwise.} \end{cases}$$

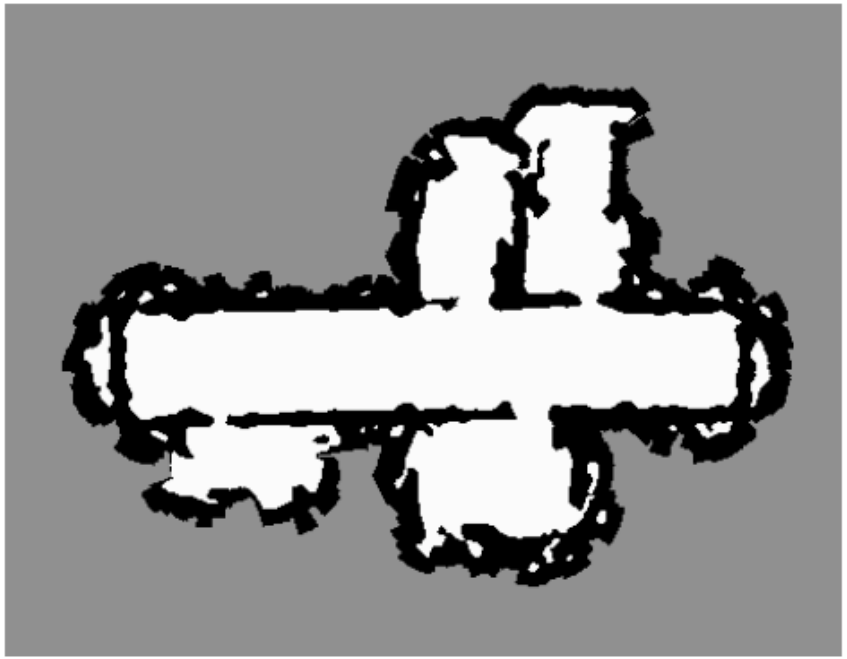
Incremental Updating of Occupancy Grids (Example)

Resulting Map Obtained with Ultrasound Sensors



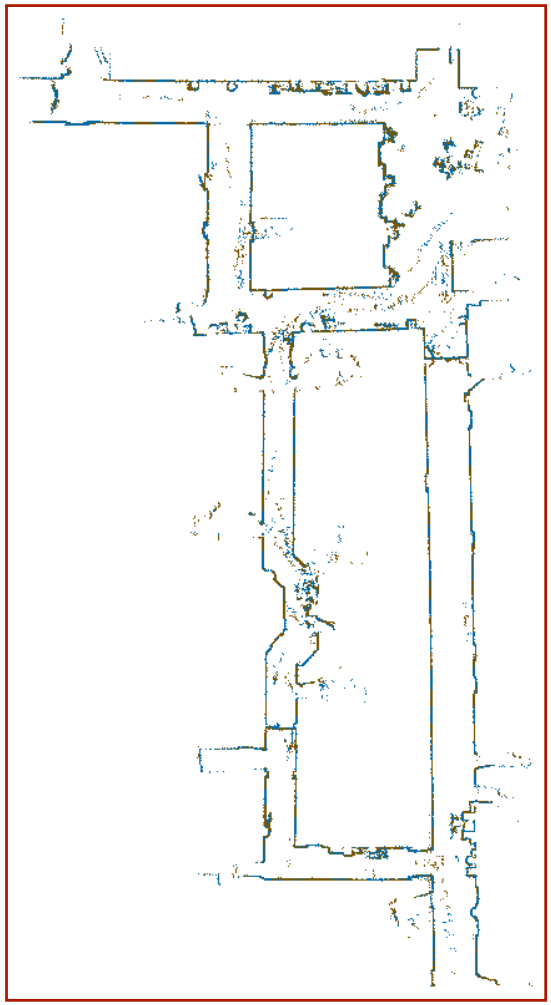
Resulting Occupancy and Maximum Likelihood Map

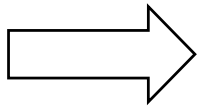


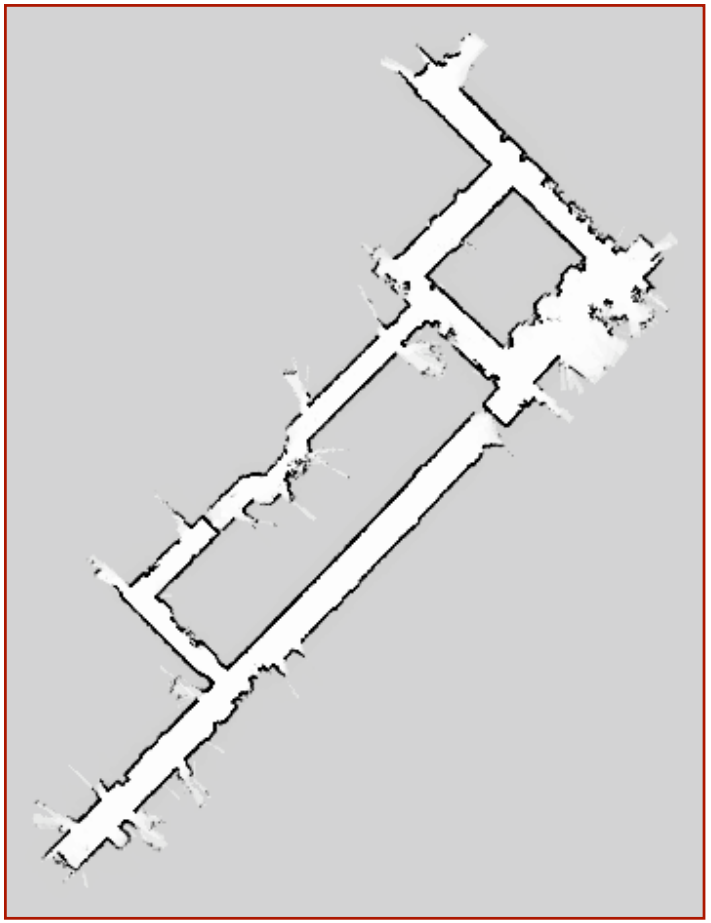


The maximum likelihood map is obtained by clipping the occupancy grid map at a threshold of 0.5

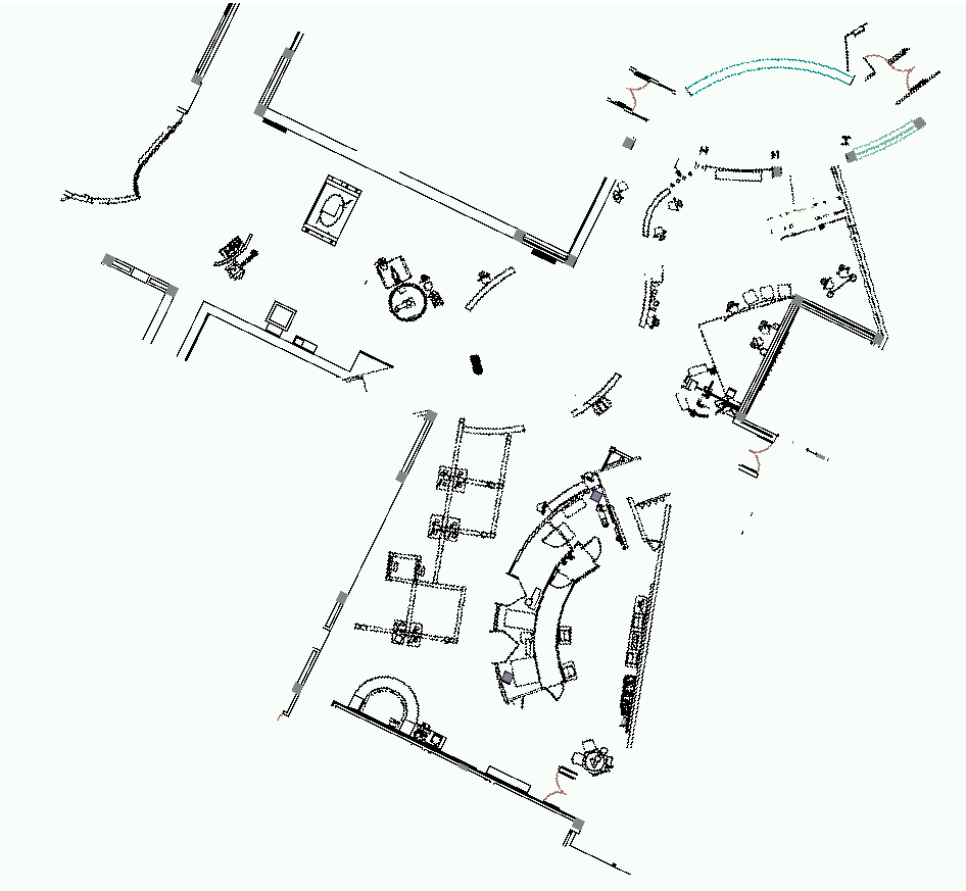
Occupancy Grids: From scans to maps



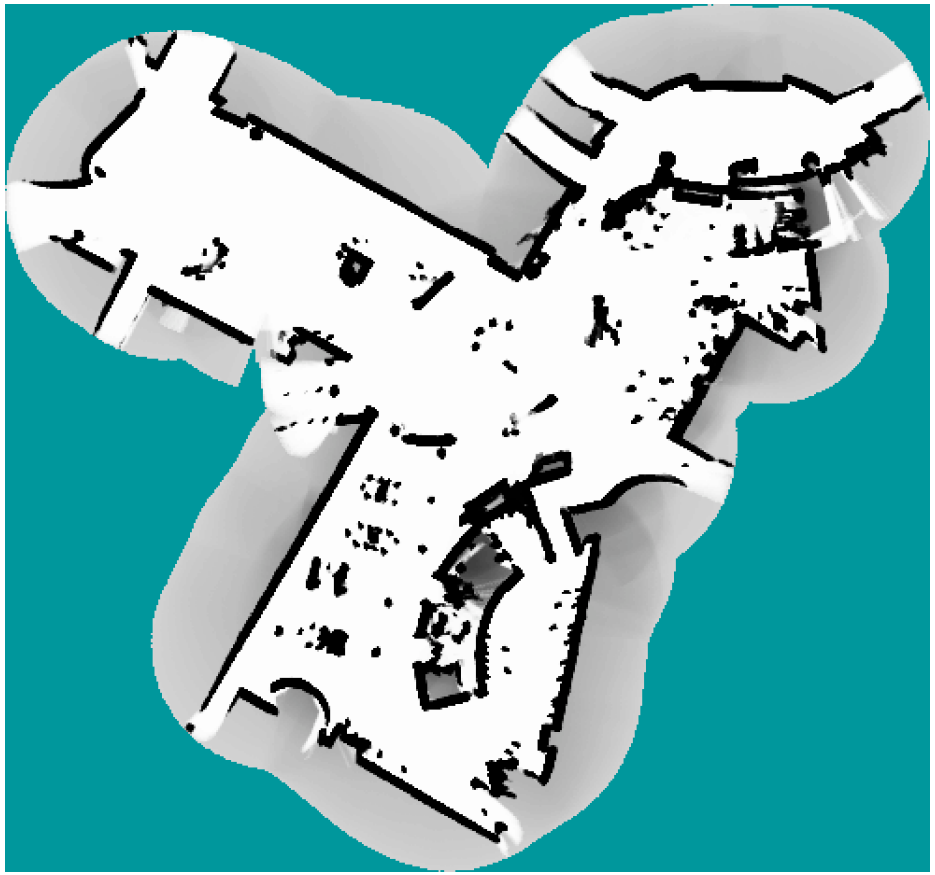




Tech Museum, San Jose



CAD map



occupancy grid map

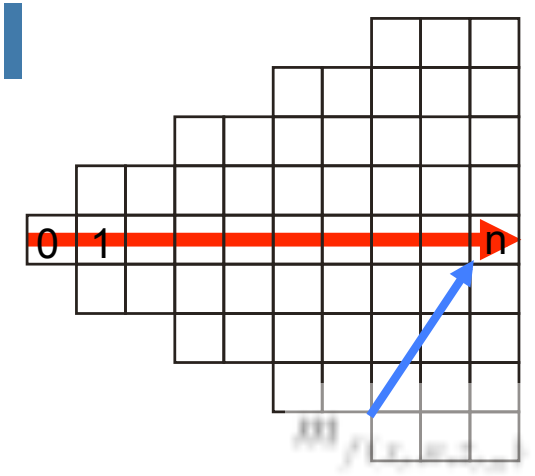
Popular Alternative: Counting

- For every cell count
  - hits(*x*,*y*): number of cases where a beam ended at [*x*,*y*]
  - misses(*x*,*y*): number of cases where a beam passed through *[x,y]*

$$Bel(m^{[xy]}) = \frac{\text{hits}(x, y)}{\text{hits}(x, y) + \text{misses}(x, y)}$$

- Estimated value: *P*(*reflects*(*x*,*y*))
- Ho to derive this model?

The Measurement Model



1. pose at time *t*: x_t
2. beam *n* of scan *t*: $z_{t,n}$
3. maximum range reading: $\varsigma_{t,n} = 1$
4. beam reflected by an object: $\varsigma_{t,n} = 0$

$$p(z_{t,n} \mid x_t, m) = \begin{cases} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \text{if } \varsigma_{t,n} = 1 \\ m_{f(x_t,n,z_{t,n})} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \text{if } \varsigma_{t,n} = 0 \end{cases}$$

Computing the Most Likely Map

- Compute values for m that maximize

$$m^* = \operatorname*{argmax}_m P(m \mid z_{1,\ldots,t}, x_{1,\ldots,t})$$

- Assuming a uniform prior probability for p(m), this is equivalent to maximizing the likelihood of the observations

$$m^* = \operatorname*{argmax}_m P(m \mid z_{1,\ldots,t}, x_{1,\ldots,t})$$

$$\operatorname*{argmax}_m \prod_{k=1}^{t} P(z_k \mid m, x_k)$$

$$\operatorname*{argmax}_m \sum_{k=1}^{t} \log P(z_k \mid m, x_k)$$

Computing the Most Likely Map

$$m^* = \arg\max_m \left[\sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{n=1}^{N} \left(I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n}) \cdot \ln m_j \right.\right.$$
$$\left.\left. + \sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j) \cdot \ln (1 - m_j) \right) \right]$$

Suppose

$$\alpha_j = \sum_{t=1}^{T} \sum_{n=1}^{N} I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n})$$

$$\beta_j = \sum_{t=1}^{T} \sum_{n=1}^{N} \left[\sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j) \right]$$

Meaning of α_j and β_j

$$\alpha_j = \sum_{t=1}^{T} \sum_{n=1}^{N} I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n})$$

corresponds to the number of times a beam that is not a maximum range beam ended in cell j (*hits(j)*)

$$\beta_j = \sum_{t=1}^{T} \sum_{n=1}^{N} \left[\sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j) \right]$$

corresponds to the umber of times a beam intercepted cell j without ending in it (*misses(j)*).

Computing the Most Likely Map

We assume that all cells m_j are independent:

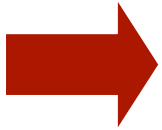
$$m^* = \arg\max_m \left(\sum_{j=1}^{J} \alpha_j \ln m_j + \beta_j \ln(1 - m_j) \right)$$

If we set

$$\frac{\partial m}{\partial m_j} = \frac{\alpha_j}{m_j} - \frac{\beta_j}{1 - m_j} = 0$$

we obtain

$$m_j = \frac{\alpha_j}{\alpha_j + \beta_j}$$



Computing the most likely map amounts to counting how often a cell has reflected a measurement and how often it was intercepted.

Difference between Occupancy Grid Maps and Counting

- The counting model determines how often a cell reflects a beam.
- The occupancy model represents whether or not a cell is occupied by an object.

- Although a cell might be occupied by an object, the reflection probability of this object might be very small.

Example Occupancy Map

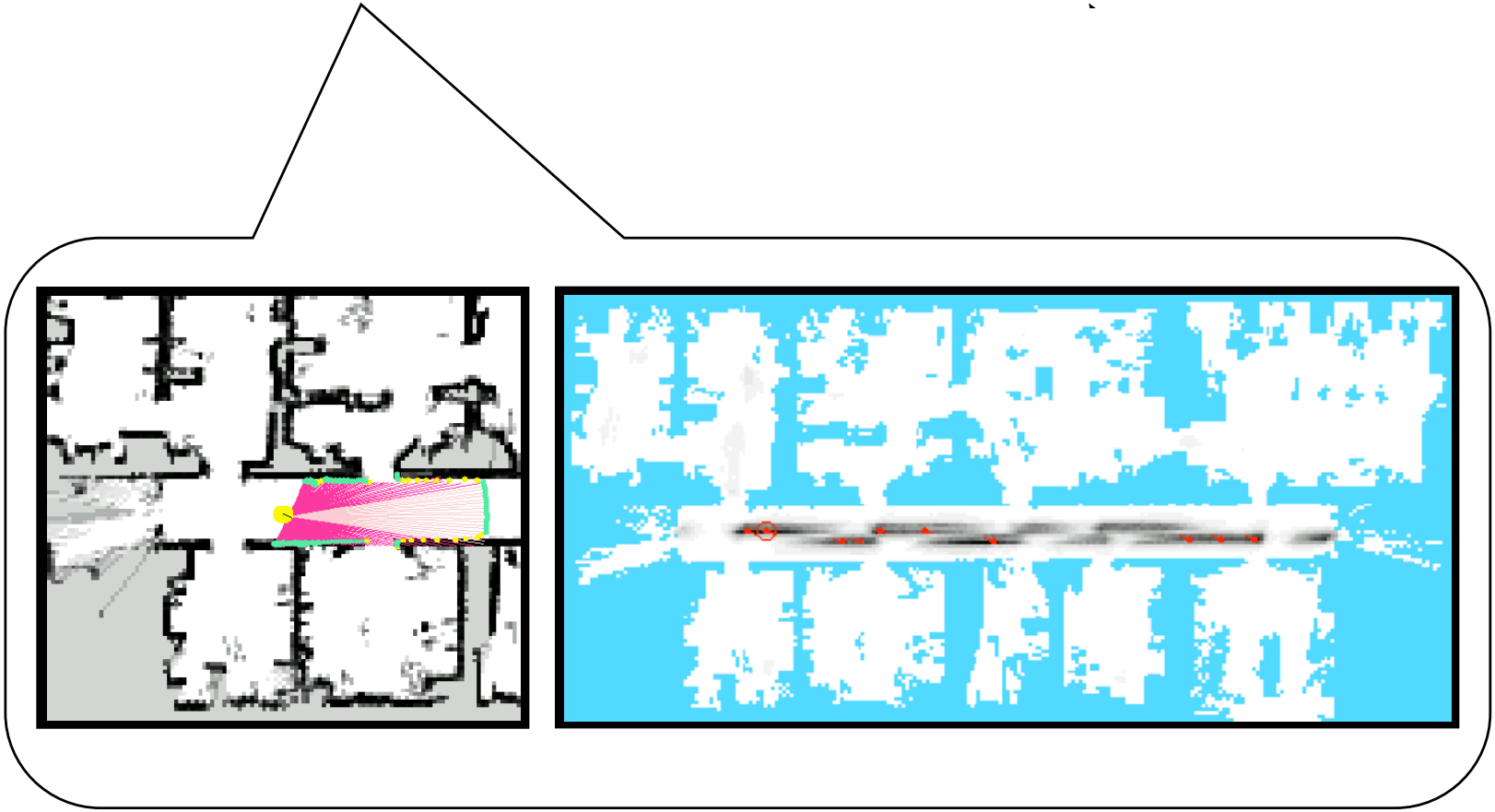
Example Reflection Map

Summary

- **Occupancy grid maps** are a popular approach to represent the environment of a mobile robot given known poses.
- It stores the **posterior probability that the corresponding area in the environment** is occupied.
- Occupancy grid maps can be learned efficiently using a probabilistic approach.
- **Reflection maps** are an alternative representation.
- They store in each cell the **probability that a beam is reflected by this cell**.
- We provided a sensor model for computing the likelihood of measurements and showed that the **counting procedure** underlying reflection maps **corresponds to** that **model** and yields the **maximum likelihood map**.
- Both approaches **consider the individual cells to be independent of each other**.

Localization with Bayes Filters

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



Implementations

- Kalman filters
- Multi-hypothesis tracking
- Grid-based approaches
- Topological maps
- Particle filters

Particle Filters

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

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

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

Particle Filter Algorithm

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

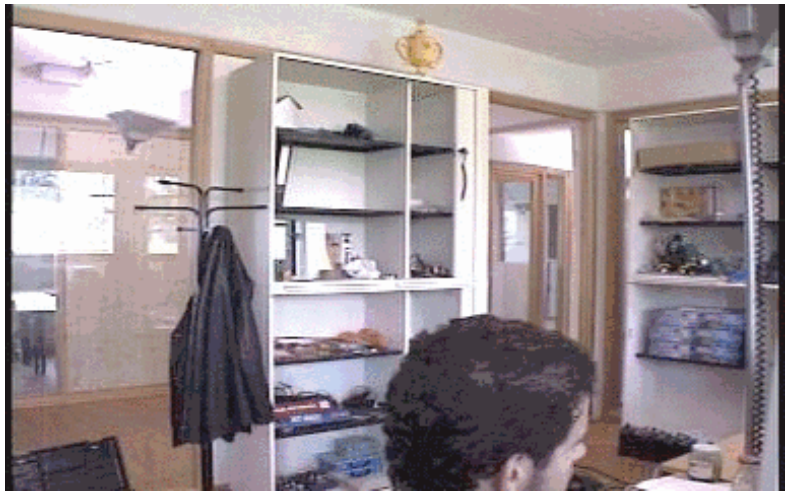
1. Draw x' from $Bel(x)$

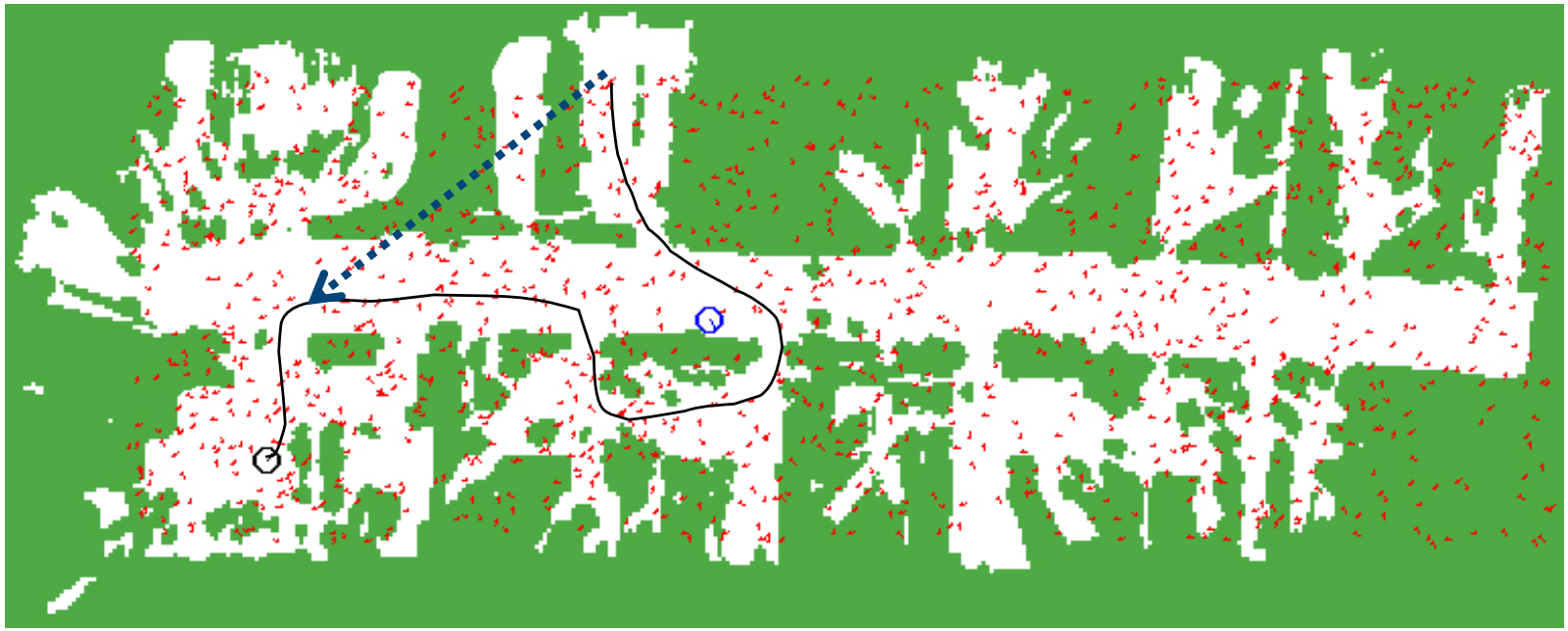
2. Draw x from $p(x \mid u, x')$

3. Importance factor for $x : w = \alpha\, p(z \mid x)$

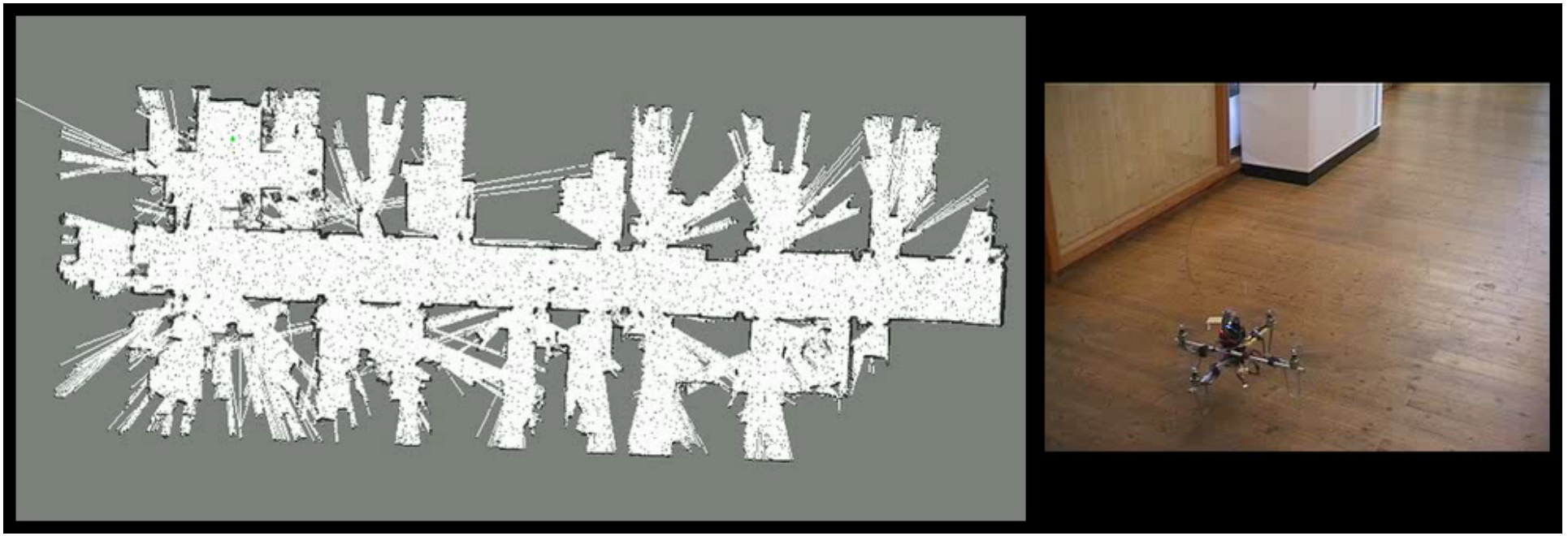
4. Re-sample

Vision-based Localization





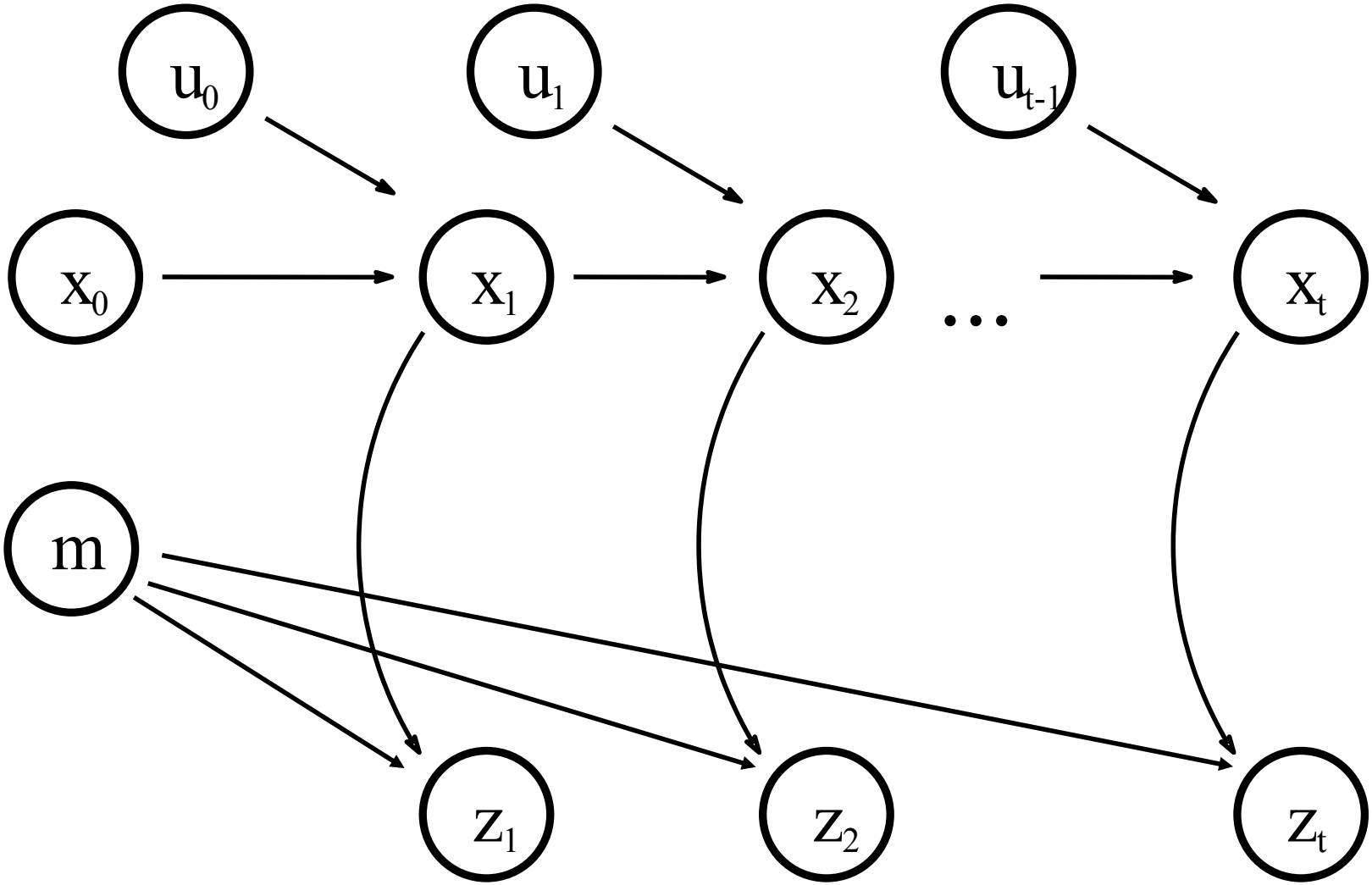
Quadcopter Localization



Simultaneous Localization and Mapping (SLAM)

- To determine its position, the **robot needs a map**.
- During mapping, the robot **needs to know its position** to learn a consistent model
- Simultaneous localization and mapping (SLAM) is a "**chicken and egg problem**"

A Graphical Model for SLAM

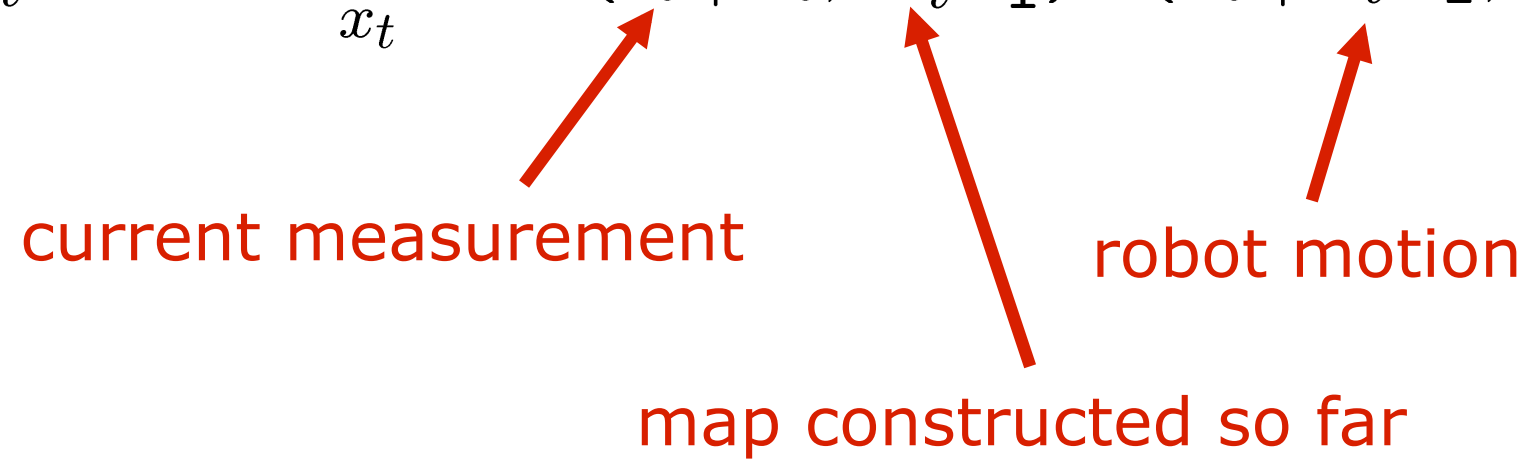


Why SLAM is Hard: Raw Odometry

Scan Matching

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

$$x_t^* = \operatorname*{argmax}_{x_t} P(z_t \mid x_t, m_{t-1}^*) P(x_t \mid u_{t-1}, x_{t-1}^*)$$



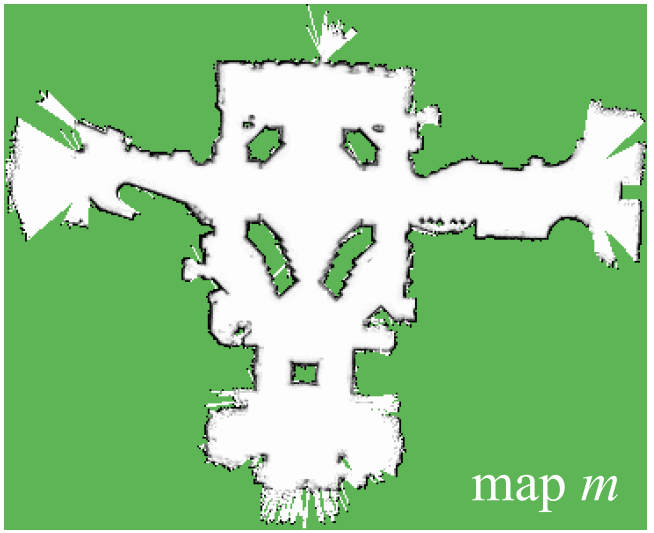
Mapping using Scan Matching

Probabilistic Formulation of SLAM

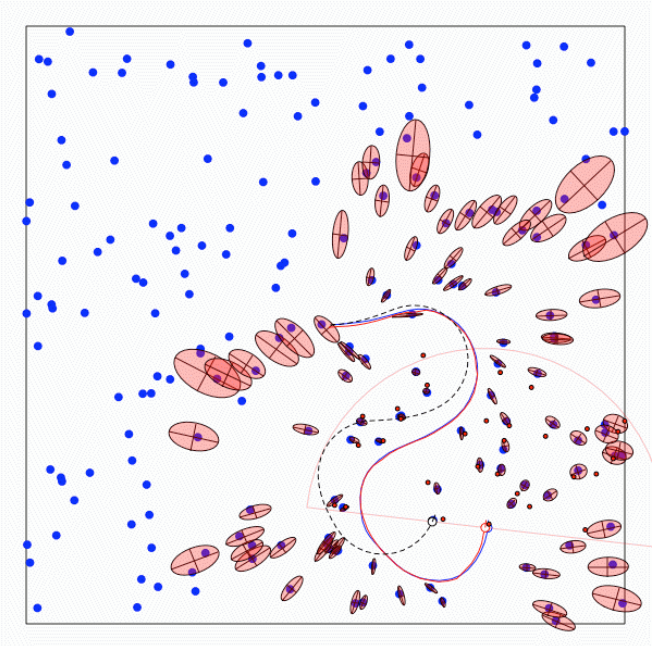
$$Bel(x, m \mid z, u) \;=\; \alpha\, p(z \mid x, m) \int_{x'} p(x \mid u, x') Bel(x', m) dx'$$

n=axb dimensions

three dimensions



Landmark-based EKF SLAM



A robot is exploring an unknown, static environment.

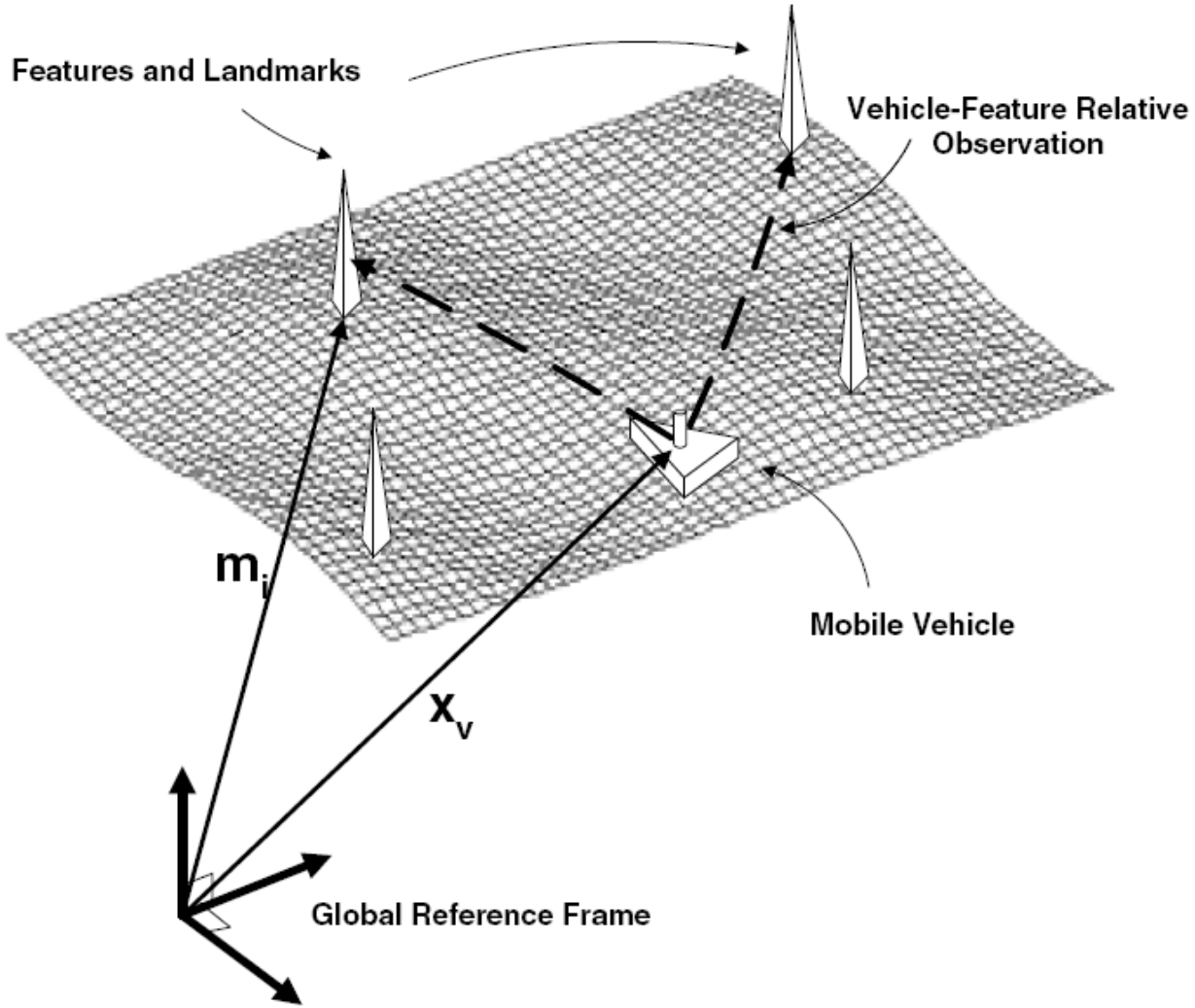
Given:

- The robot's controls
- Observations of nearby point features

Estimate:

- Map of features
- Path of the robot

Structure of the Landmark-based SLAM-Problem



Kalman Filter Algorithm

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

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

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

9. Return μ_t, Σ_t

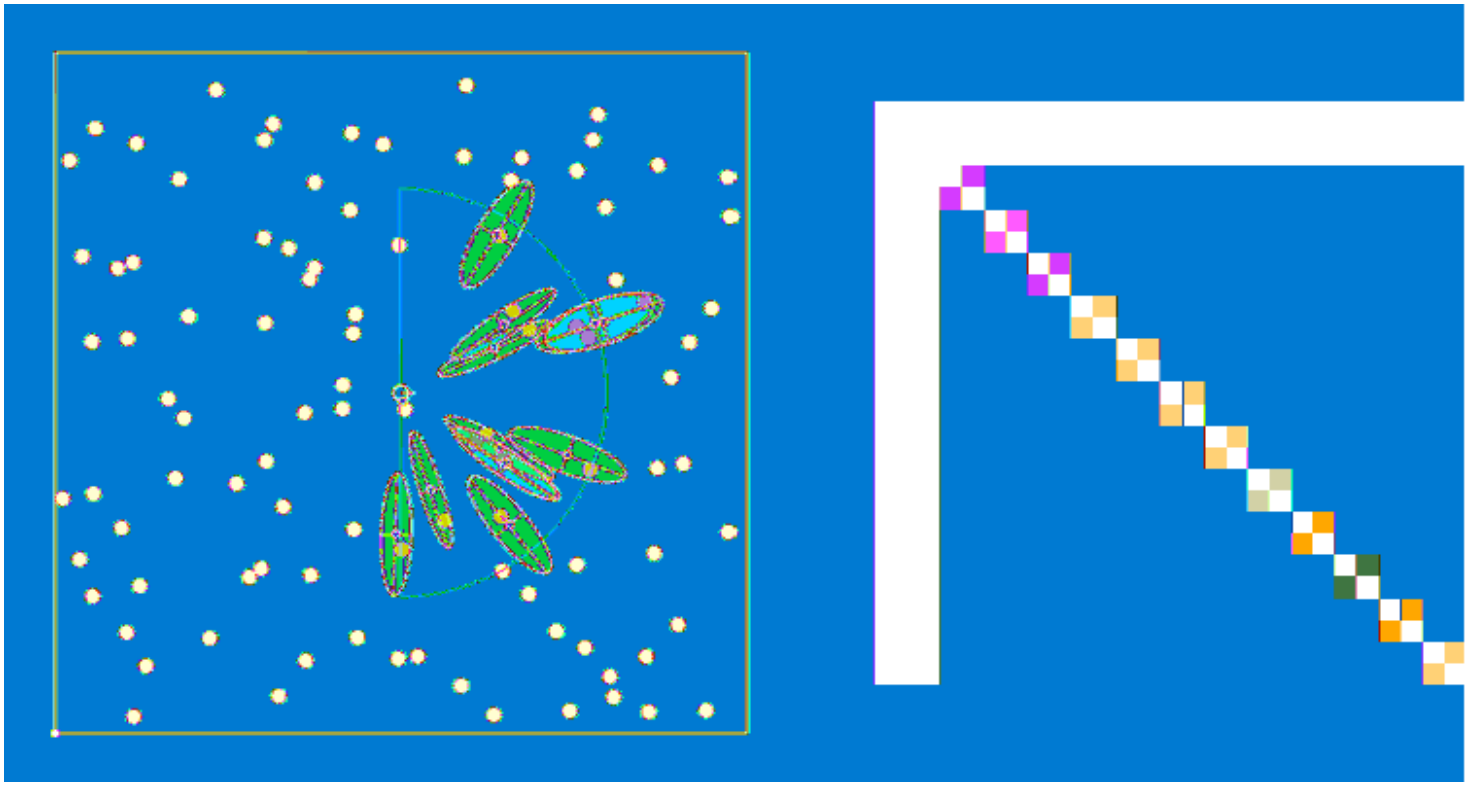
(E)KF-SLAM

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

$$Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \mathsf{M} \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \mathsf{L} & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \mathsf{L} & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \mathsf{L} & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \mathsf{L} & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \mathsf{L} & \sigma_{l_2 l_N} \\ \mathsf{M} & \mathsf{M} & \mathsf{M} & \mathsf{M} & \mathsf{M} & \mathsf{O} & \mathsf{M} \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \mathsf{L} & \sigma_{l_N}^2 \end{pmatrix} \right\rangle$$

- Can handle hundreds of dimensions

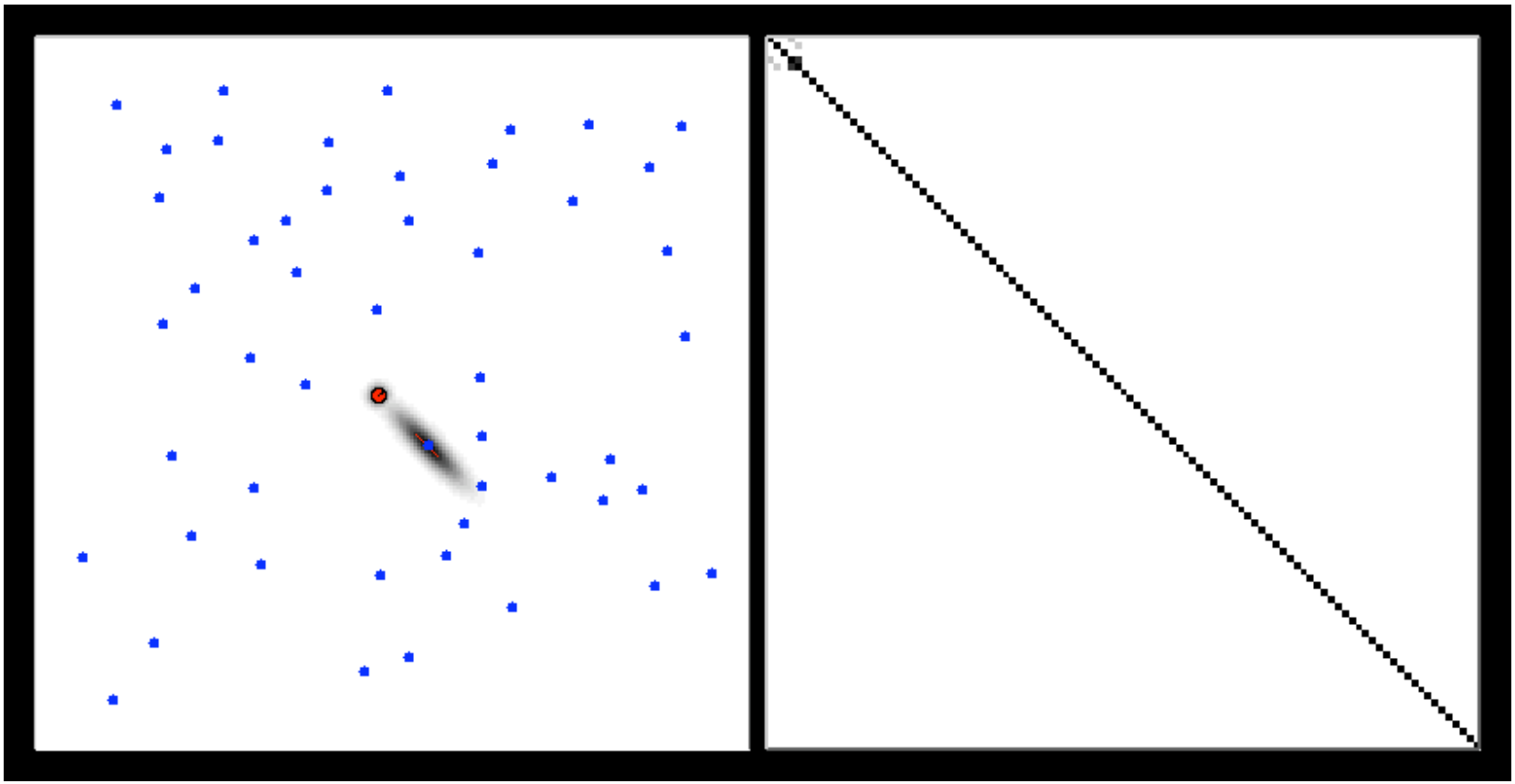
Classical Solution – The EKF



Blue path = true path **Red path** = estimated path **Black path** = odometry

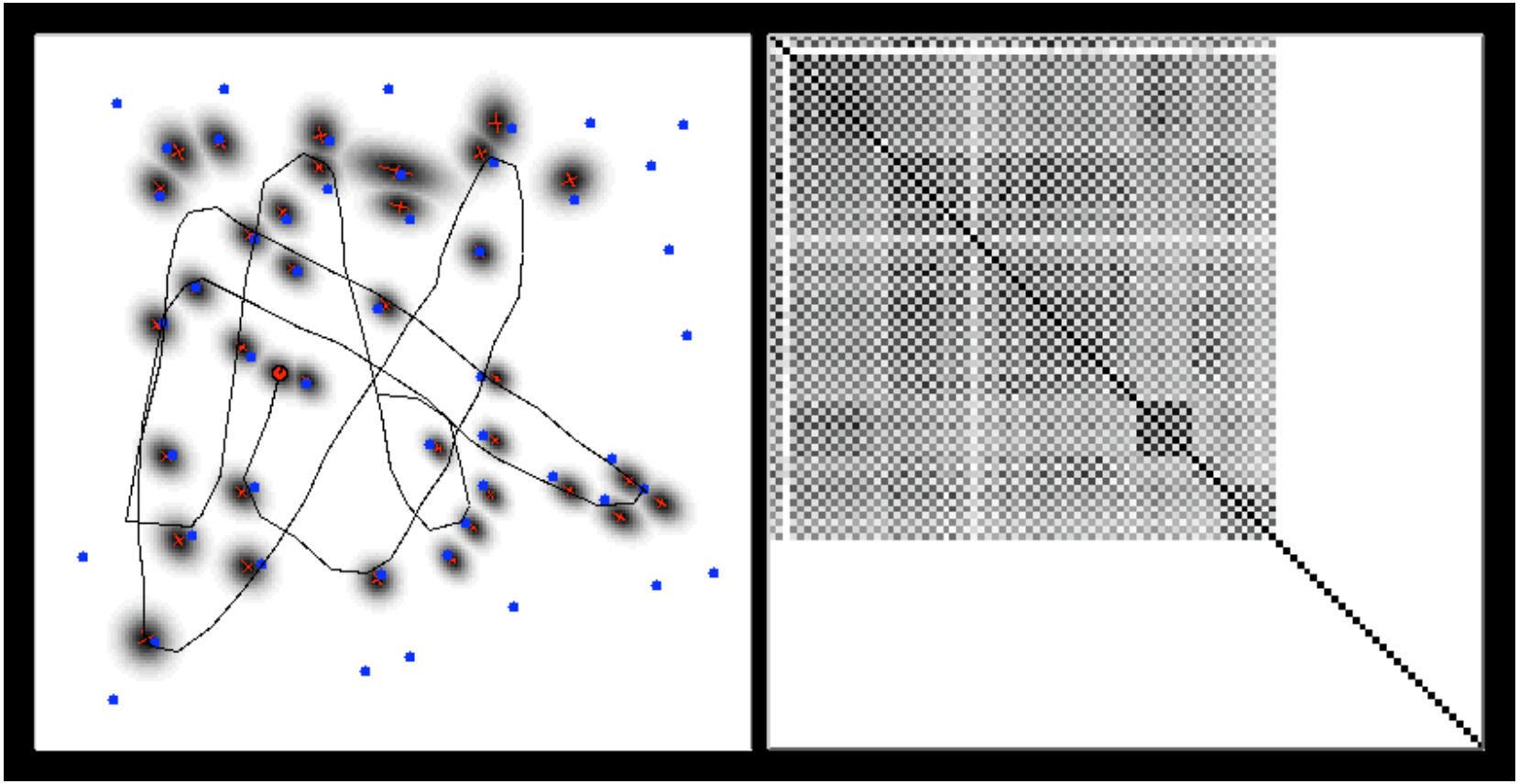
- Approximate the SLAM posterior with a high-dimensional Gaussian [Smith & Cheesman, 1986] …
- Single hypothesis data association

EKF-SLAM



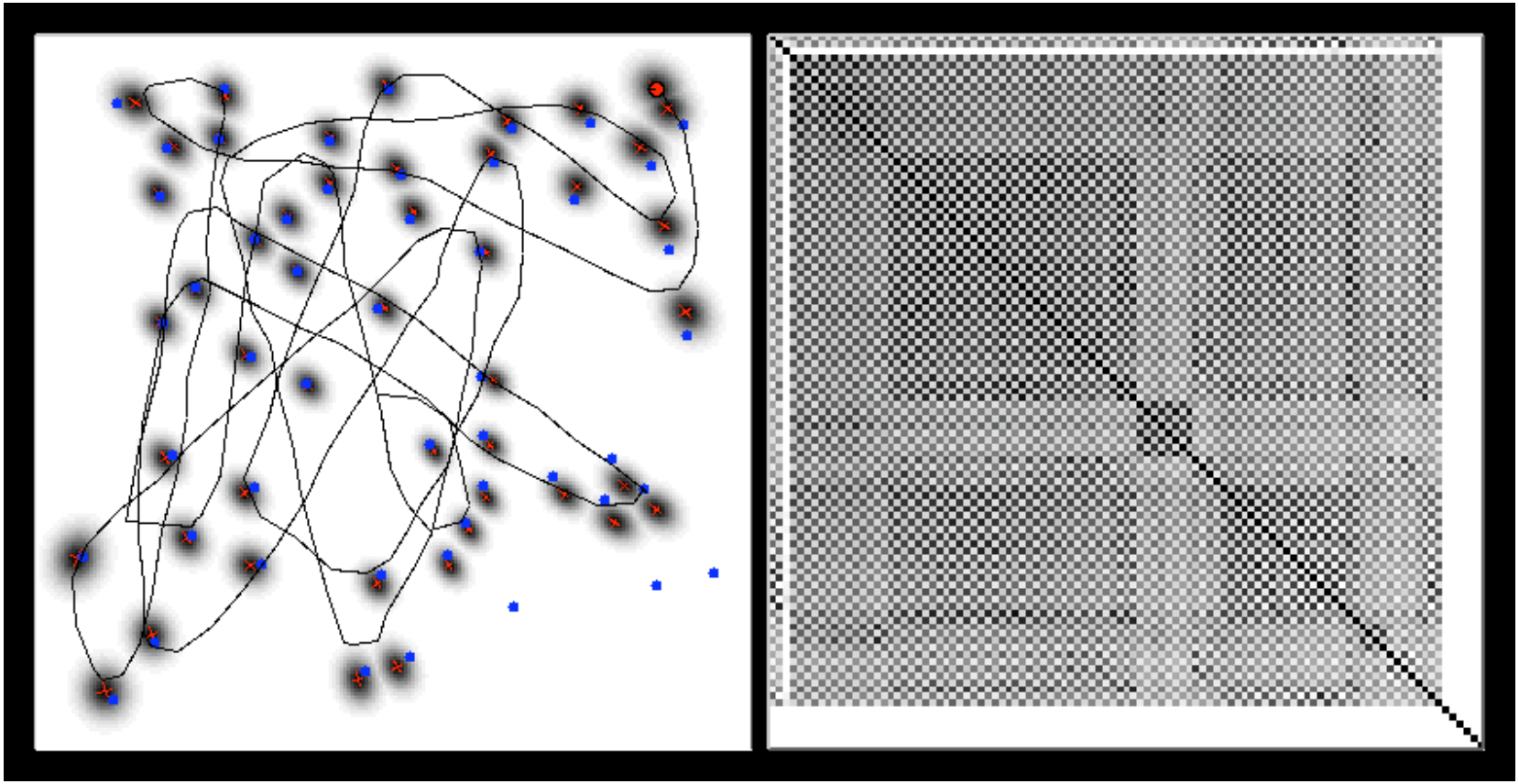
Map Correlation matrix

EKF-SLAM



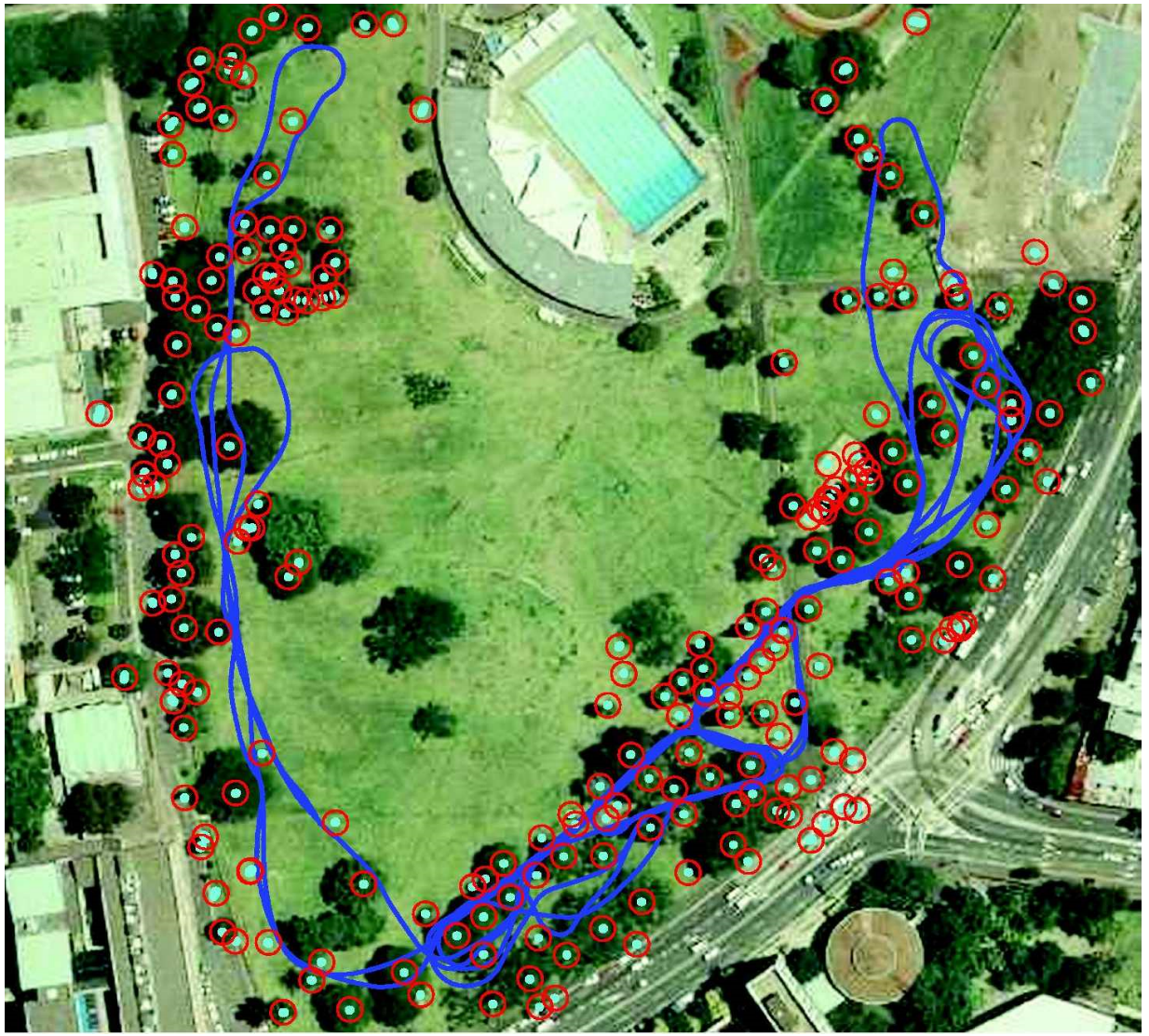
Map Correlation matrix

EKF-SLAM



Map Correlation matrix

Victoria Park Data Set



[courtesy by E. Nebot]

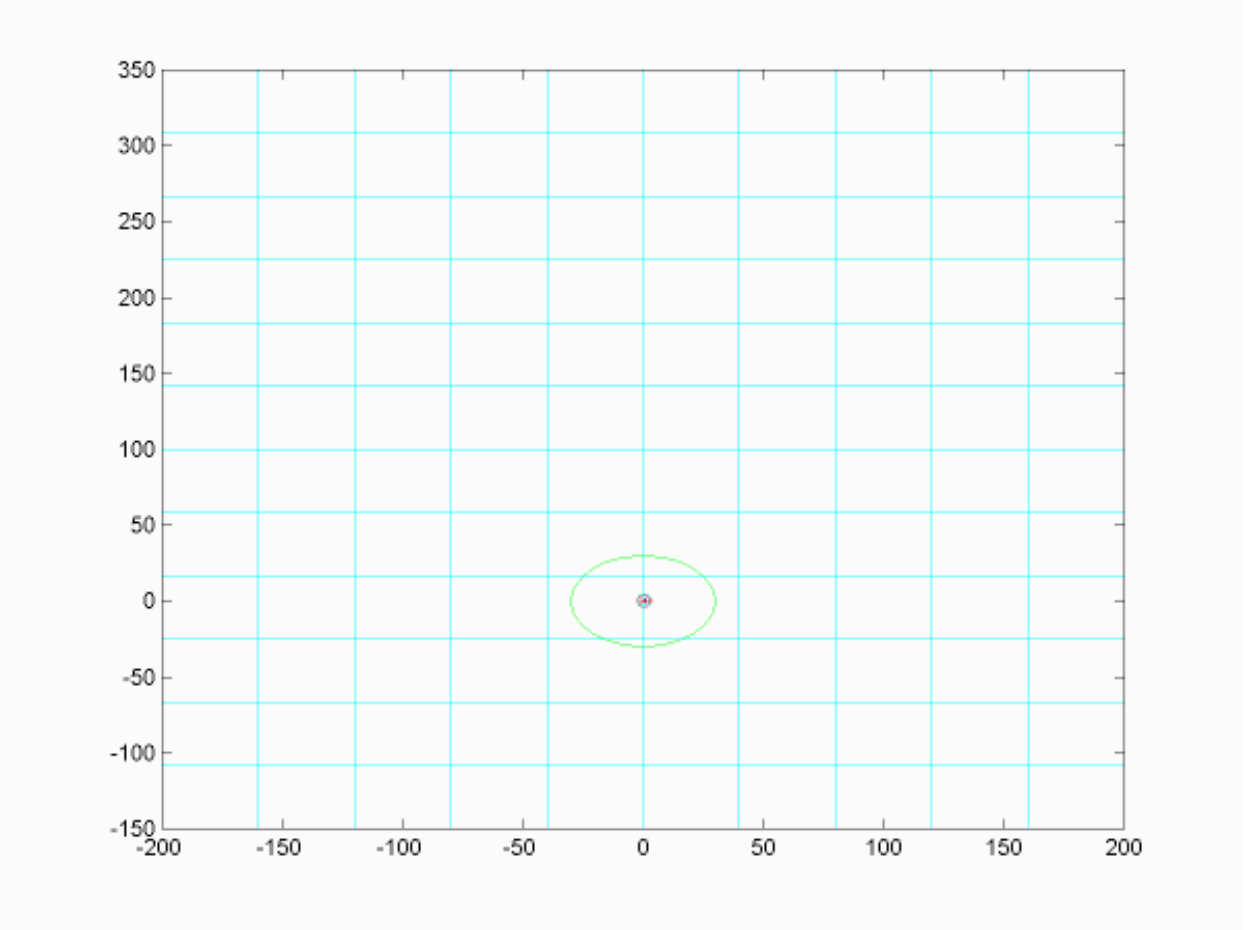
Victoria Park Data Set Vehicle

[courtesy by E. Nebot]

Data Acquisition

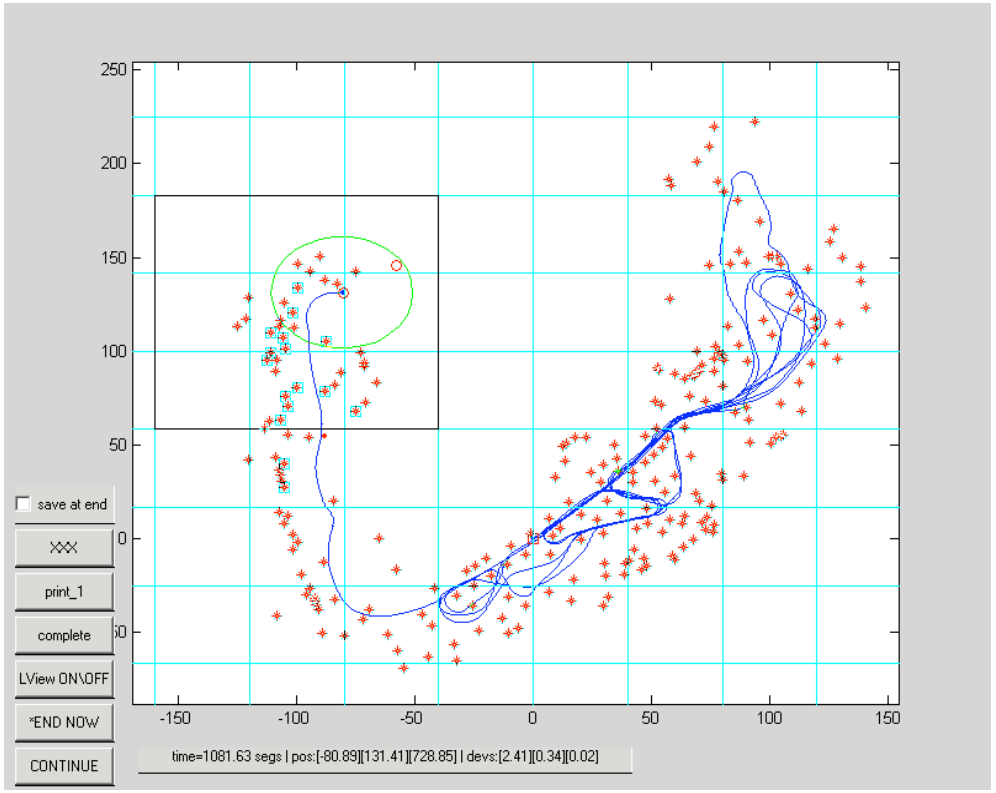
[courtesy by E. Nebot]

SLAM



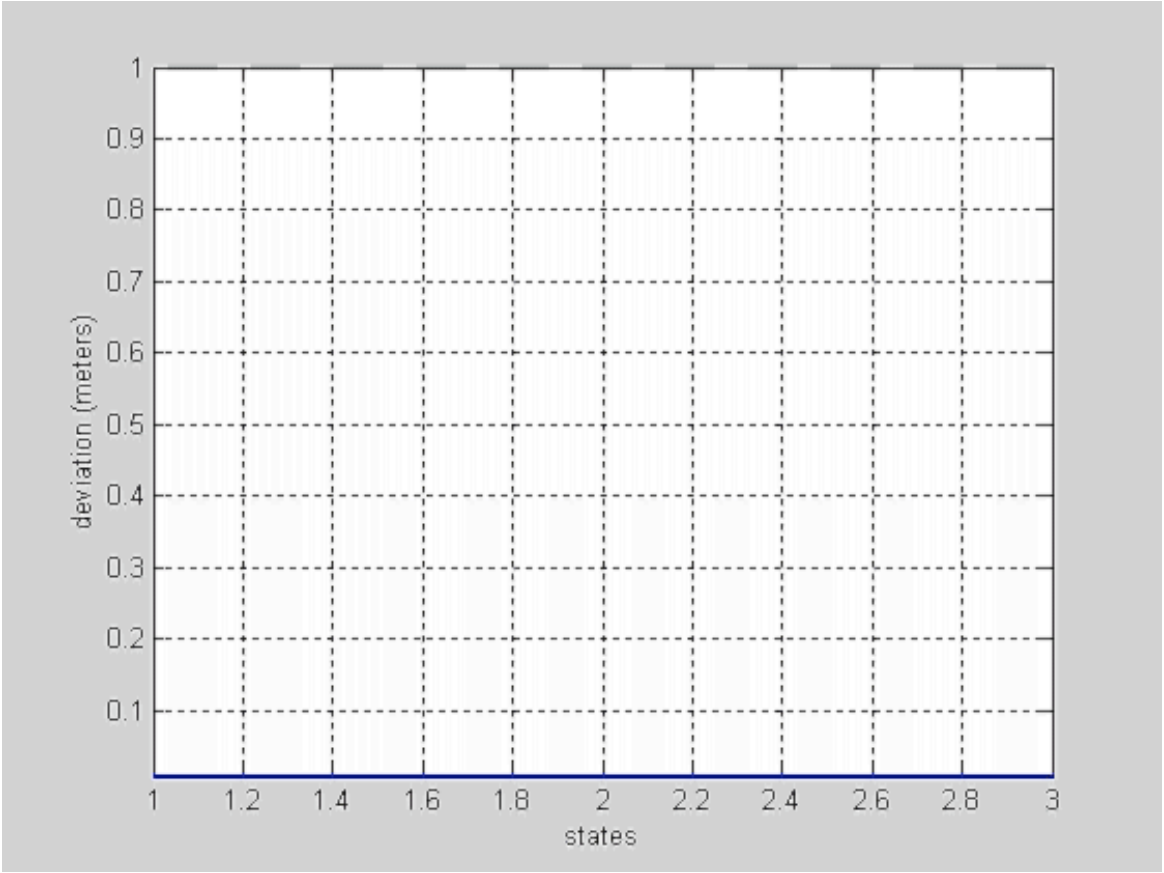
[courtesy by E. Nebot]

Map and Trajectory



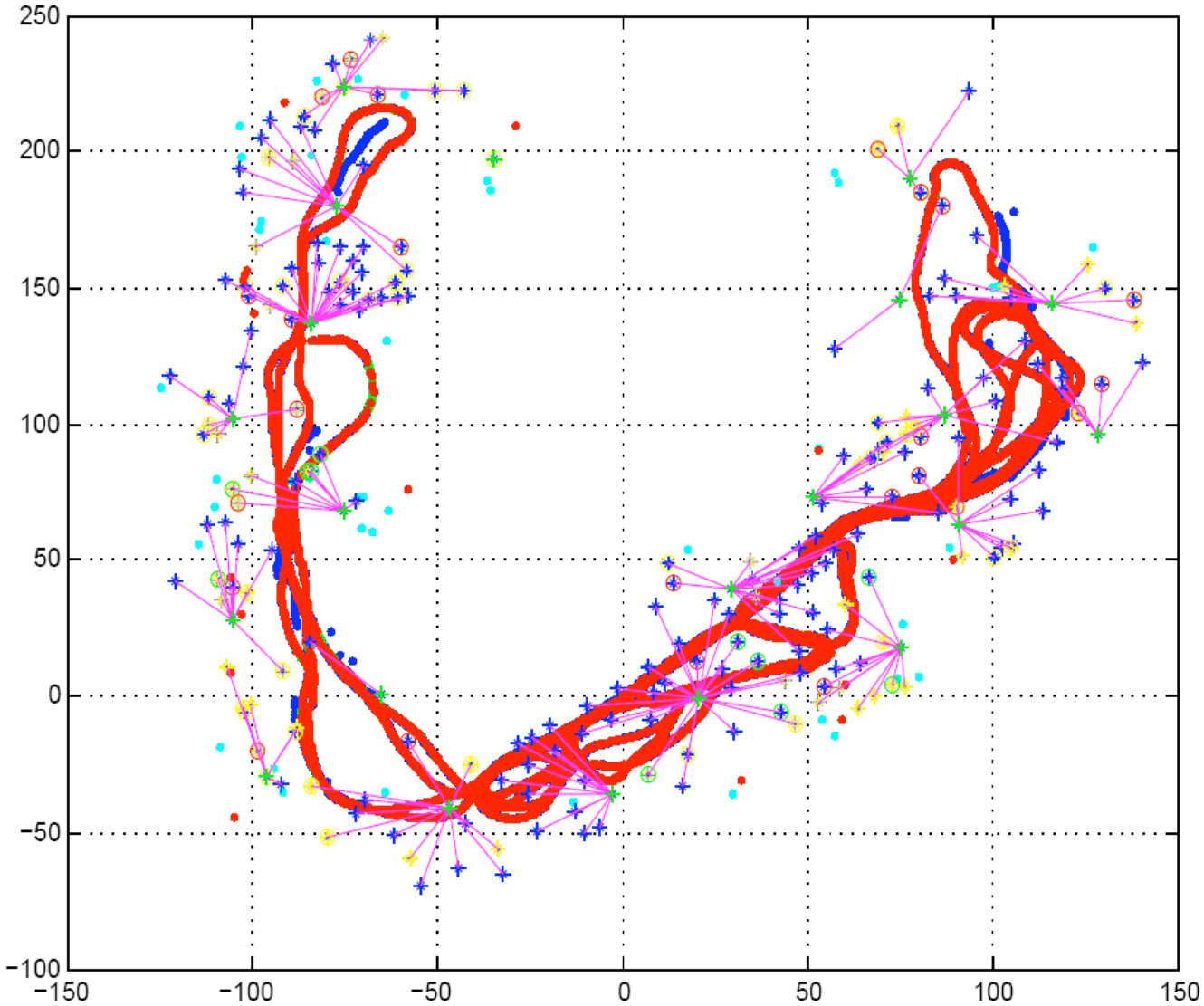
[courtesy by E. Nebot]

Landmark Covariance



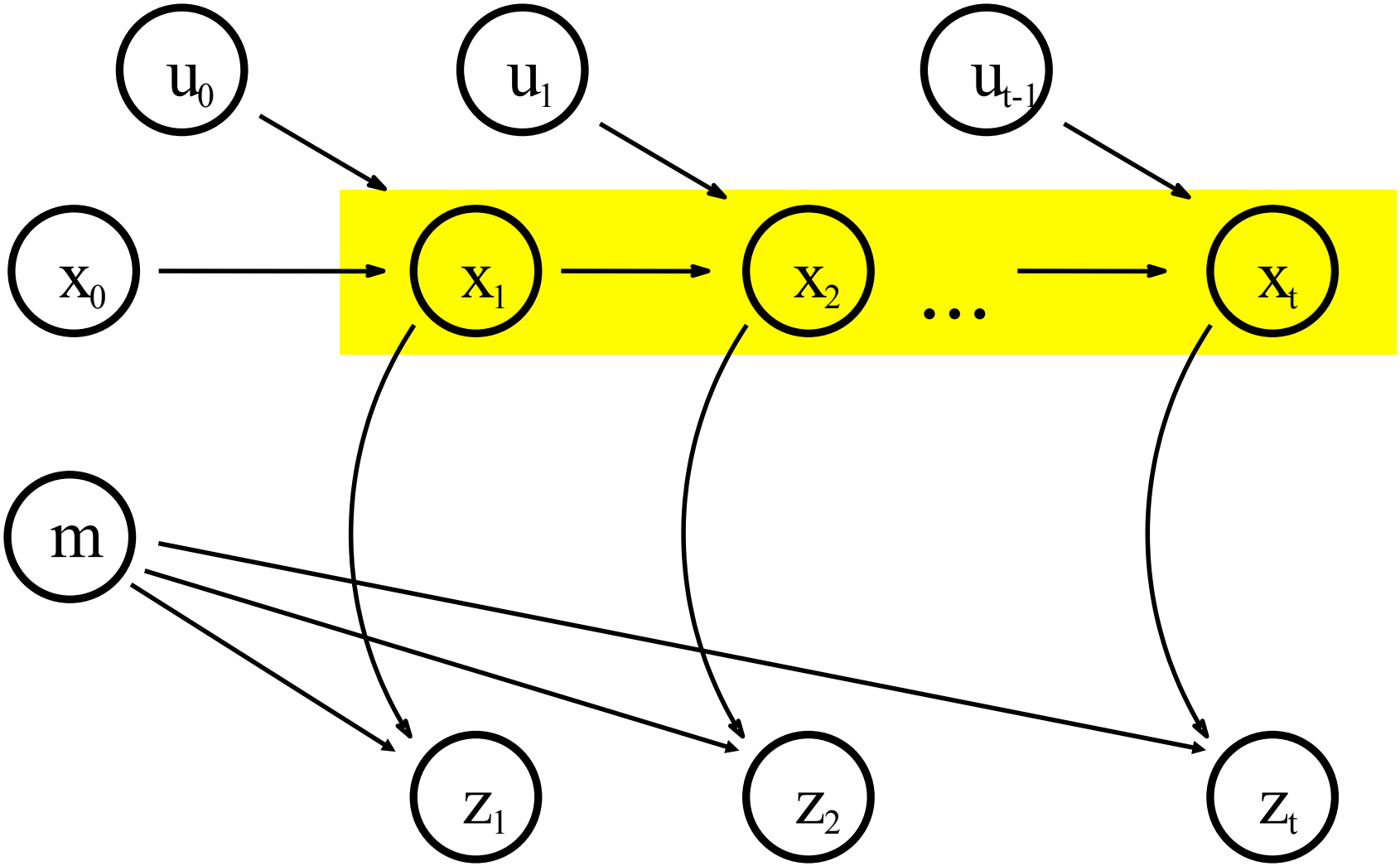
[courtesy by E. Nebot]

Estimated Trajectory



[courtesy by E. Nebot]

FAST-SLAM / Mapping with Rao-Blackwellized PFs



Rao-Blackwellized Particle Filters for SLAM

Observation:

Given the true trajectory of the robot, we can efficiently compute the map
(mapping with known poses).

Idea:

- Use a particle filter to represent potential trajectories of the robot.
- Each particle carries its own map.
- Each particle survives with a probability that is proportional to the likelihood of the observation given that particle and its map.

[Murphy et al., 99]

Factorization Underlying Rao-Blackwellization

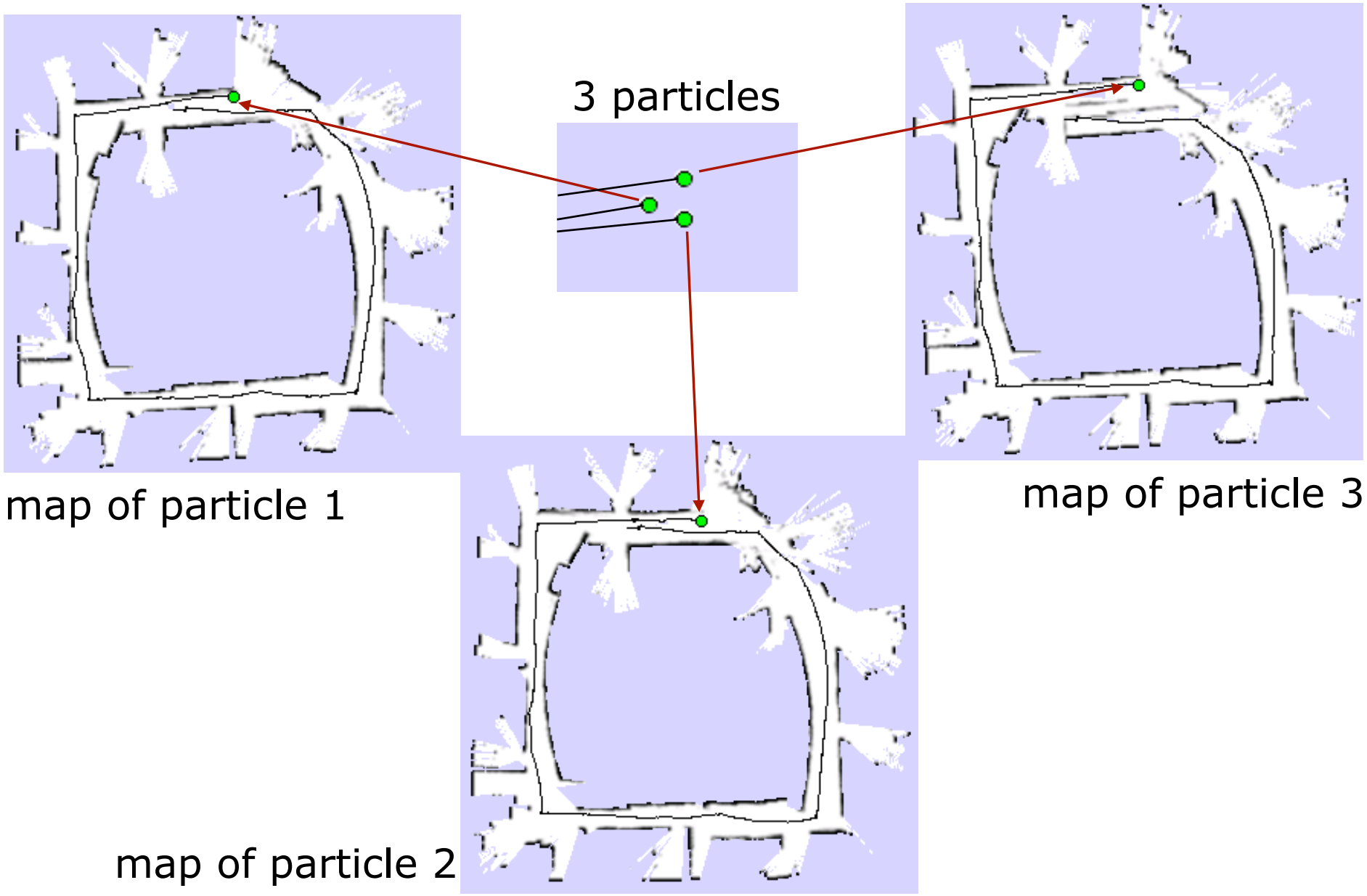
$$Bel(x, m \mid z, u)$$

$$= \; p(m \mid x, z, \cancel{u}) p(x \mid z, u)$$

Mapping with known poses

Particle filter representing trajectory hypotheses

Example


3 particles
map of particle 1
map of particle 3
map of particle 2

Limitations

- A huge number of particles is required.
- This introduces enormous memory and computational requirements.
- It prevents the approach from being applicable in realistic scenarios.

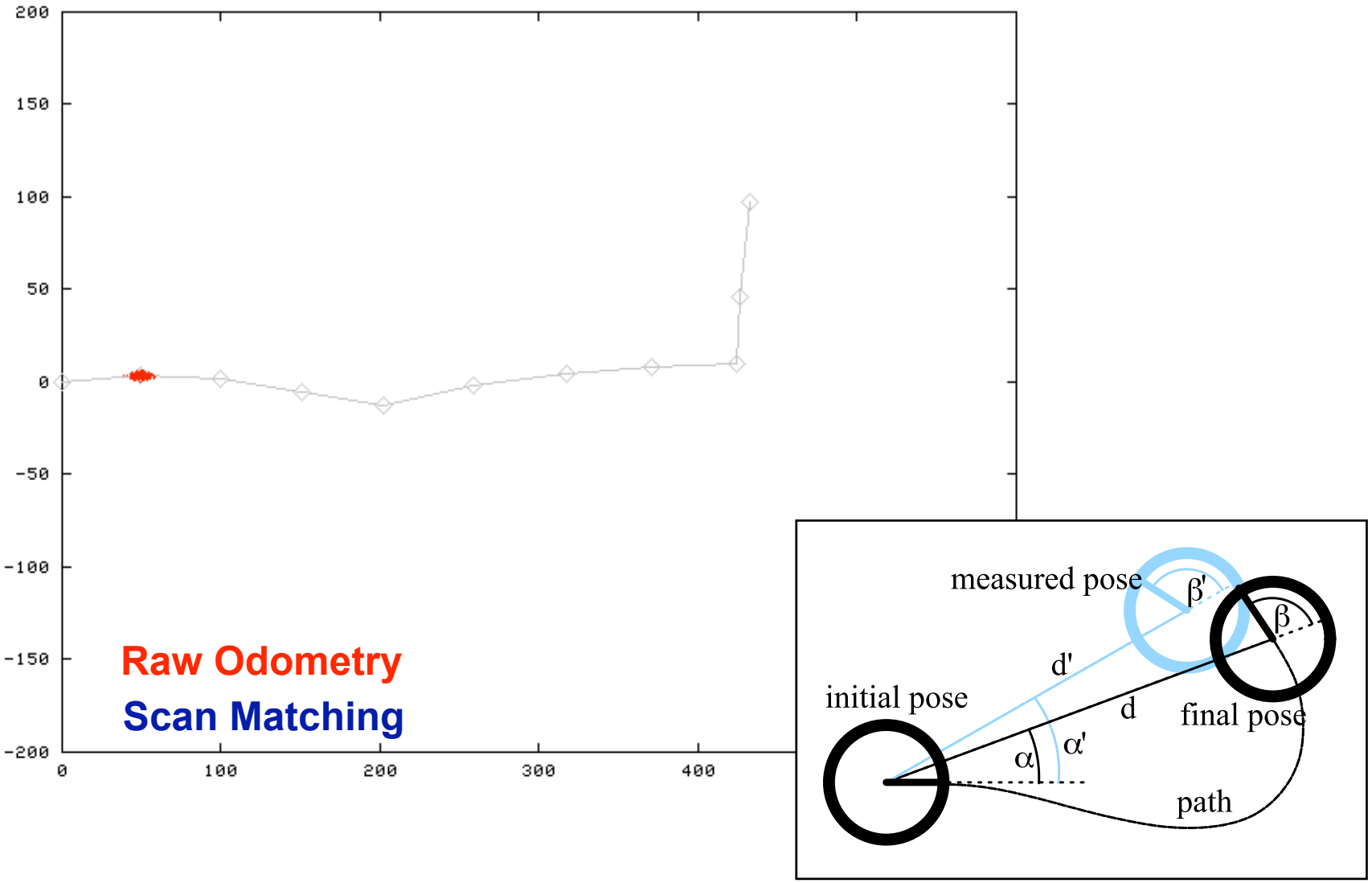
Challenge

Reduction of the number of particles.

Approaches:

- Focused proposal distributions
  (keep the samples in the right place)
- Adaptive re-sampling
  (avoid depletion of relevant particles)

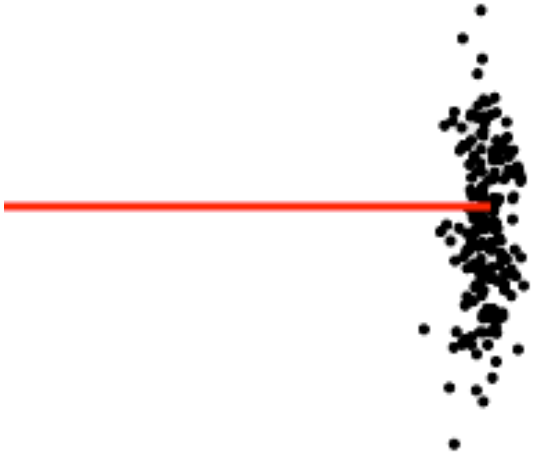
Motion Model for Scan Matching



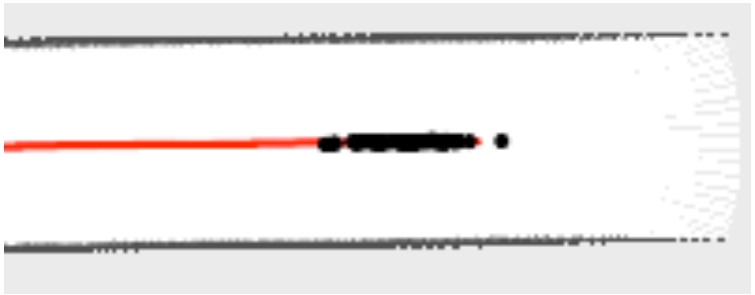
Incorporating the Current Measurement

End of a corridor:

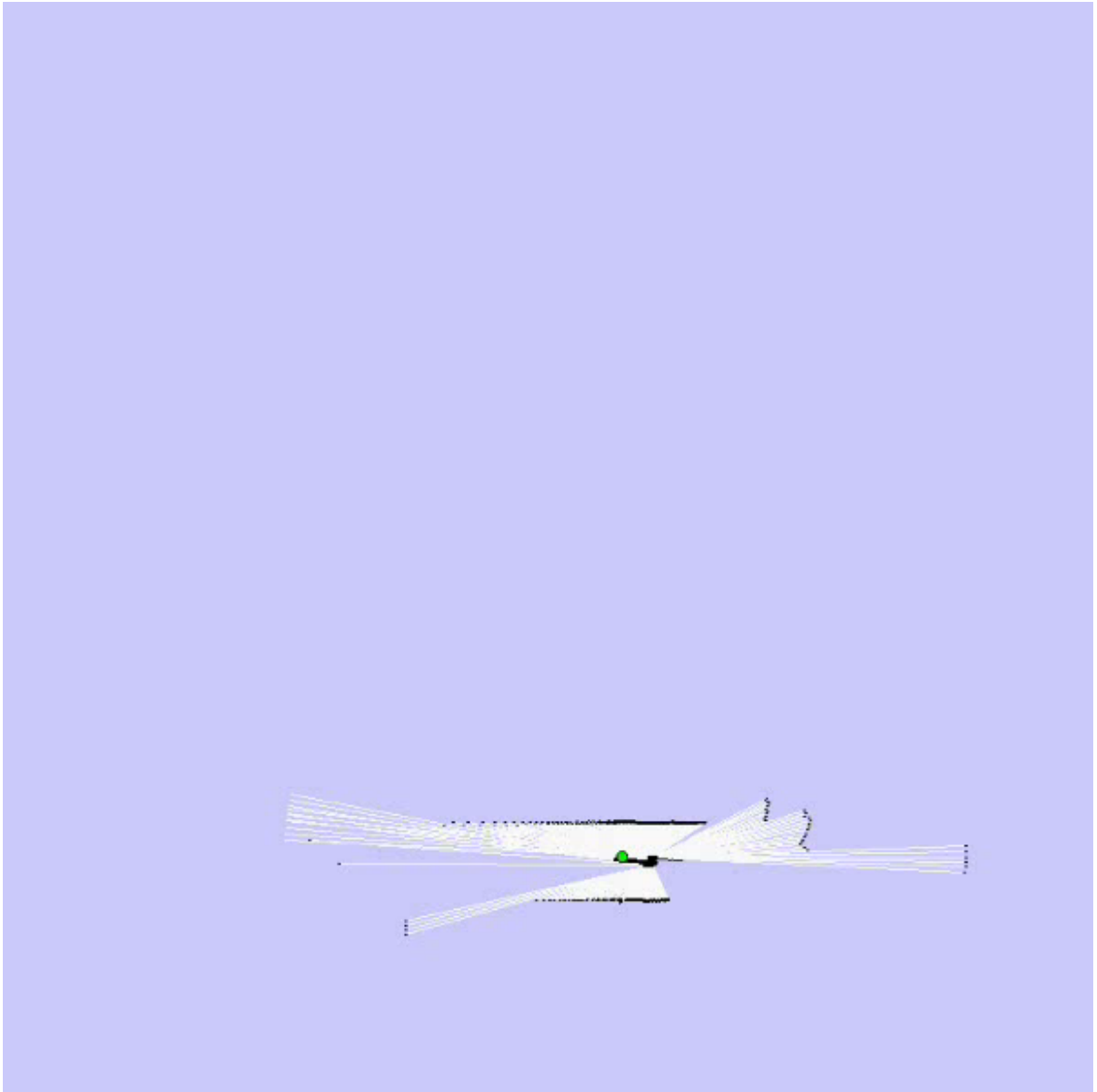
Free space:



Corridor:



Application Example



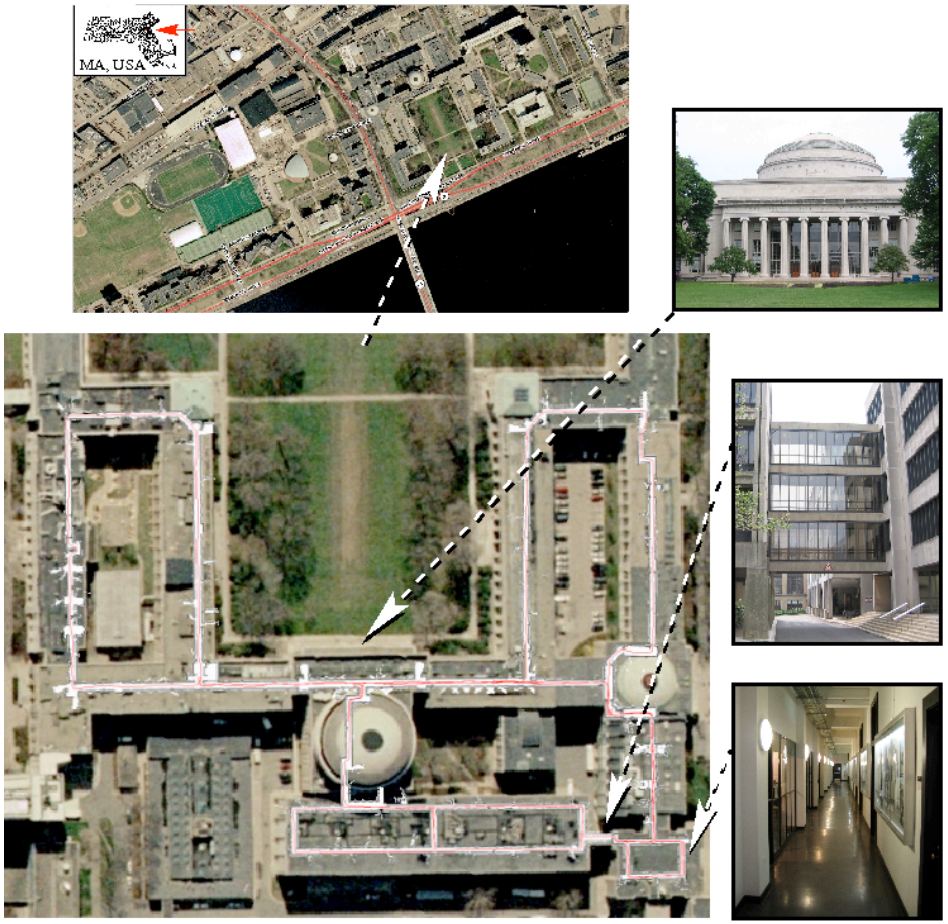
Map of the Intel Lab

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

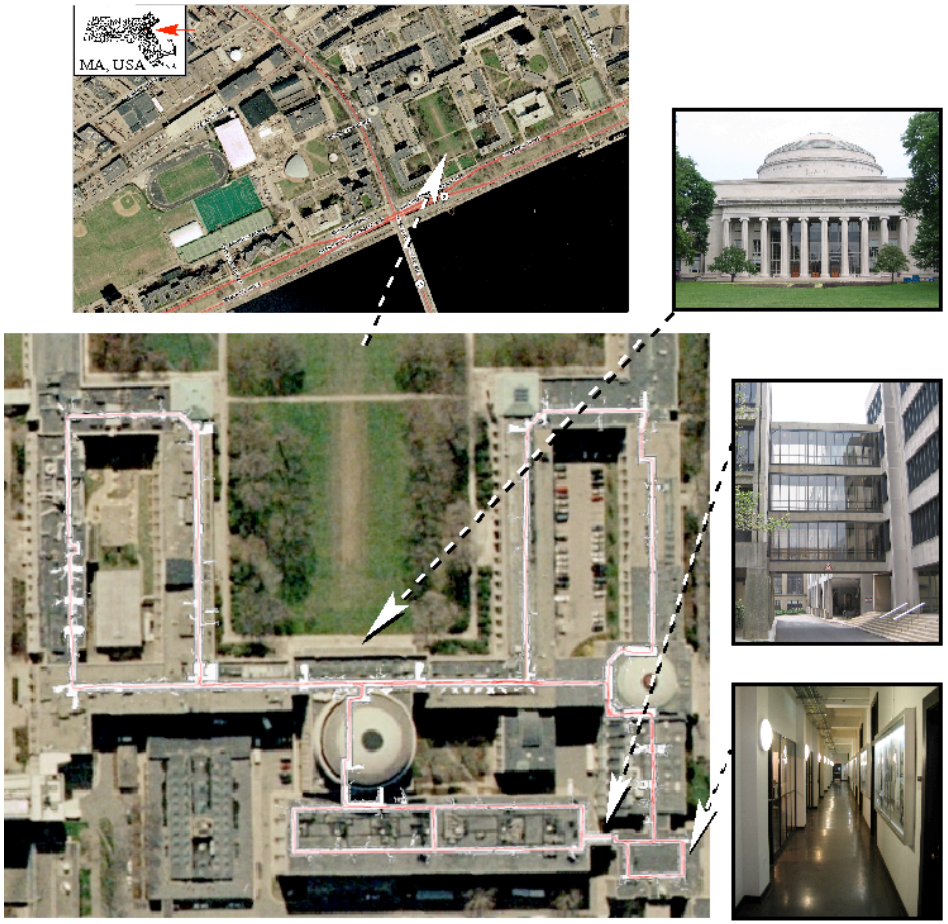
Outdoor Campus Map

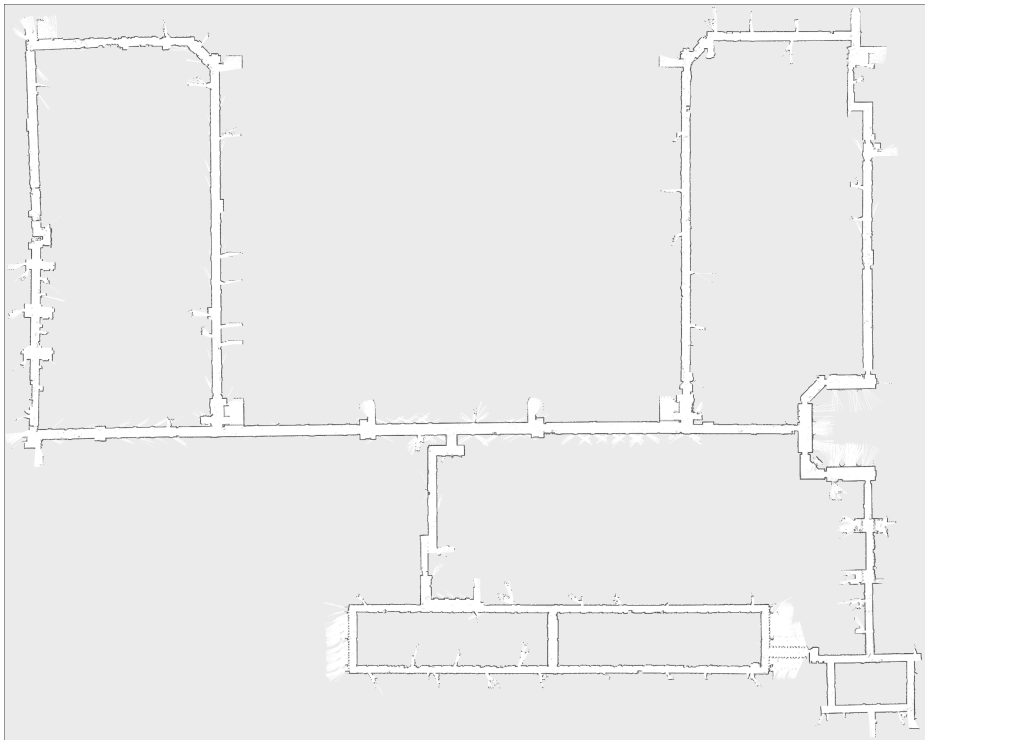
- **30 particles**
- 250x250m^2
- 1.75 km (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map

MIT Kilian Court



MIT Kilian Court





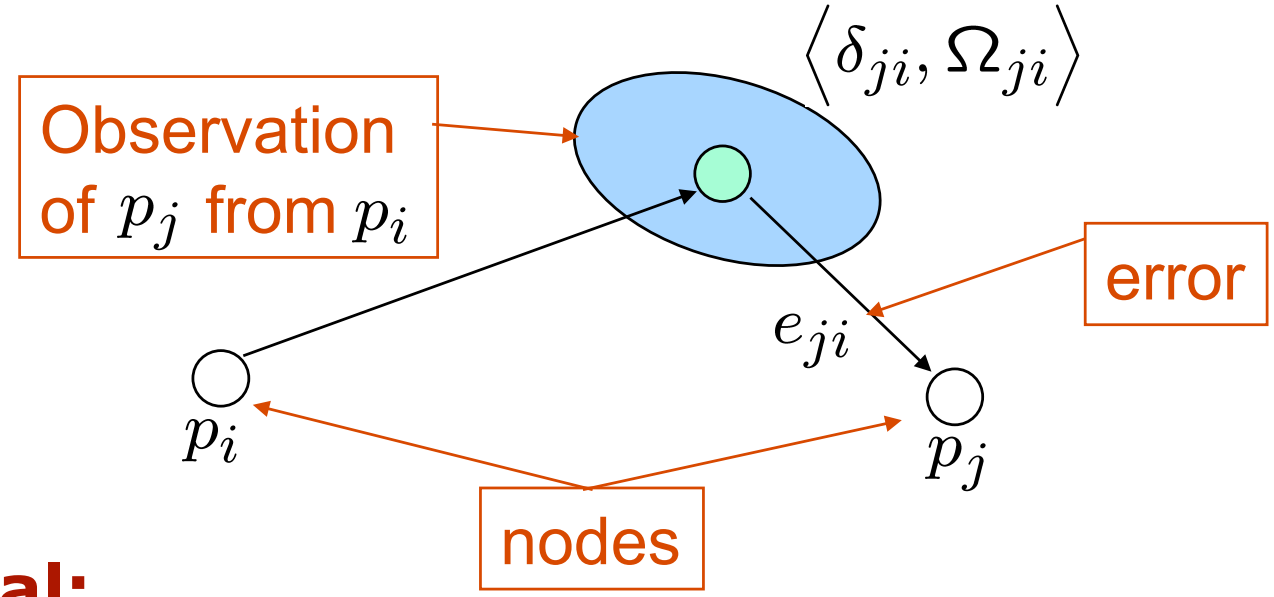
Graph-based Formulation

- Use a **graph** to represent the problem
- **Every node** in the graph **corresponds to a pose** of the robot during mapping
- **Every edge** between two nodes **corresponds to the spatial constraints** between them

- **Goal**:
  Find a configuration of the nodes that **minimize the error** introduced by the constraints

Problem Formulation

- The problem can be described by a graph



Goal:

- Find the assignment of poses to the nodes of the graph which minimizes the negative log likelihood of the observations:

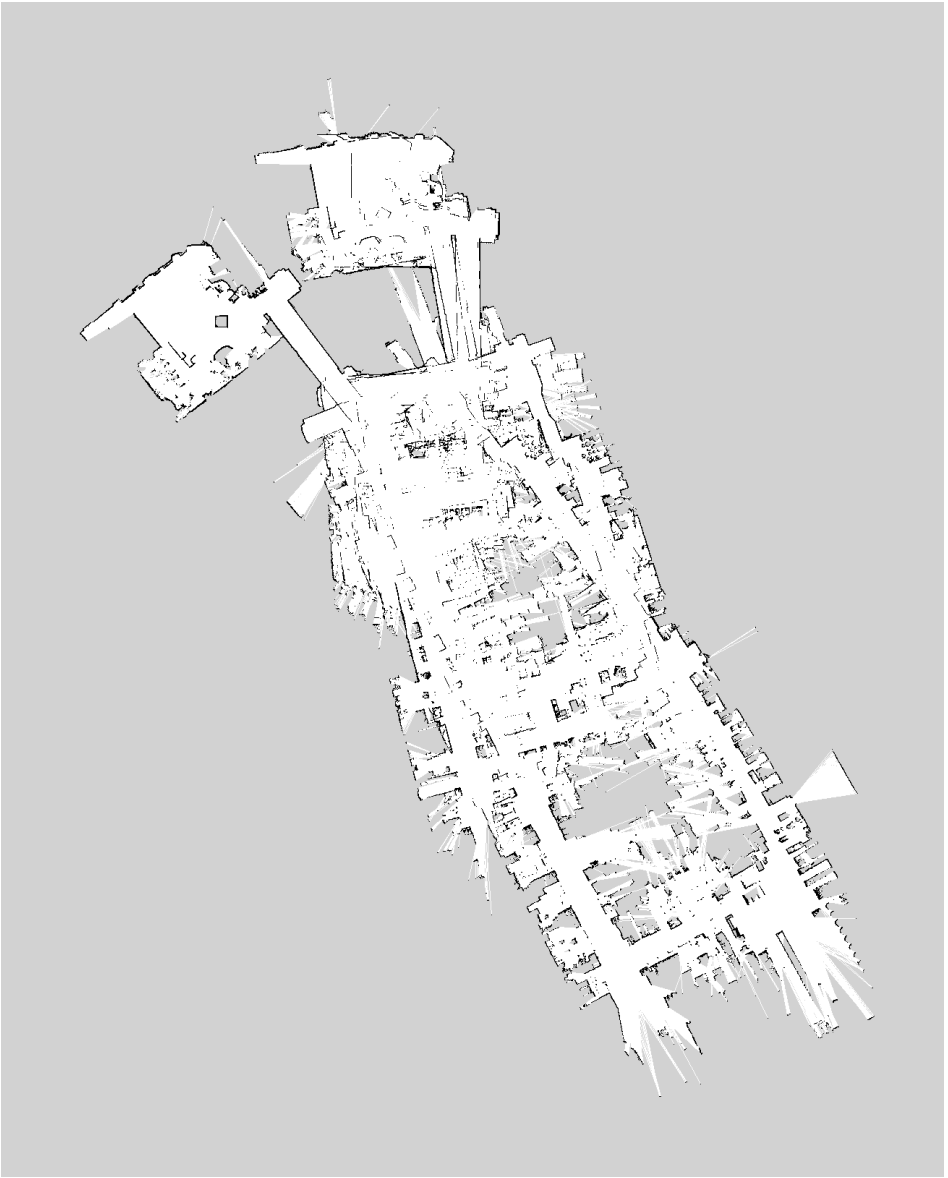
$$\mathbf{p}^* = \text{argmin} \sum_{ji} e_{ji}^T \Omega'_{ji} e_{ji}^T$$

Approaches

- 2D approaches:
  - Lu and Milios, ‘97
  - Montemerlo et al., ‘03
  - Howard et al., ‘03
  - Dellaert et al., ‘03
  - Frese and Duckett, ‘05
  - Olson et al., ‘06
  - Grisetti et al., ’07
  - Tipaldi et al.,’ 07
- 3D approaches:
  - Nuechter et al., ‘05
  - Dellaert et al., ‘05
  - Triebel et al., ’06
  - Grisetti et al., ’08/’09

Graph-Based SLAM in a Nutshell

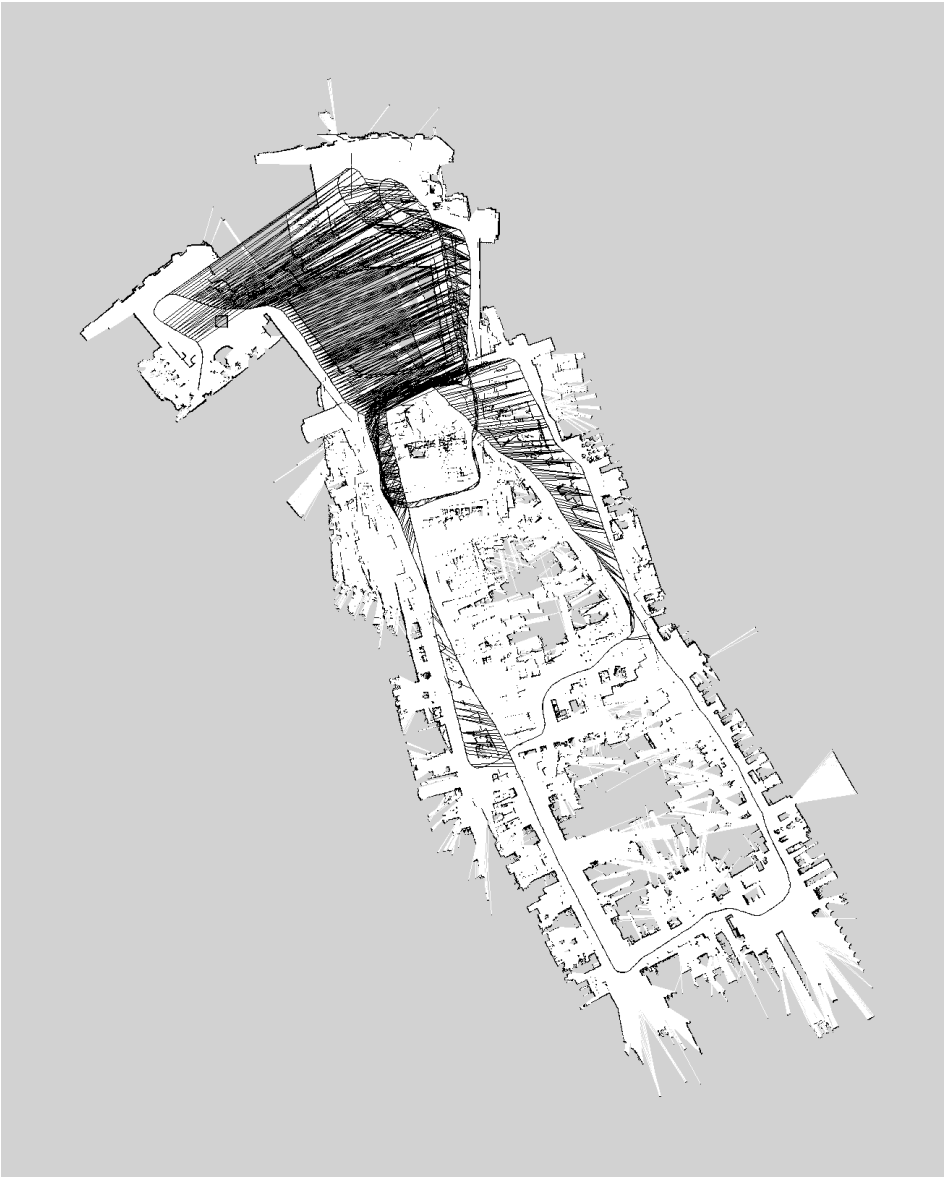
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



[KUKA Hall 22, courtesy P. Pfaff & G. Grisetti]

Graph-Based SLAM in a Nutshell

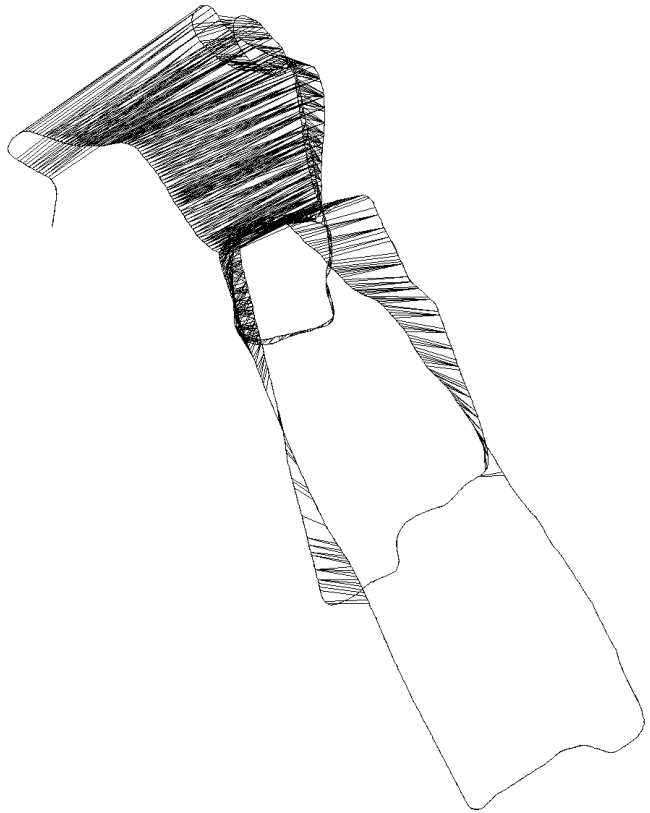
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



[KUKA Hall 22, courtesy P. Pfaff & G. Grisetti]

Graph-Based SLAM in a Nutshell

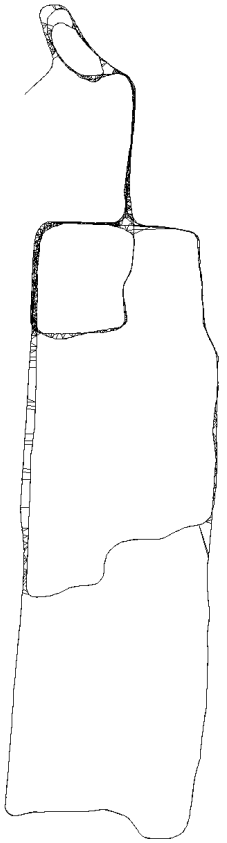
- Once we have the graph, we determine the most likely map by "moving" the nodes



[KUKA Hall 22, courtesy P. Pfaff & G. Grisetti]

Graph-Based SLAM in a Nutshell

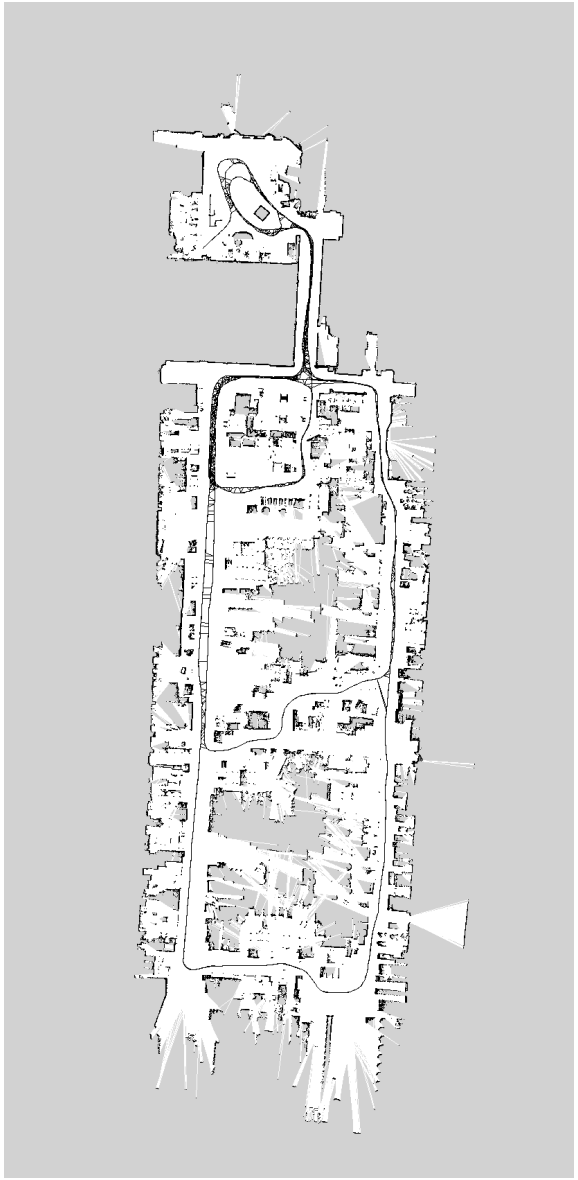
- Once we have the graph, we determine the most likely map by "moving" the nodes
- … like this.



[KUKA Hall 22, courtesy P. Pfaff & G. Grisetti]

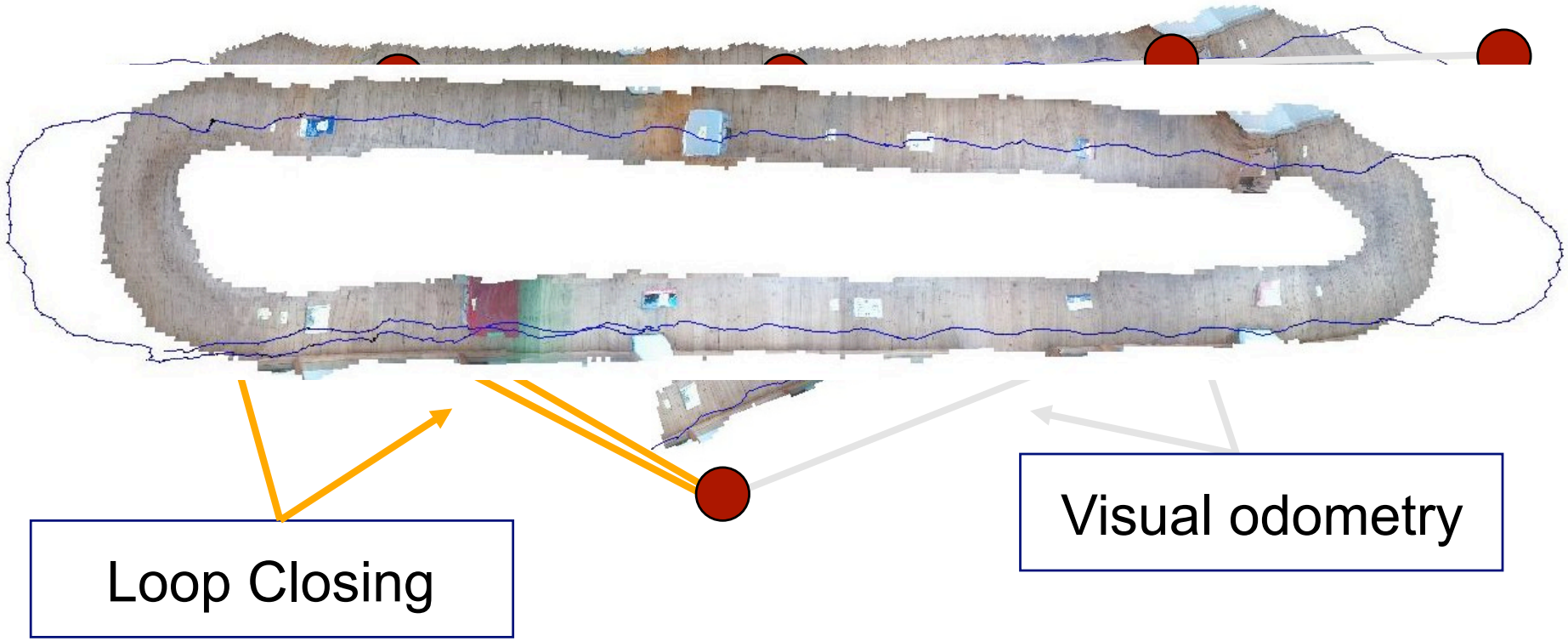
Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by "moving" the nodes
- … like this.
- Then we render a map based on the known poses



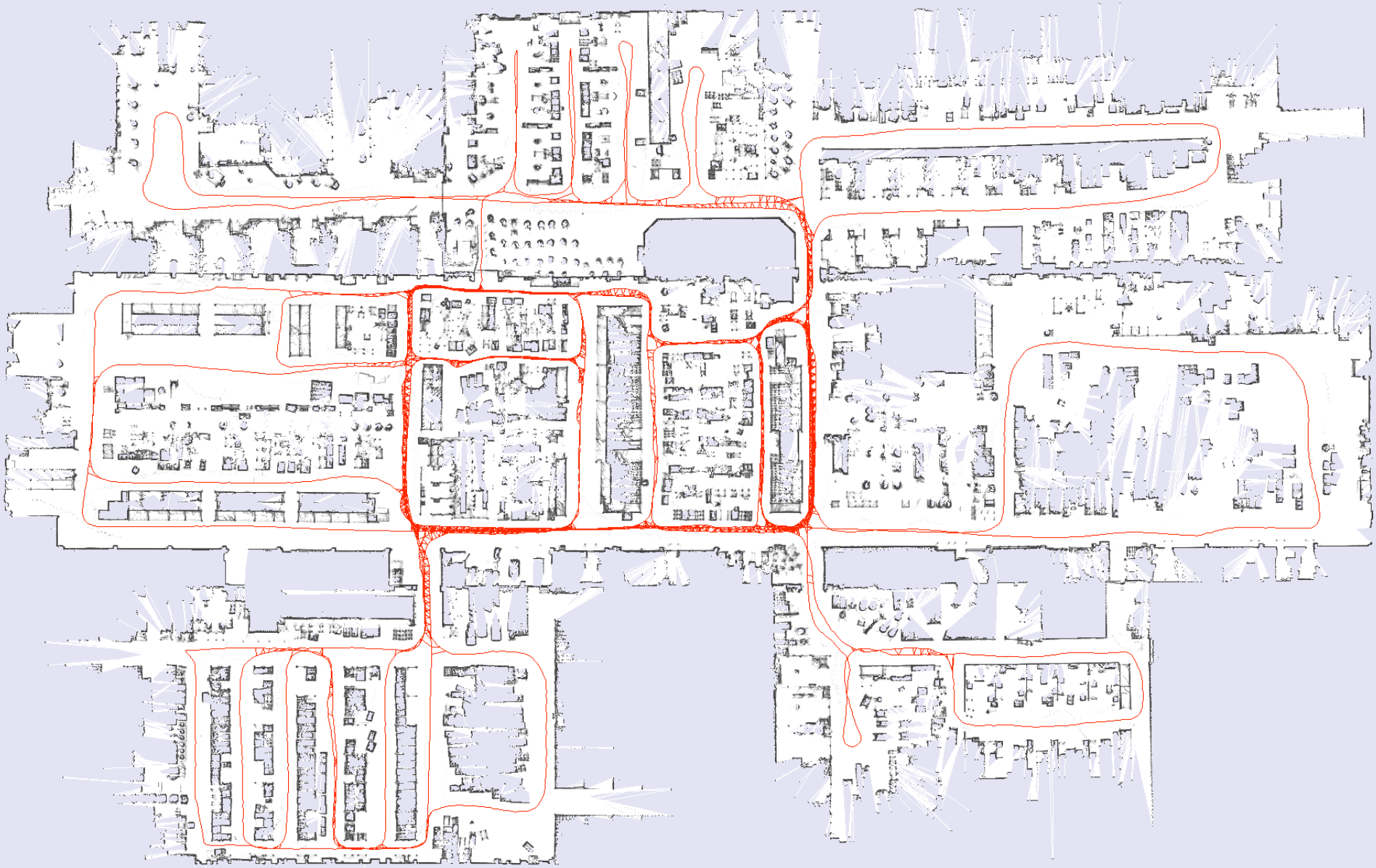
[KUKA Hall 22, courtesy P. Pfaff & G. Grisetti]

Graph-based Visual SLAM



[courtesy B. Steder]

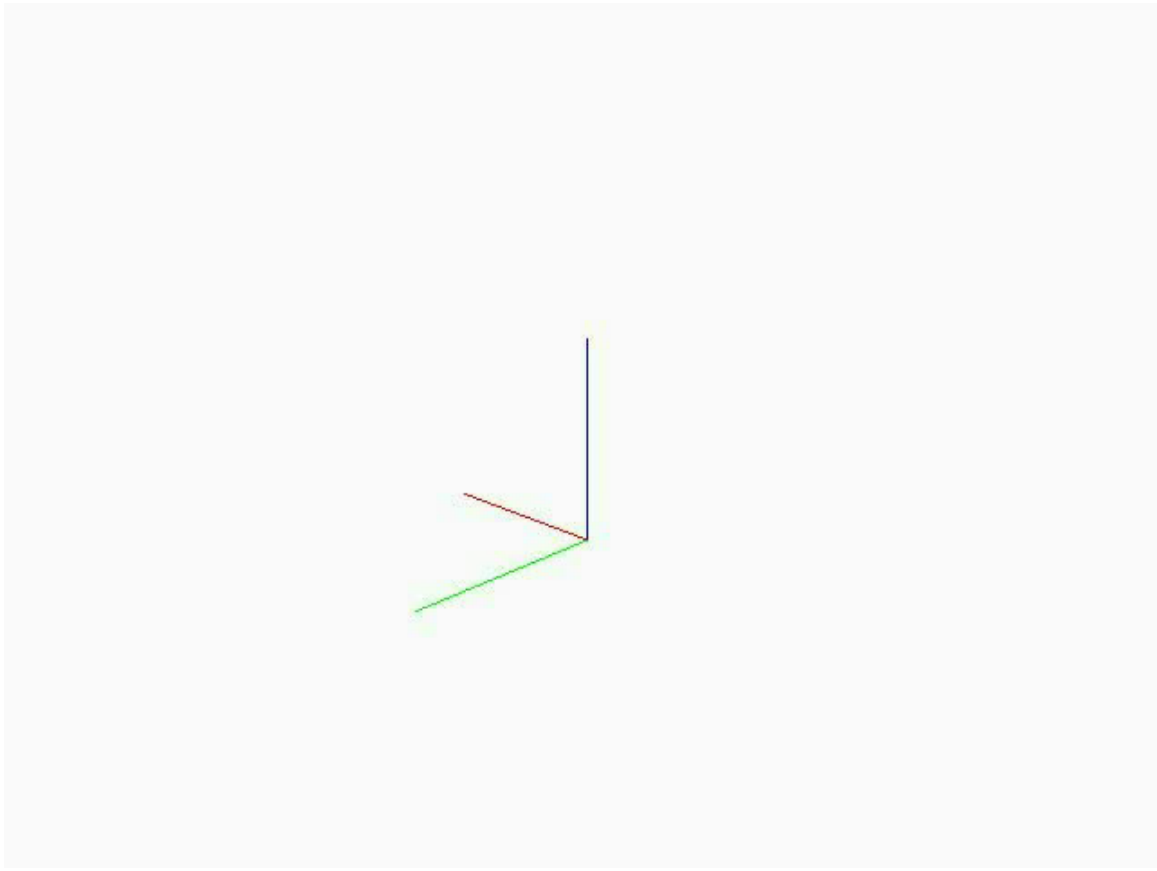
The KUKA Production Site



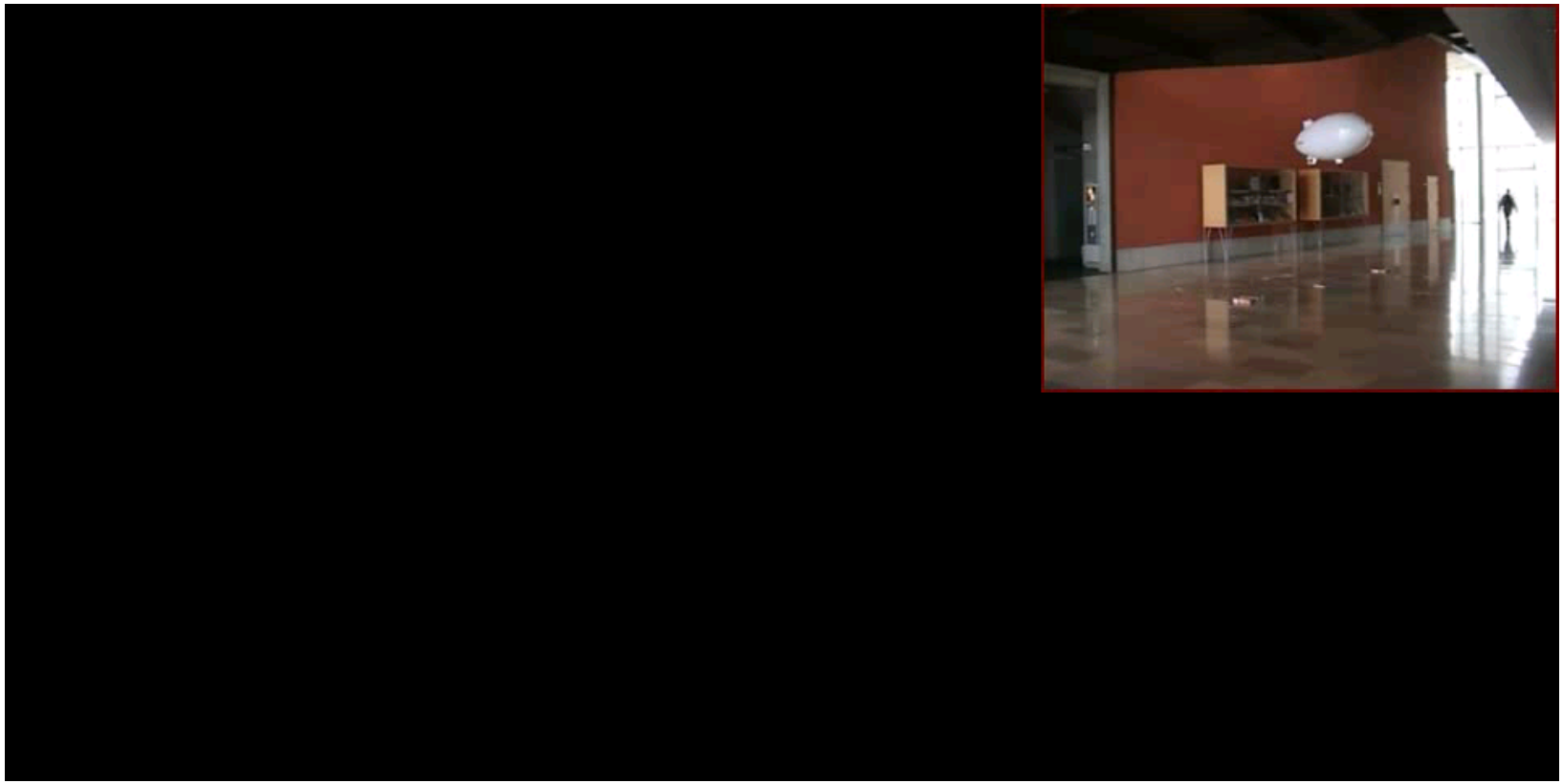
The KUKA Production Site

scans	59668
total acquisition time	4,699.71 seconds
traveled distance	2,587.71 meters
total rotations	262.07 radians
size	180 x 110 meters
processing time	< 30 minutes

Micro-Helicopters

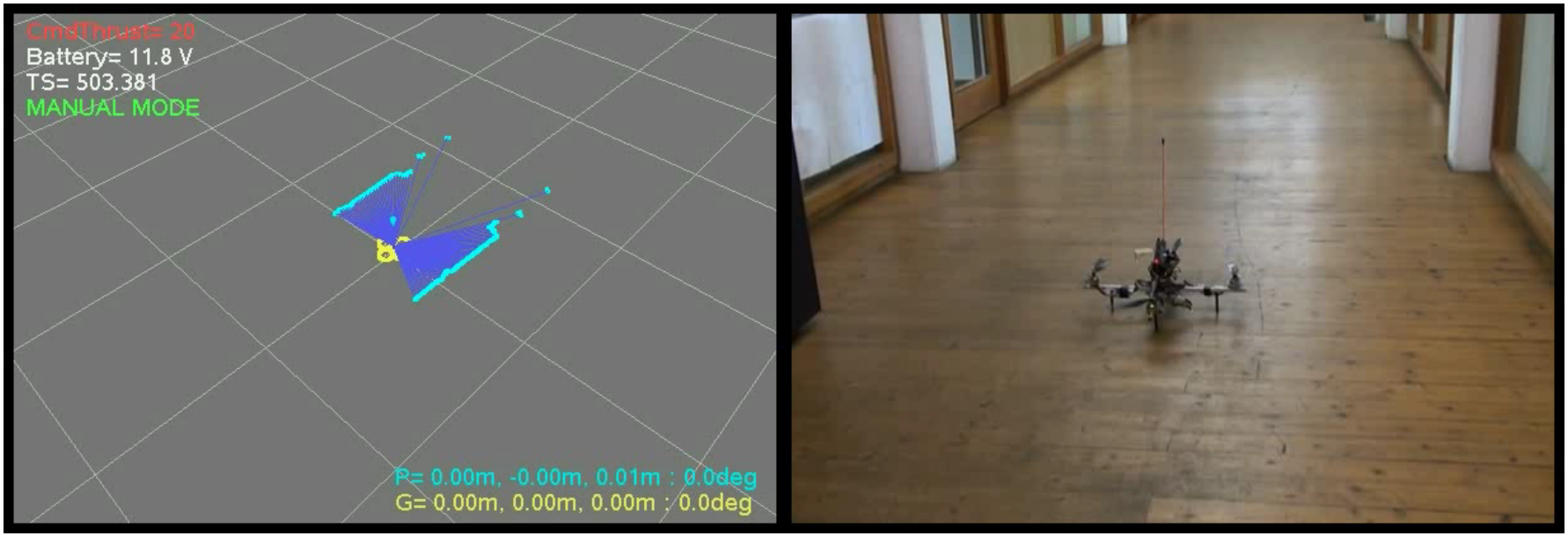


Autonomous Blimp

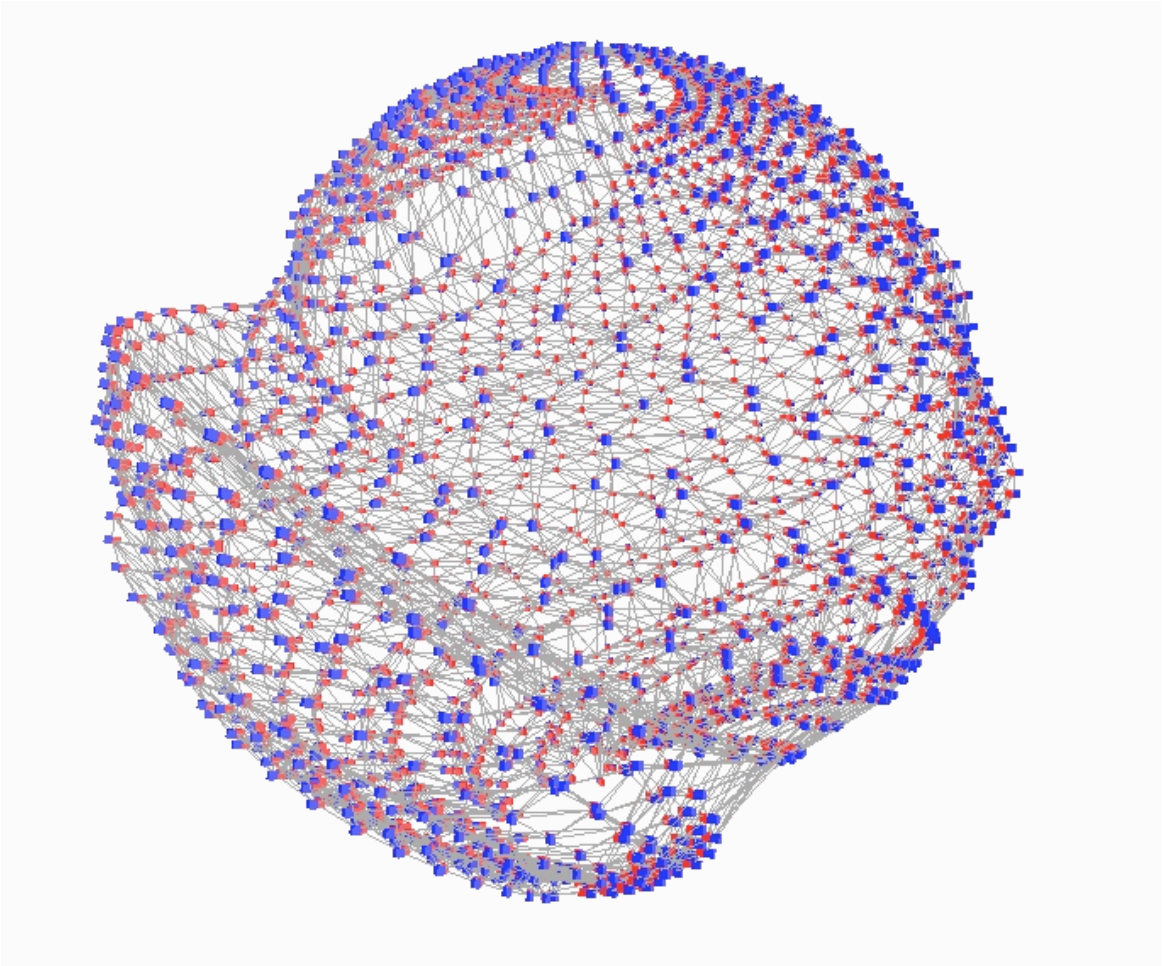


Quadcopter SLAM

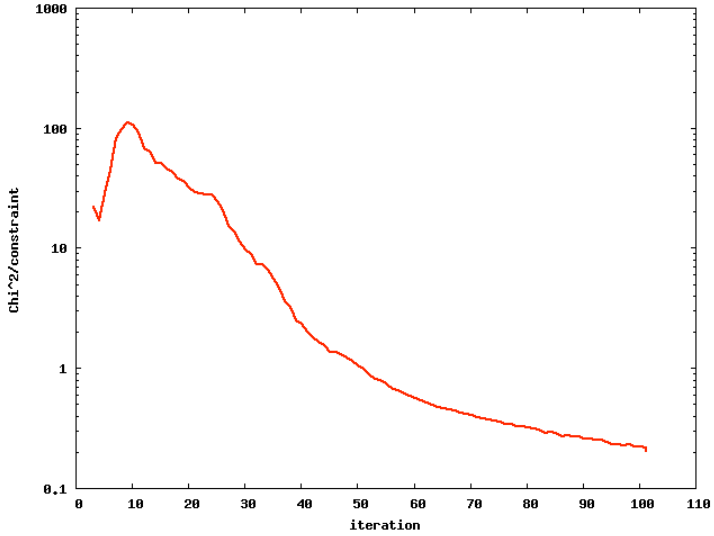
The AIS Quadcopter



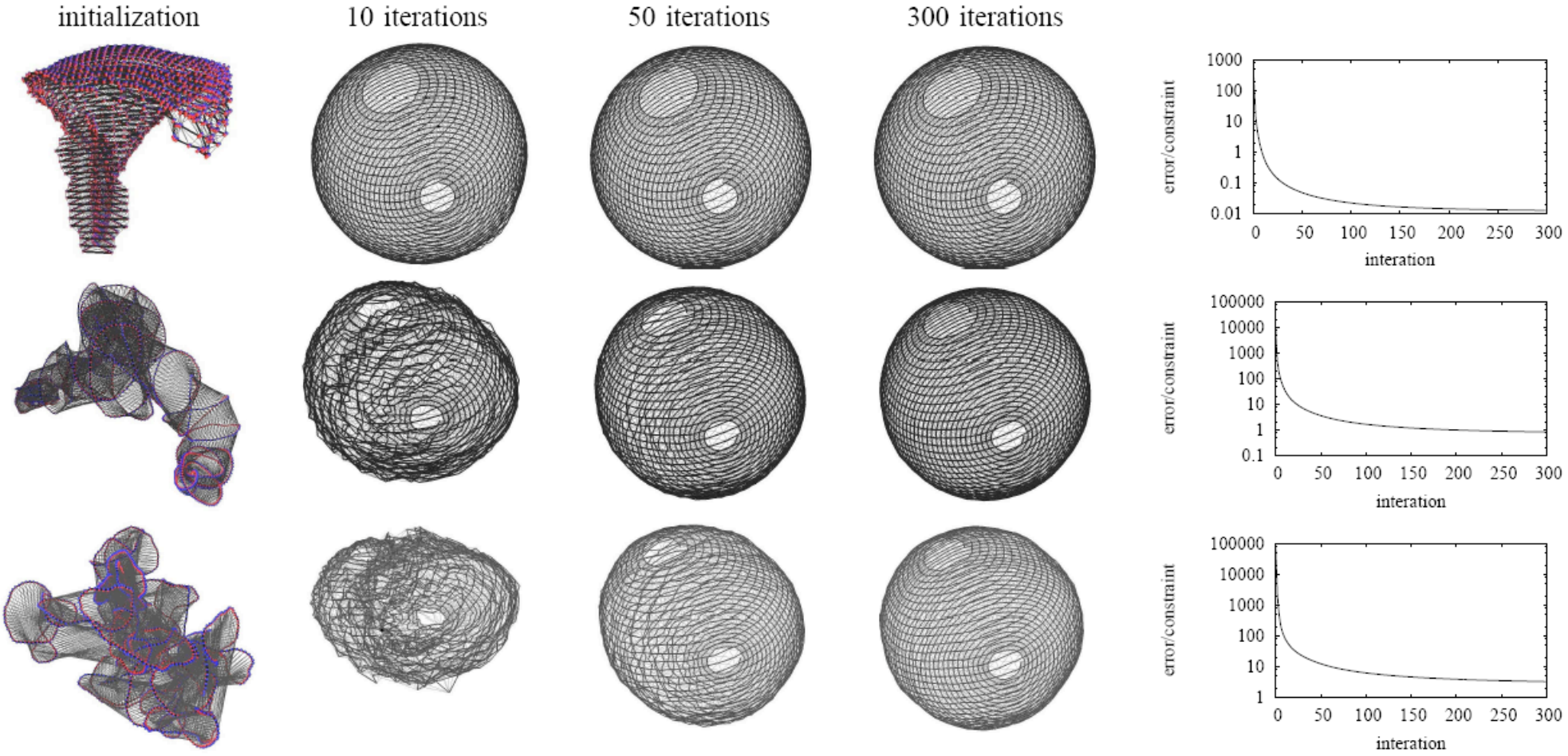
3D Simulated Experiment



- Highly connected graph
- Poor initial guess
- LU & variants fail
- 2200 nodes
- 8600 constraints



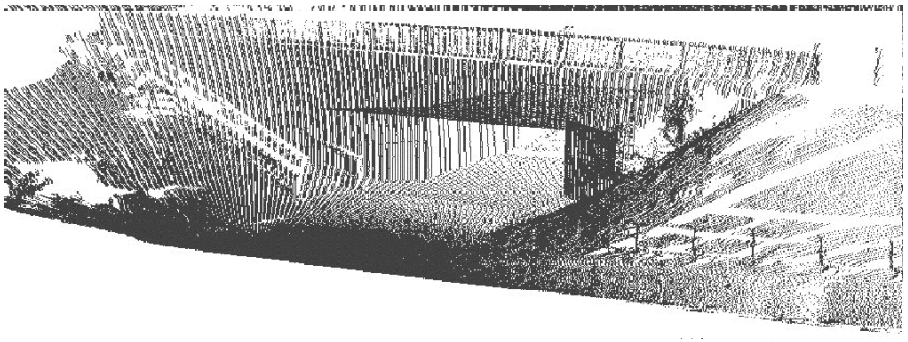
Initialization with Varying Noise



Approximations for 3D Maps

- Surface maps
- Planes
- Gaussian processes

Terrain Maps



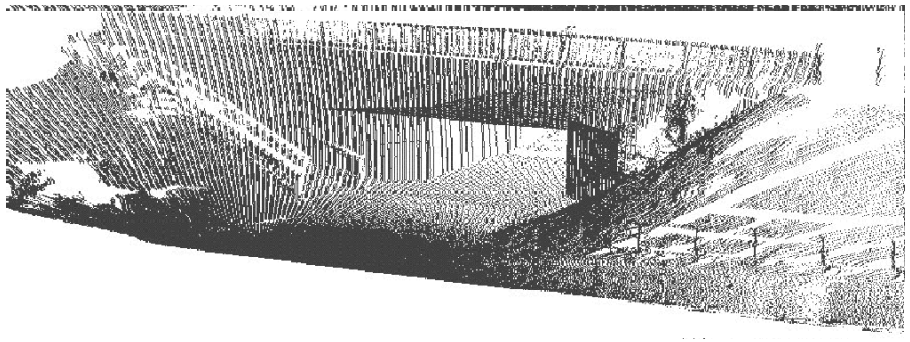
Point cloud

\+ Highly accurate representation

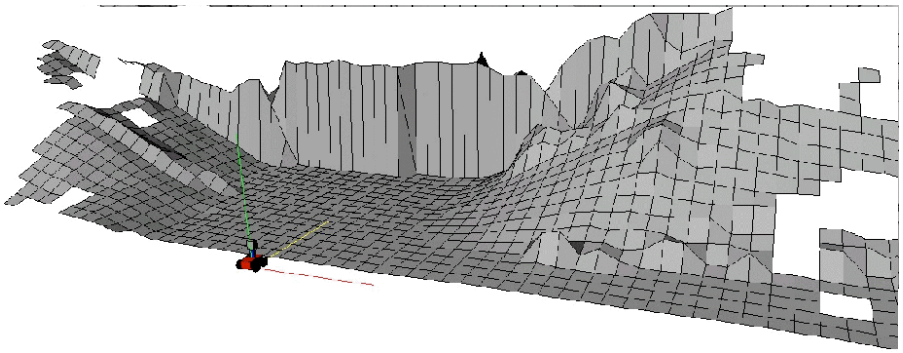
– Planning/navigation hard

– Huge memory requirements

Terrain Maps



Point cloud



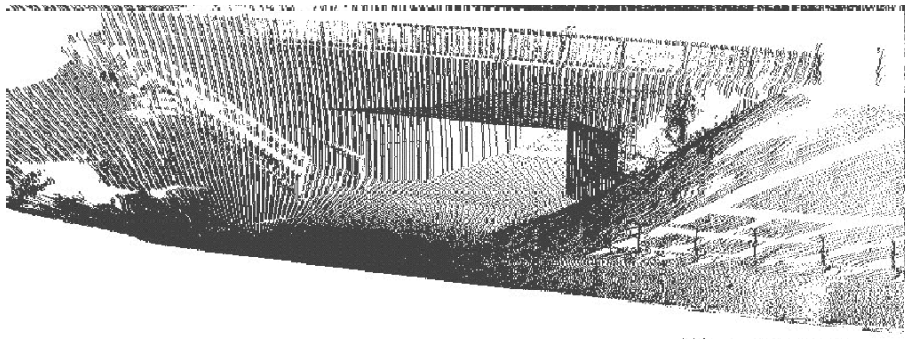
Standard elevation map

\+ Planning and navigation possible (2.5D grid structure)

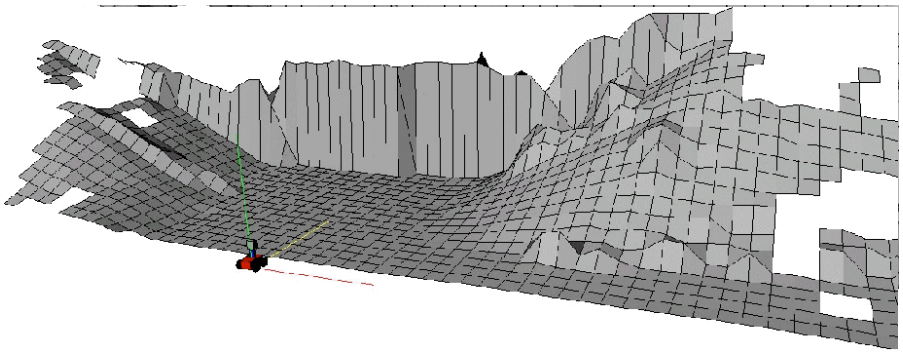
\+ Compact representation

– Bridges appear as big obstacles

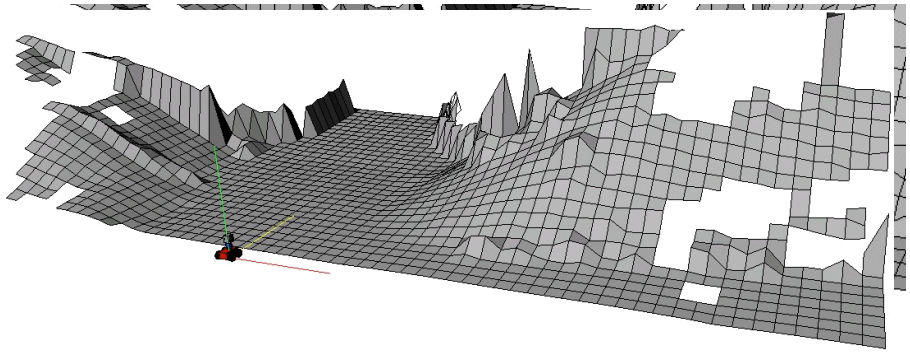
Terrain Maps



Point Cloud



Standard Elevation Map

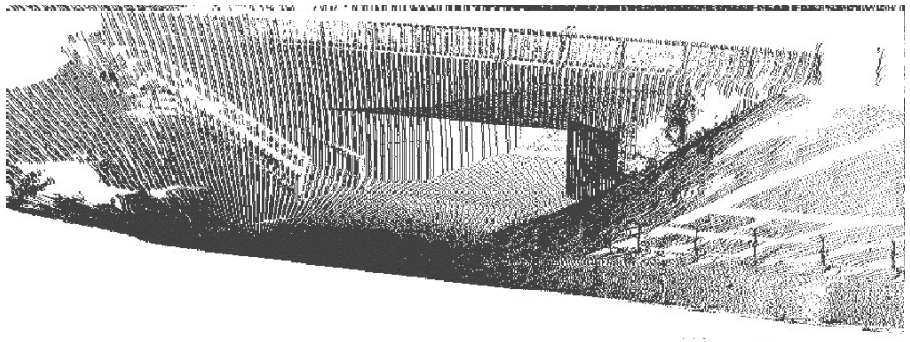


Extended elevation map

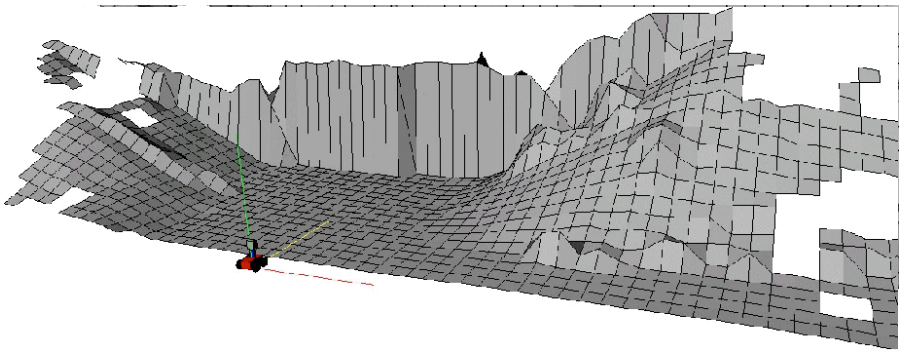
\+ Planning with underpasses possible (cells with vertical gaps)

– No paths *passing under* and *crossing over* bridges possible (only one level per grid cell)

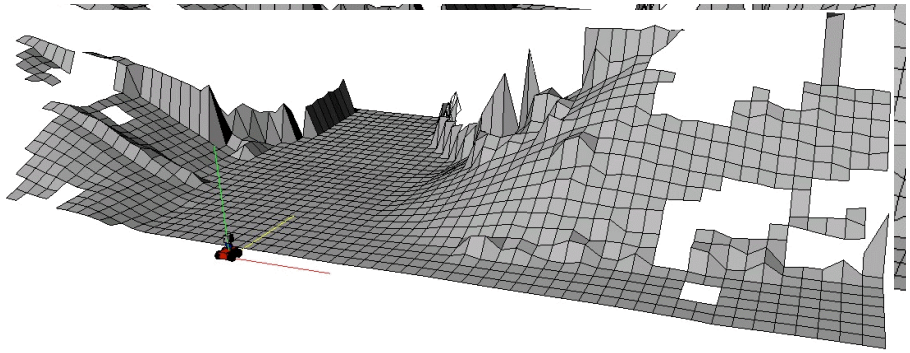
Terrain Maps



Point cloud



Standard elevation map



Extended elevation map

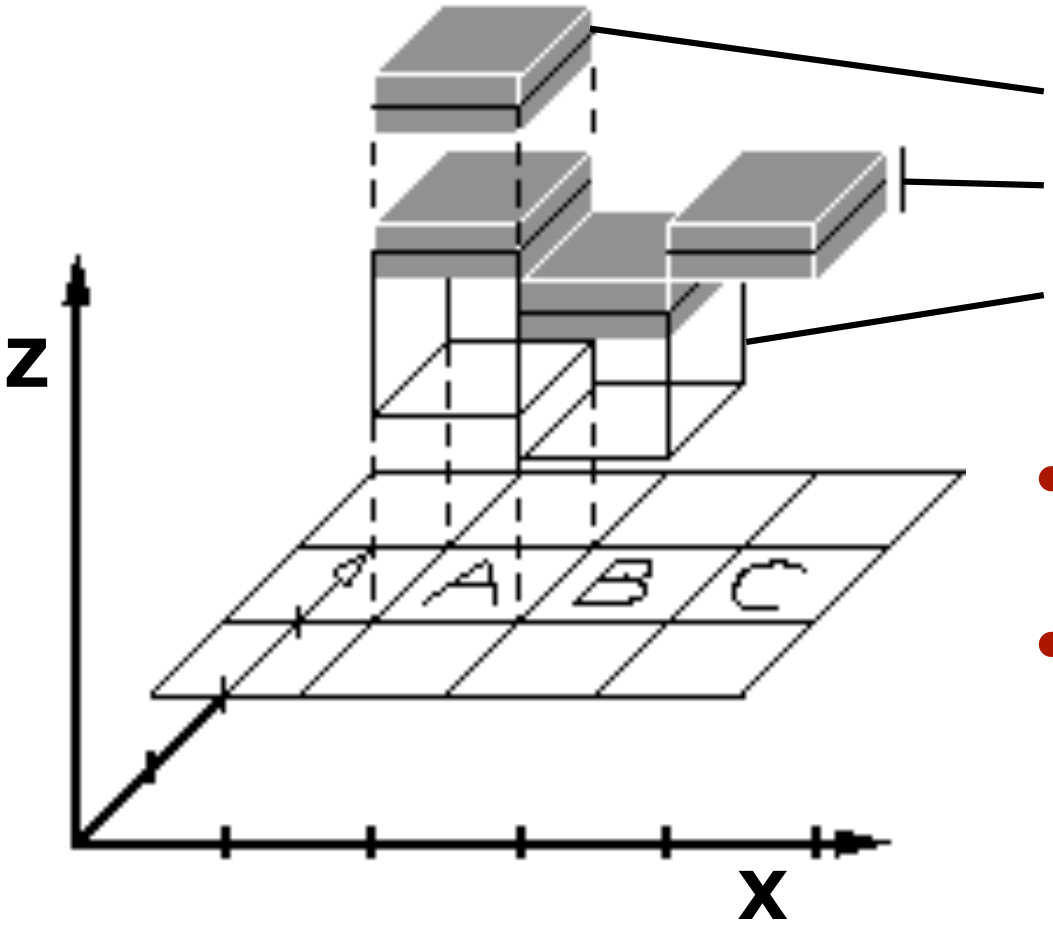
Multi-level surface map

Design Goals

Multi-level surface maps (MLS maps) should

- model uncertainty in height
- represent large-scale data
- represent several height levels
- be updated consistently
- be useful for local map matching (ICP)
- allow us to join them

MLS Map Representation



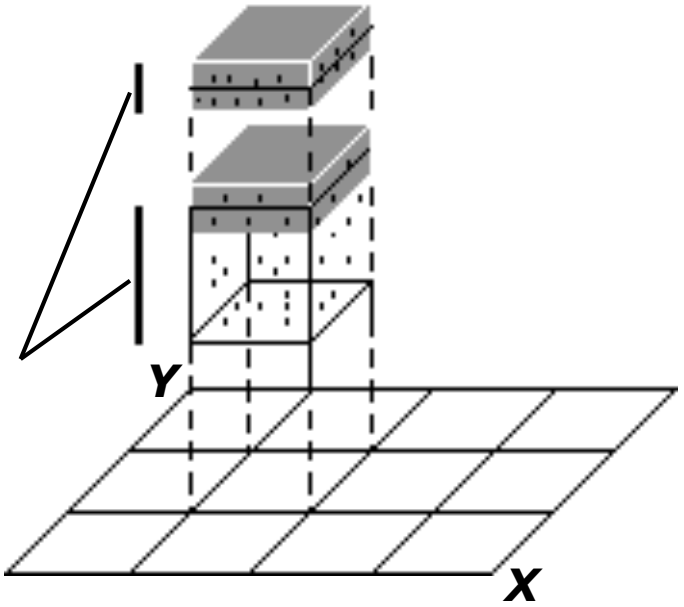
Each 2D *cell* stores various *patches* consisting of:

- A height mean μ
- A height variance σ
- A depth value d

- A *patch* can have no depth (flat objects, e.g., floor)
- A *cell* can have one or many patches (vertical gap cells, e.g., bridges)

From Point Clouds to MLS Maps

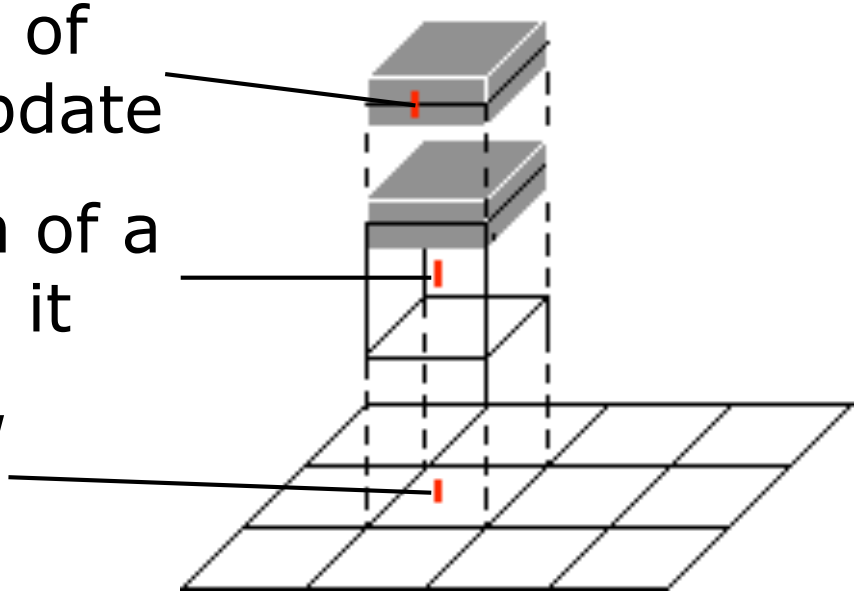
- Map creation:
  - Determine x,y cell for each point
  - Compute vertical *intervals*
  - Classify into *vertical* and *horizontal* intervals



  - Apply Kalman update rule to all measurements in horizontal intervals (patches)
  - Use highest measurement as mean in vertical intervals

Map Update

- Given: new measurement $z=(\boldsymbol{p},\sigma)$ with variance
- Determine the corresponding cell for z
- Find closest surface patch in the cell
- If z is inside 3 variances of the patch, do Kalman update
- If z is in occupied region of a surface patch, disregard it
- Otherwise, create a new surface patch from z



Local Map Matching

- Use Iterative Closest Point (ICP) algorithm
- Use surface patches instead of scan points
- Extract 3 classes of features from each map:
  - *vertical patches*
  - *horizontal and traversable patches*
  - *horizontal and non-traversable patches*

⇒ **Improved data association using features**

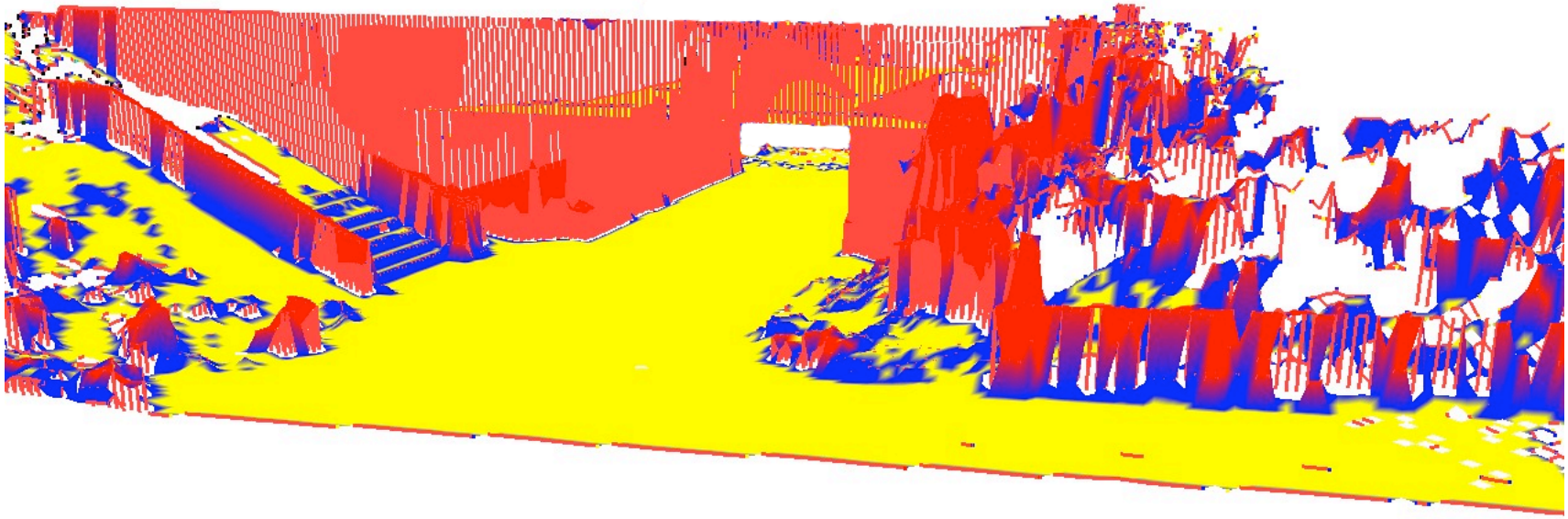
- Sub-sampling of feature classes to reduce complexity

Local Map Matching (Summary)

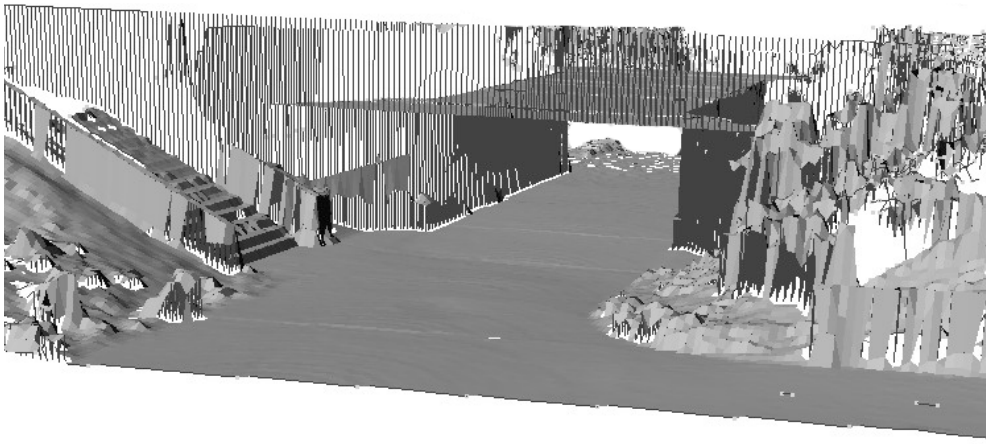
- Classify surface patches into feature classes
- Sub-sample each feature class
- Find rotation *R* and translation *t* that minimize the error function

$$e(R,\mathbf{t}) = \underbrace{\sum_{c=1}^{C_1} d_v(\mathbf{u}_{i_c}, \mathbf{u}'_{j_c})}_{\text{vertical cells}} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{v}_{i_c}, \mathbf{v}'_{j_c})}_{\text{traversable}} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{w}_{i_c}, \mathbf{w}'_{j_c})}_{\text{non-traversable}}$$

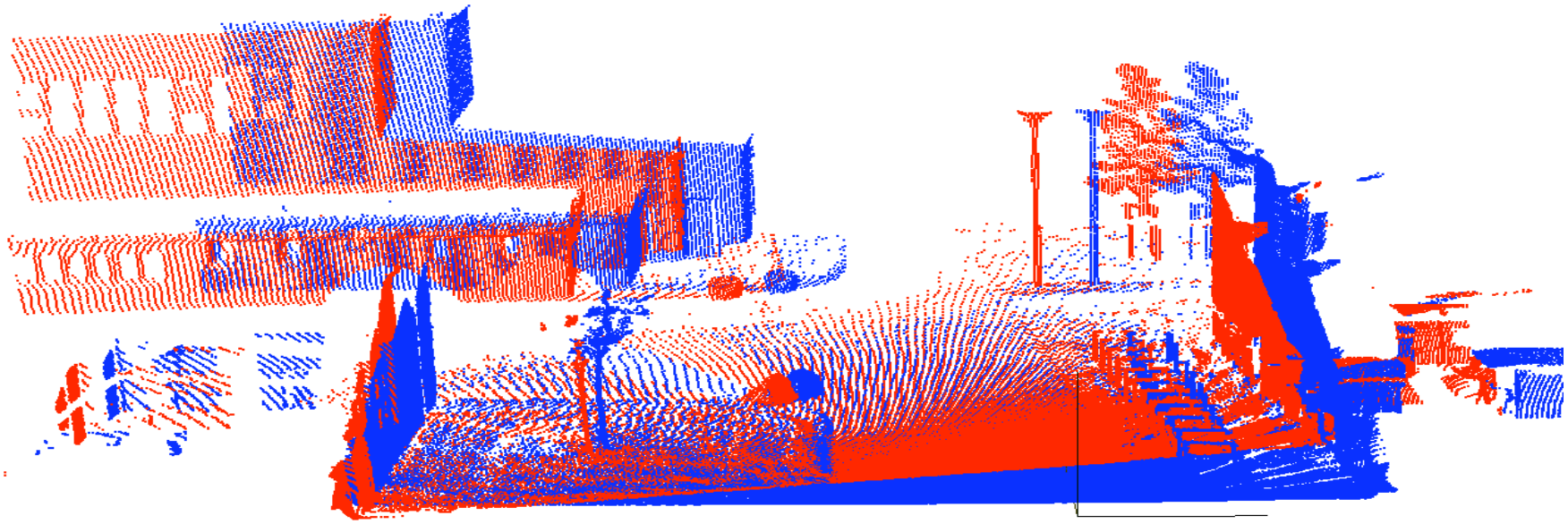
Feature Extraction



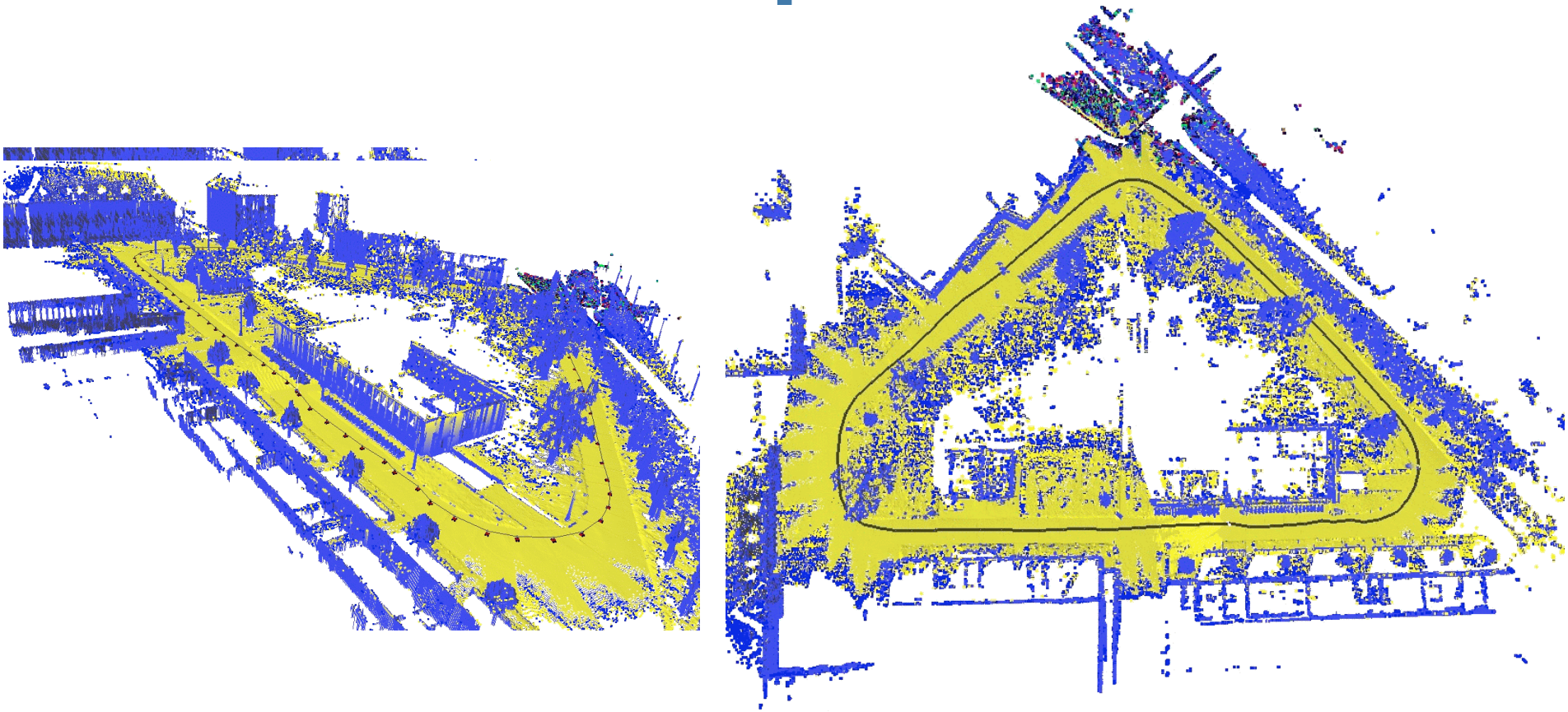
- Traversable Surface Patches
- Cells with vertical objects
- Non traversable Surface Patches



ICP for Elevation Maps

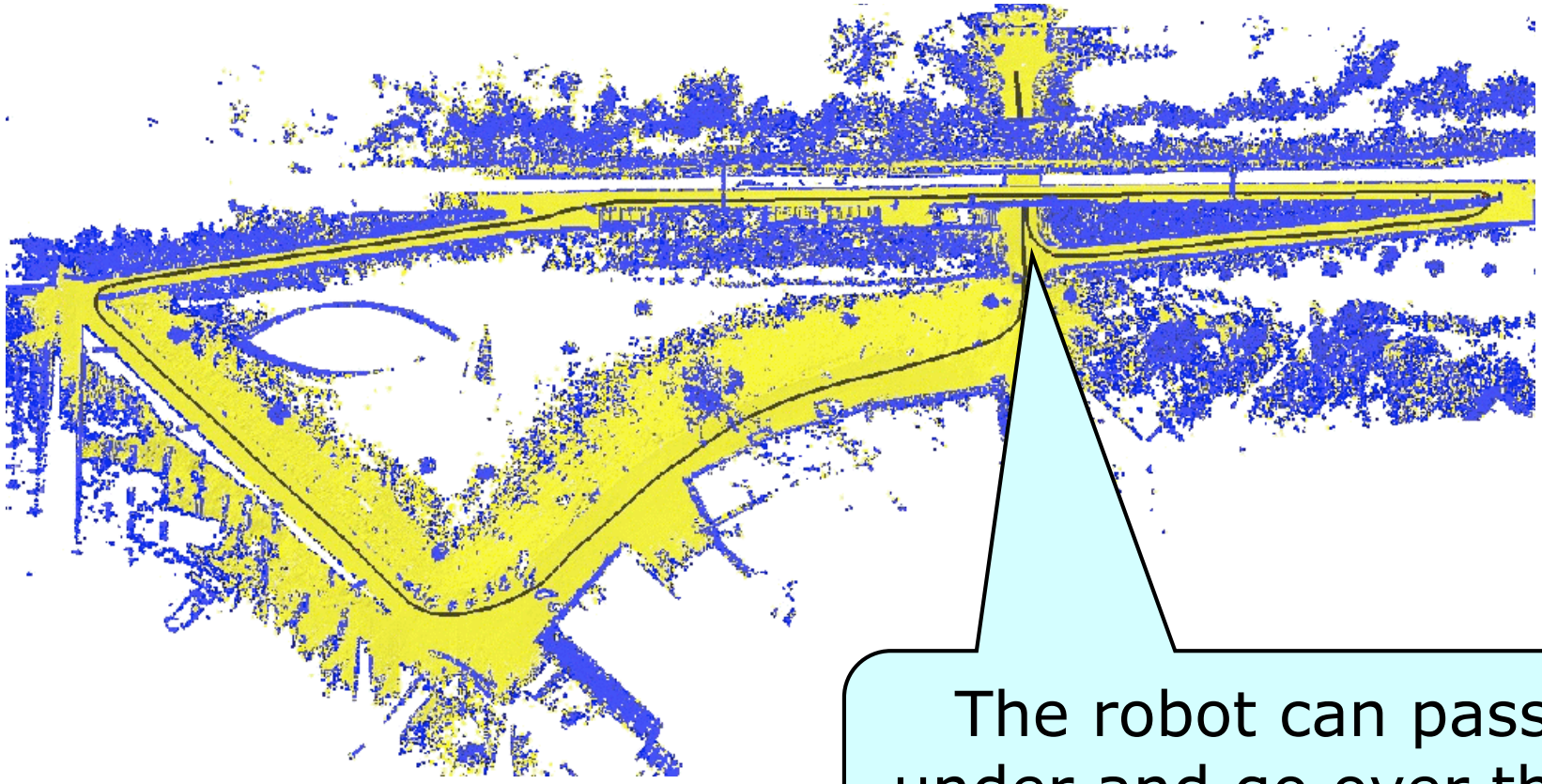


Results after Optimization



- Map size: 195 by 146 *m*
- Cell resolution: 10 *cm*
- Number of data points: 20,207,000

Results



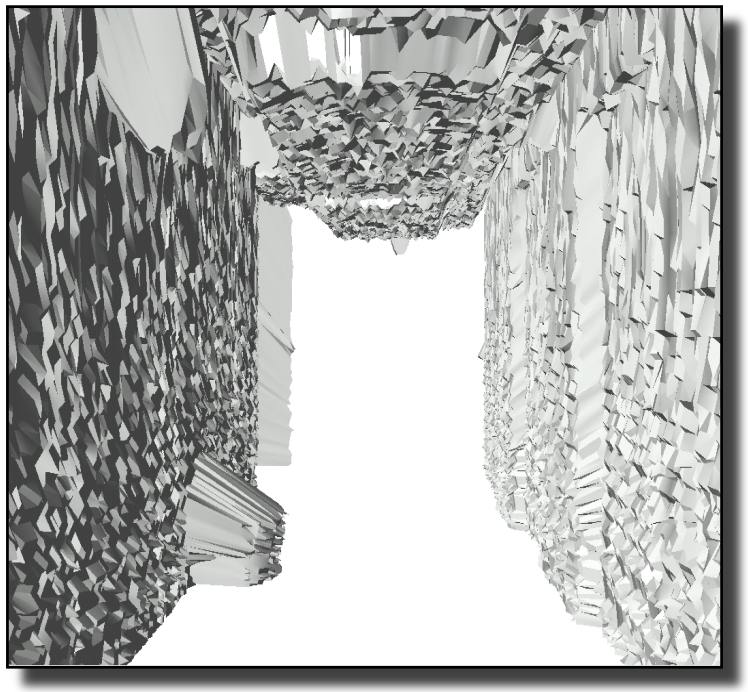
The robot can pass under and go over the bridge

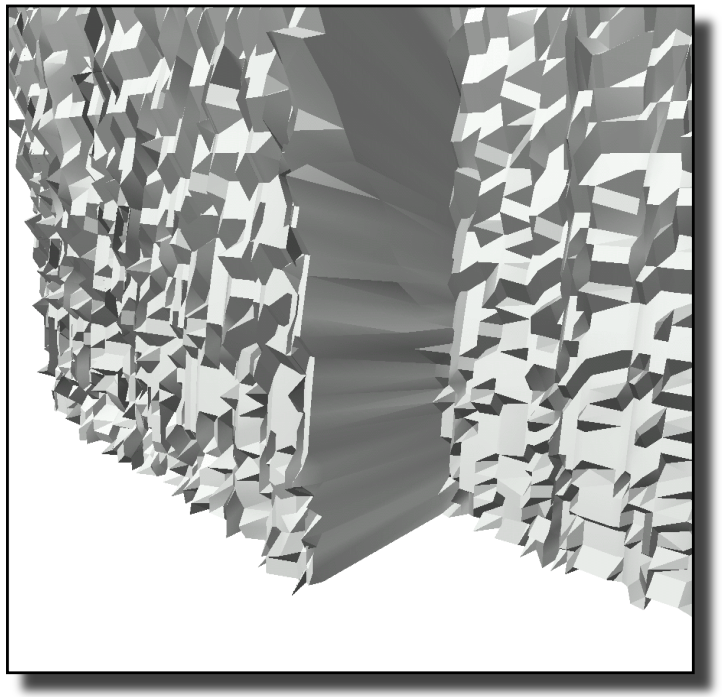
- Map size: 299 by 147 *m*
- Cell resolution: 10 *cm*
- Number of data points: 45,000,000

Resulting Map

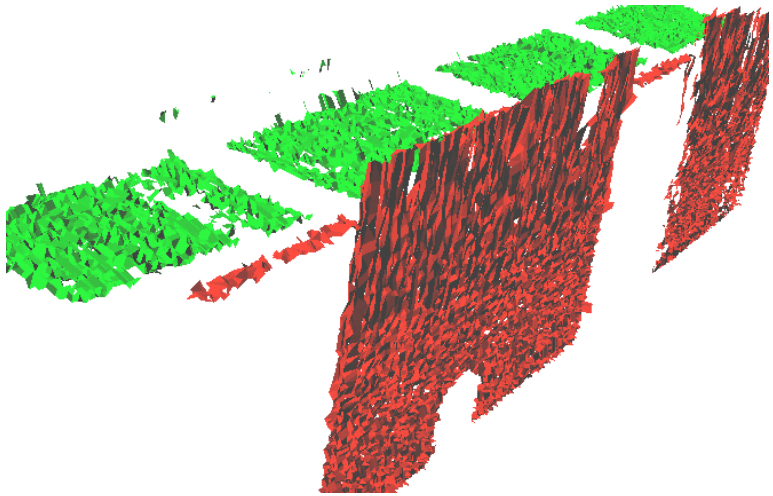
Learning Large Scale MLS Maps with Multiple Nested Loops

Why Planar Approximations?

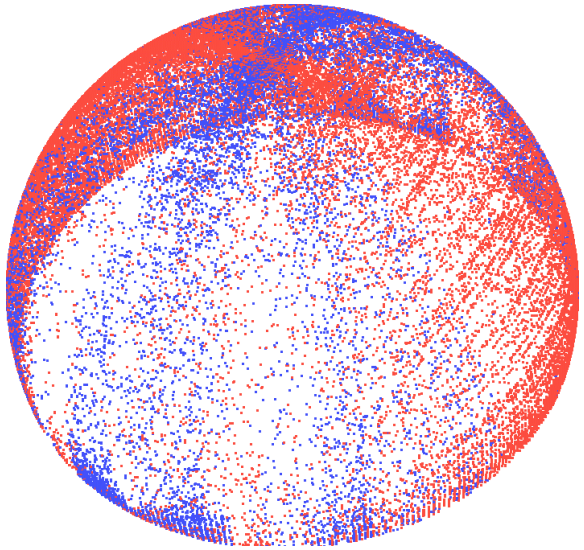




The Surface Normals



Partial data



Normals of triangles

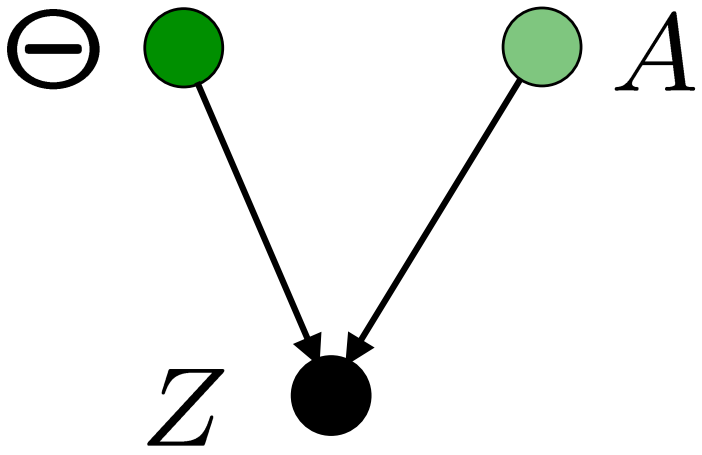
Plane Approximations

- Region growing
  [Besl and Jain ’88, …]
- Hough Transform
  [Okada et al. ’03, Sabe et al. ’04]
- Lines
  [Nuechter et al. ’03]
- Maximum Likelihood Clustering (EM)
  [Liu et al. ’03]

Plane Approximations

- Region growing
  [Besl and Jain ’88, …]
- Hough Transform
  [Okada et al. ’03, Sabe et al. ’04]
- Lines
  [Nuechter et al. ’03]
- Maximum Likelihood Clustering (EM)
  [Liu et al. ’03]

Graphical Model for Standard EM



Z	measurements
$\ominus$	planes
A	correspondences (hidden)

$$\text{JPD} = p(Z|A,\ominus)p(\ominus)p(A)$$

The E-Step in Standard EM

Calculate the expectations over the hidden variables, i.e., the probability that a particular point n belongs to a plane m:

$$p(A, \Theta, Z) \propto \exp\left\{-\frac{1}{2}\sum_n \sum_m \alpha_{nm}\left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho}\right)^2\right\}$$

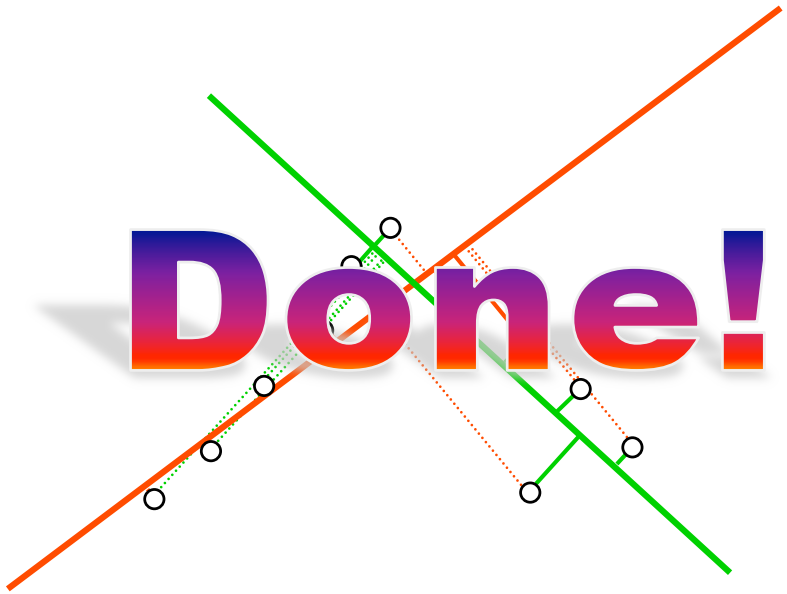
$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_m)}{\rho}\right)^2\right\}}{\sum_j \exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_j)}{\rho}\right)^2\right\}}$$

M-Step for Standard EM

- Calculate the plane parameters
  - according to the data points
  - taking into account the data associations

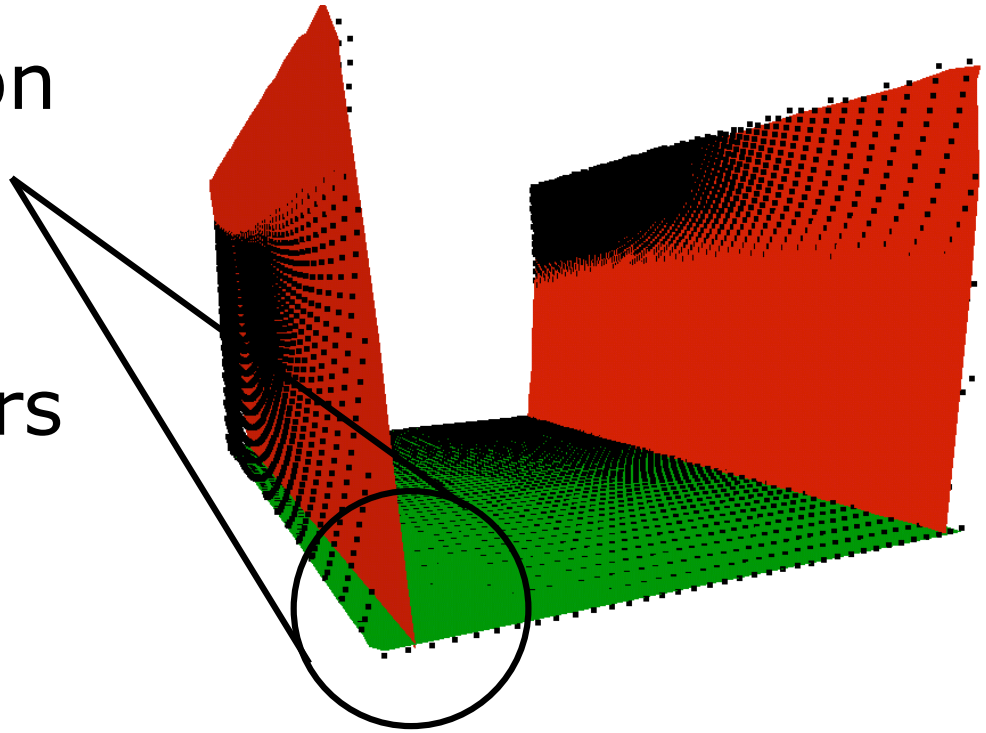
- Can be done in closed form.

EM Example



Typical Problem in Standard EM

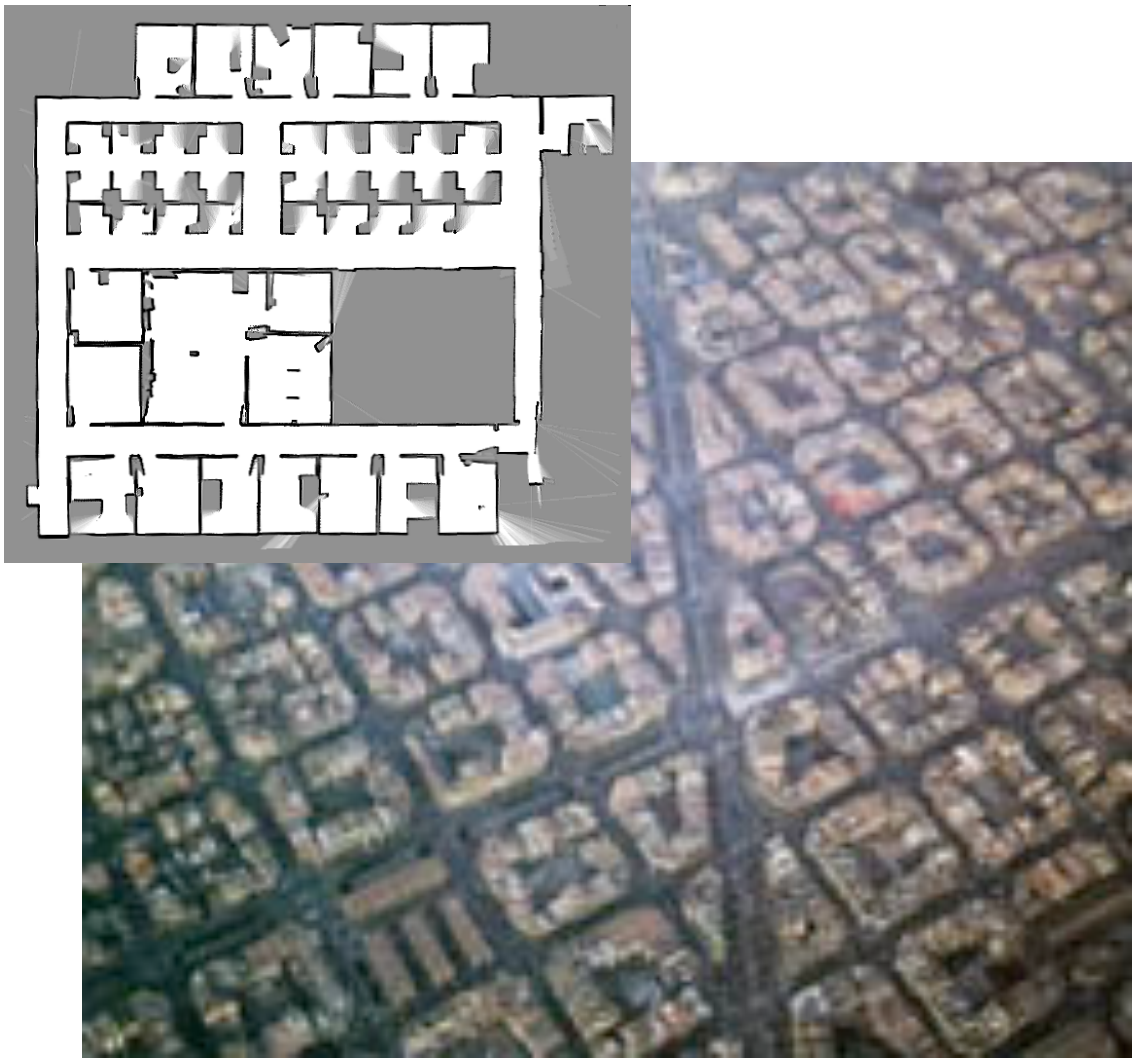
- Data association uncertainty
- results in alignment errors in M-step



Idea: Incorporate higher-level knowledge about planes (parallelism)

$\Rightarrow$ **Hierarchical EM**

Man-made Environments



- Similar structures
- Many planes are parallel
- Can be utilized during mapping

Hierarchical Model

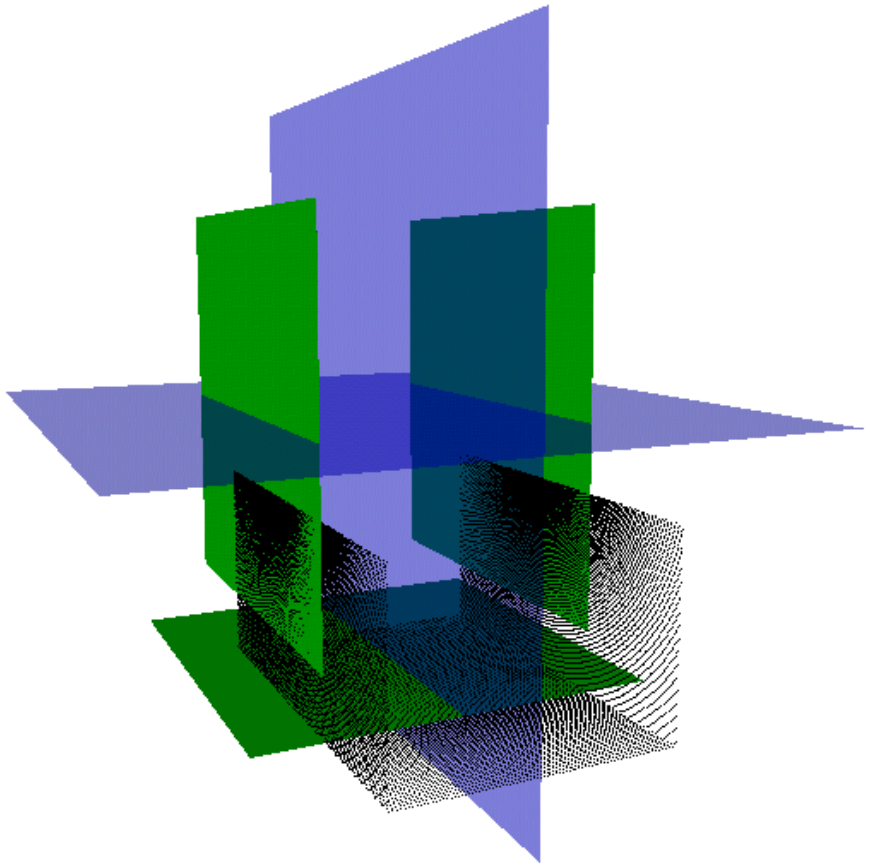
Idea: **simultaneously**

- cluster points into planes and
- planes into *main directions* to

- obtain the *most likely* set of planes and main directions.

Maximum Likelihood Estimation

Example



84,660 points

$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{84,660}$

3 planes

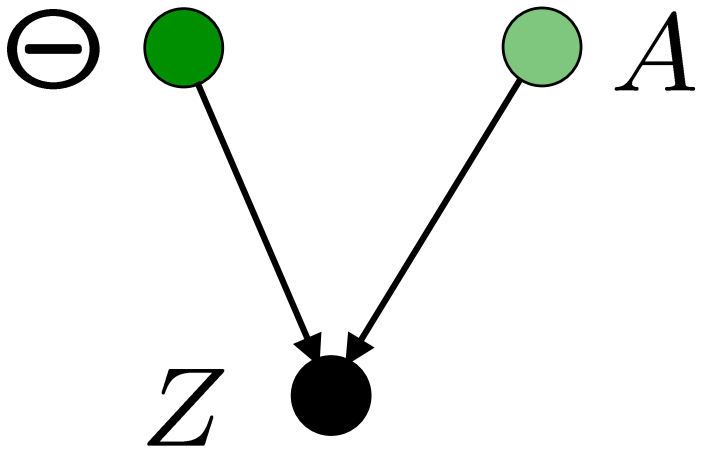
$\theta_1, \theta_2, \theta_3$

2 main directions

Φ_1, Φ_2

Main directions are normal vectors of planes.

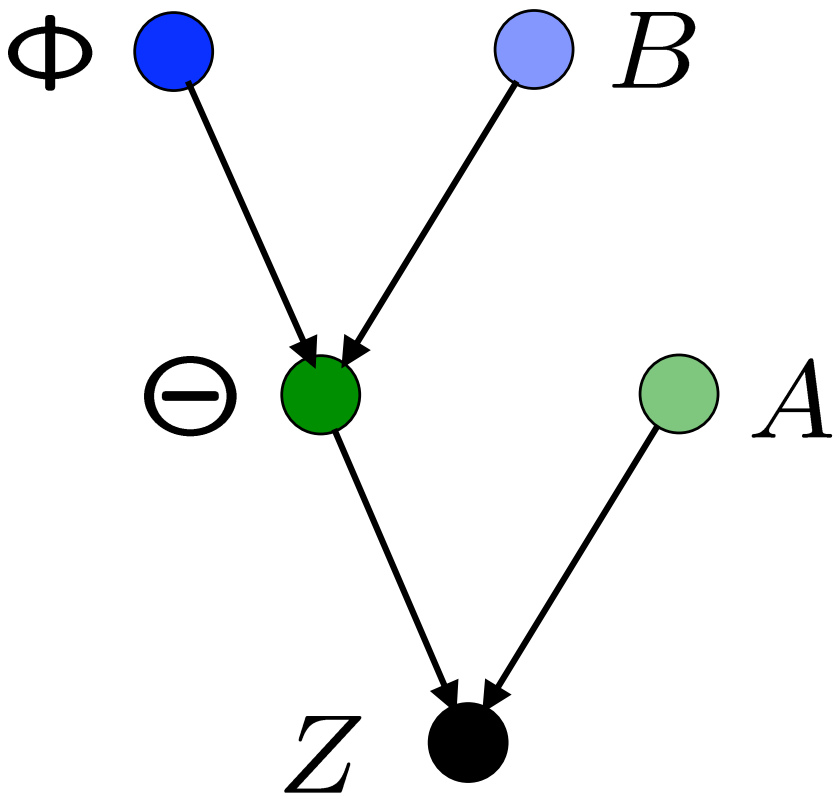
Graphical Model for Standard EM



Z	measurements
$\ominus$	planes
A	correspondences (hidden)

$$\text{JPD} = p(Z|A,\ominus)p(\ominus)p(A)$$

Graphical Model for Hier. EM



Z	measurements
Θ	planes
A	correspondences (hidden)
B	correspondences (hidden)
Φ	main directions

$$\text{JPD} = p(Z|A,\Theta)p(\Theta|B,\Phi)p(\Phi)p(A)p(B)$$

Expectation Maximization

- Goal: Find Θ and Φ so that JPD is maximized
- Take expectation over hidden variables
- Start with initial planes and main directions
- Iterate until convergence:

$$(\Theta^{[i+1]}, \Phi^{[i+1]}) =$$
$$\text{argmax}_{(\Theta,\Phi)} E_{AB}\left[\log p(A, B, \Phi, \Theta, Z) \mid \Theta^{[i]}, \Phi^{[i]}\right]$$

The E-Step in Hierarchical EM

$$p(A, B, \Phi, \Theta, Z) \propto \exp\left\{ -\frac{1}{2} \sum_n \sum_m \alpha_{nm} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 - \frac{1}{2} \sum_m \sum_k \beta_{mk} \left(\frac{d_2(\theta_m, \varphi_k)}{\sigma} \right)^2 \right\}$$

$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho} \right)^2 \right\}}{\sum_j \exp\left\{ -\frac{1}{2} \left(\frac{d_1(\mathbf{z}_n, \theta_j)}{\rho} \right)^2 \right\}}$$

$$E[\beta_{mk} \mid \Phi] = \frac{\exp\left\{ -\frac{1}{2} \left(\frac{d_2(\theta_m, \varphi_k)}{\sigma} \right)^2 \right\}}{\sum_j \exp\left\{ -\frac{1}{2} \left(\frac{d_2(\theta_j, \varphi_k)}{\sigma} \right)^2 \right\}}$$

The E-Step in Hierarchical EM

$$p(A, B, \Phi, \Theta, Z) \propto \exp\left\{-\frac{1}{2}\sum_n \sum_m \alpha_{nm}\left(\frac{d_1(\mathbf{z}_n, \theta_m)}{\rho}\right)^2 \right\}$$

$$E[\alpha_{nm} \mid \Theta] = \frac{\exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_m)}{\rho}\right)^2\right\}}{\sum_j \exp\left\{-\frac{1}{2}\left(\frac{d_1(\mathbf{z}_n,\theta_j)}{\rho}\right)^2\right\}}$$

The M-Step

- Find planes and main directions so that the expected log-likelihood is maximized
- No closed-form solution (LL is non-linear)
- Optimization technique: conjugate gradient based method

Model Complexity

- Problem: number of planes / main directions not known
- Evaluate model complexity using Bayesian Information Criterion (BIC):

$$BIC = -2L + (3M + 2K)\ln(N)$$

Choose model with minimum BIC

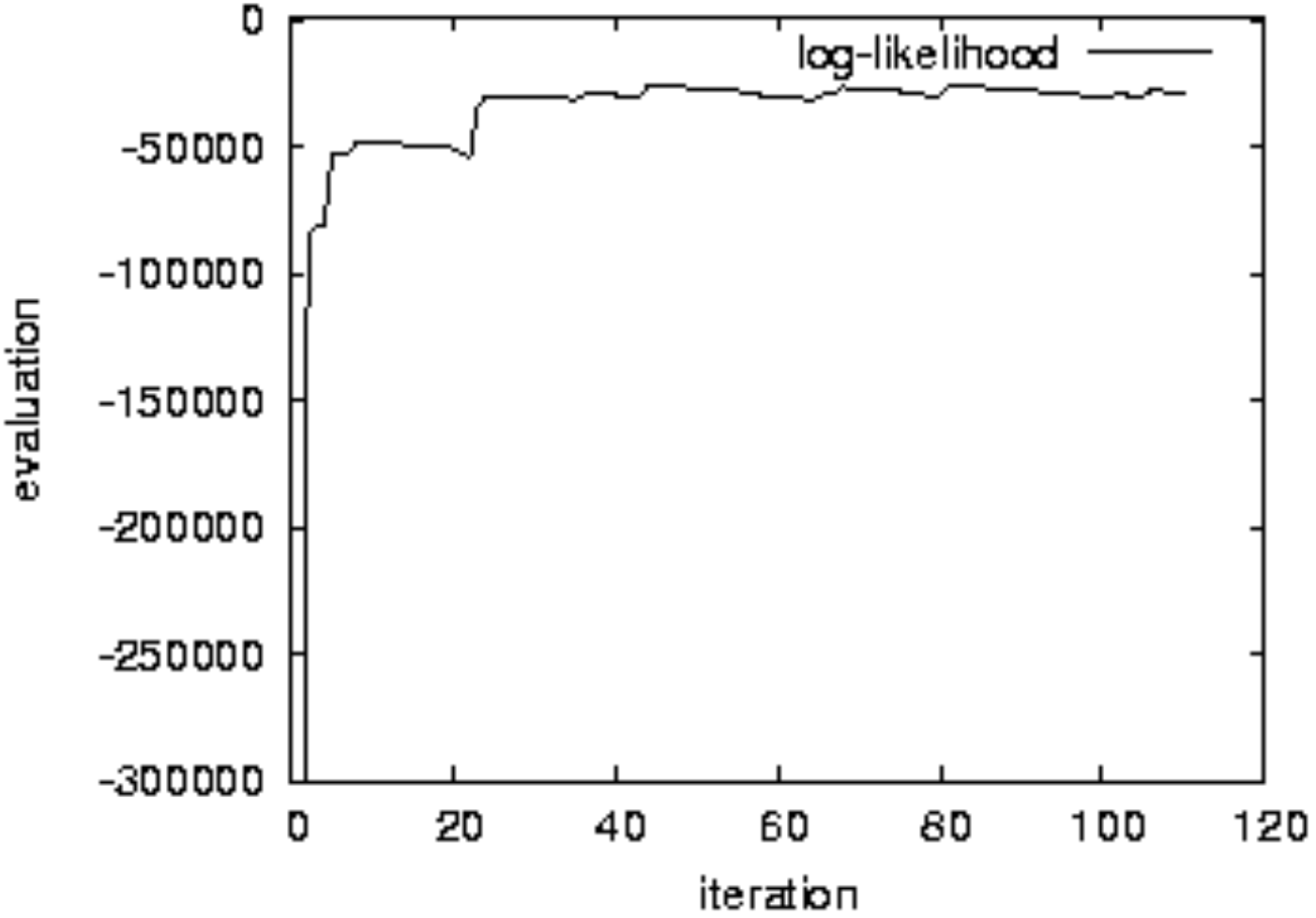
Overall Algorithm

1. Start with initial Θ and Φ
2. Run EM until convergence
3. While BIC decreases:
   a) Remove one plane and calculate new BIC
   b) Remove one main direction and calculate new BIC
4. If no change stop, else:
   a) Insert new plane and main direction
   b) Go to 2.

Post Processing

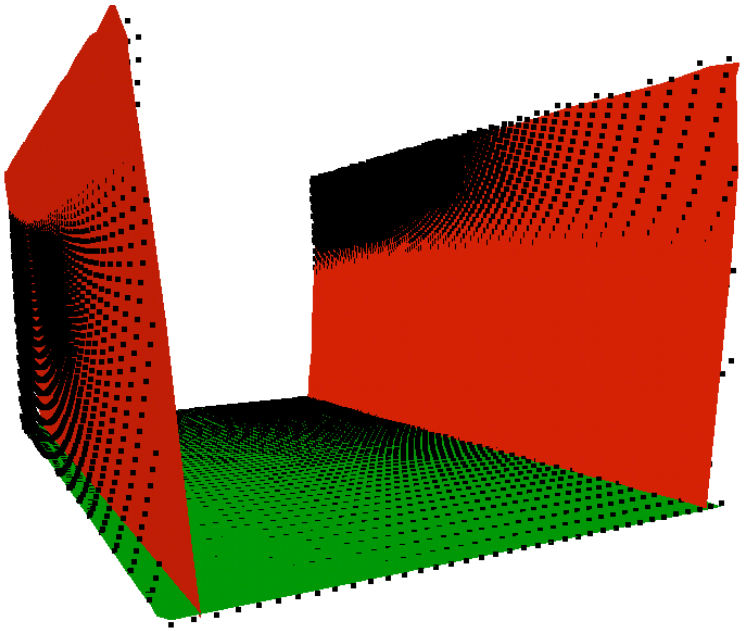
- Project points belonging to a plane into the plane
- Cluster points into polygonal areas (region-growing)
- Find 2D contours for each cluster (alpha-shapes)
- Constraint: polygonal areas should not contain holes

Real Data Experiment (2)

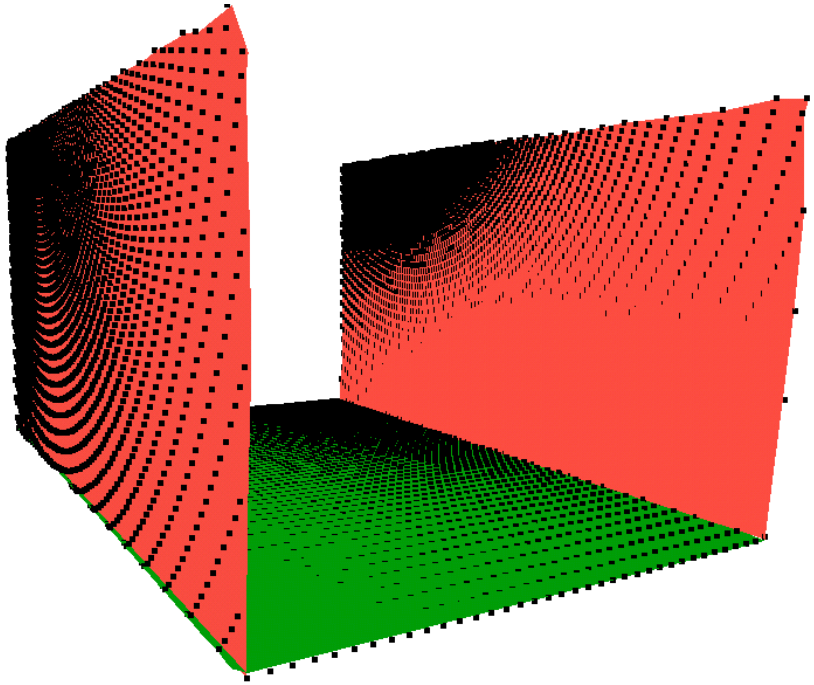


Typical Result

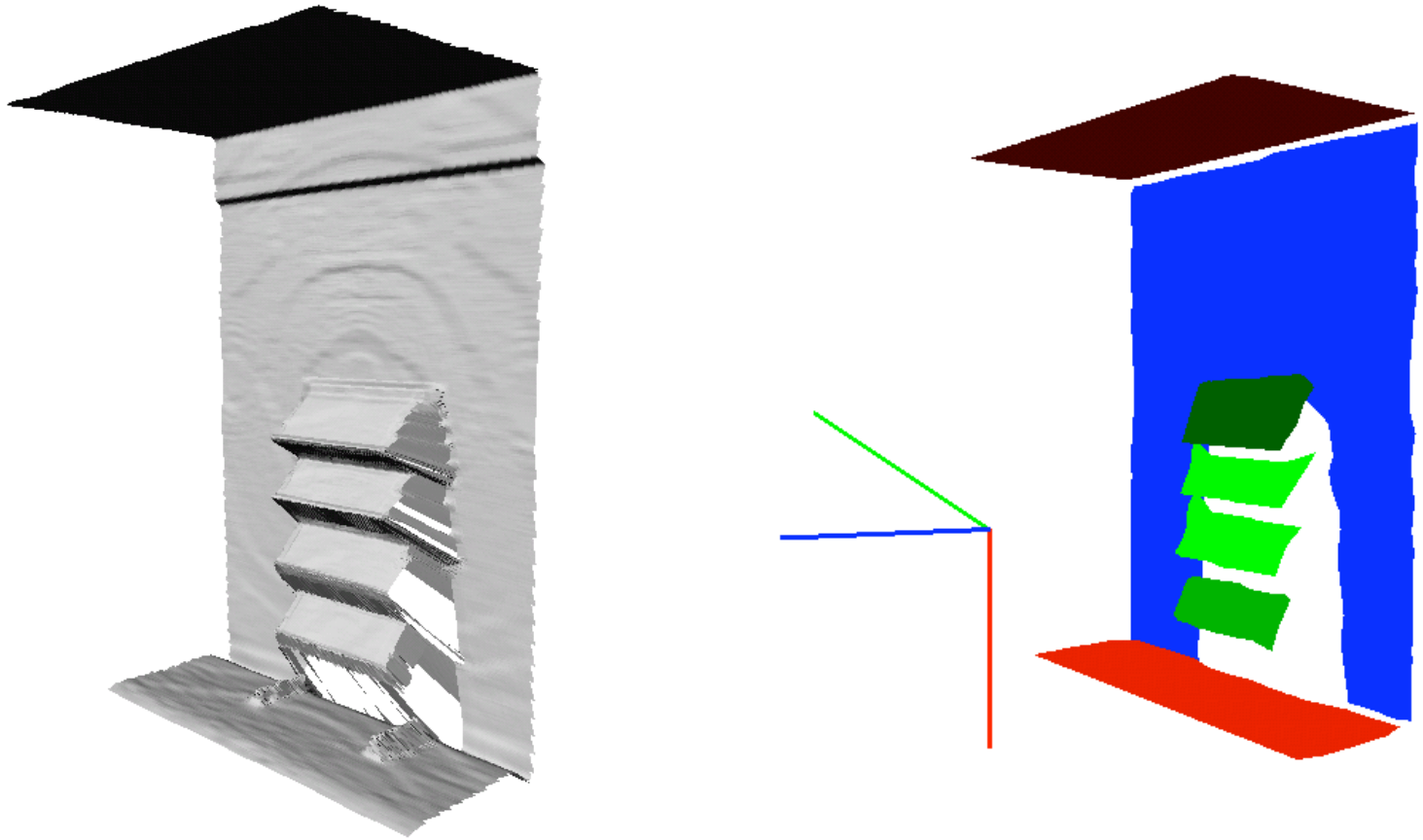
Standard EM:



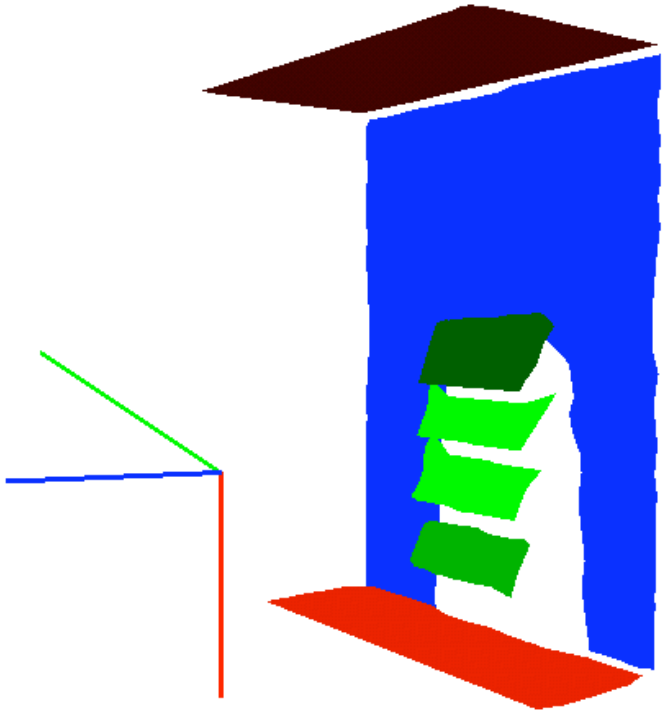
Hierarchical EM:

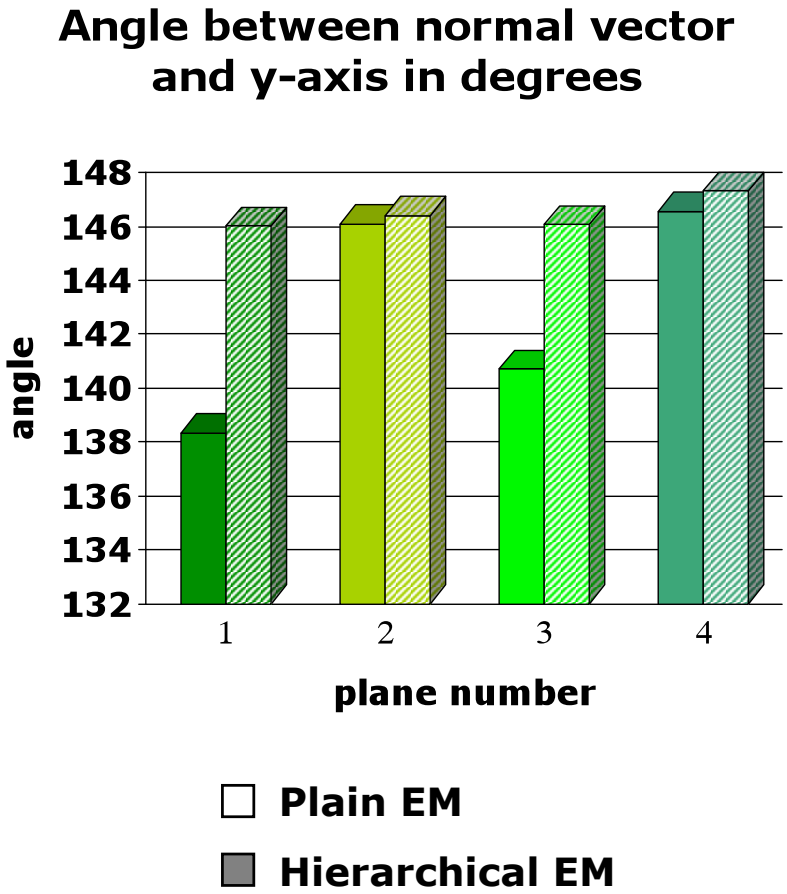


Real Data Experiment

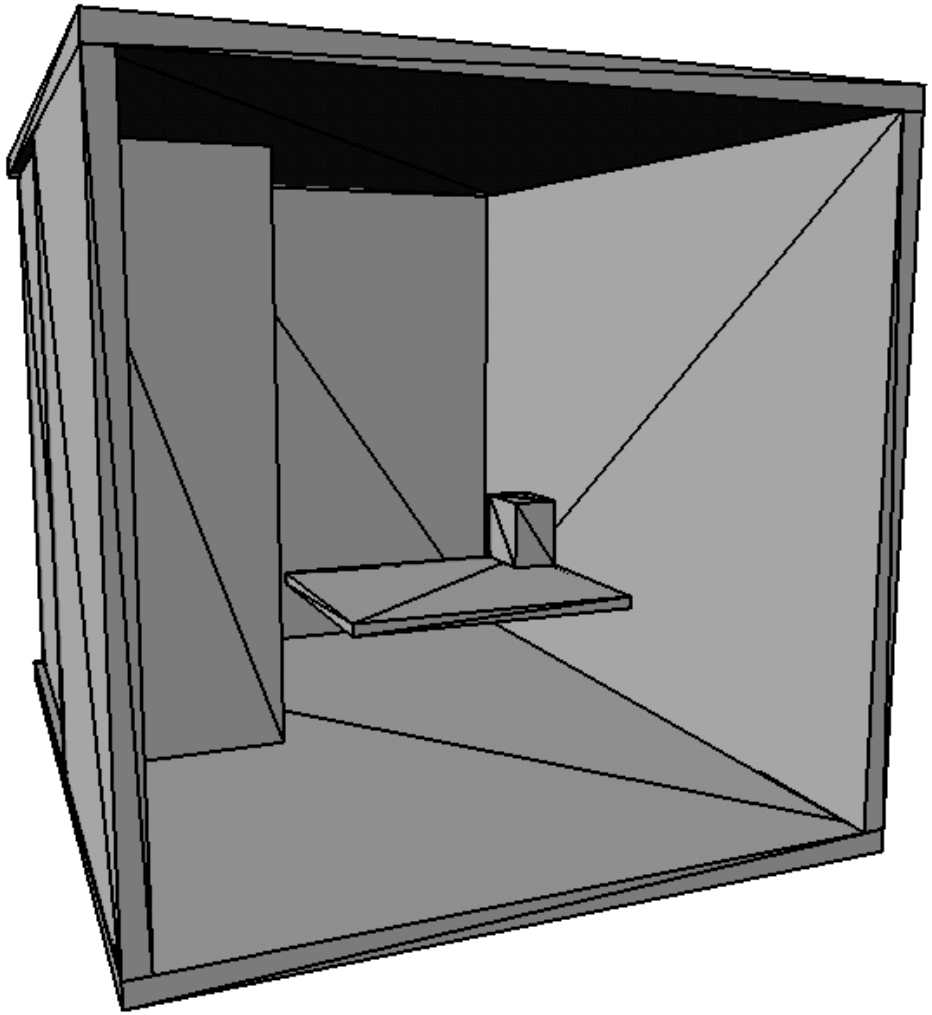


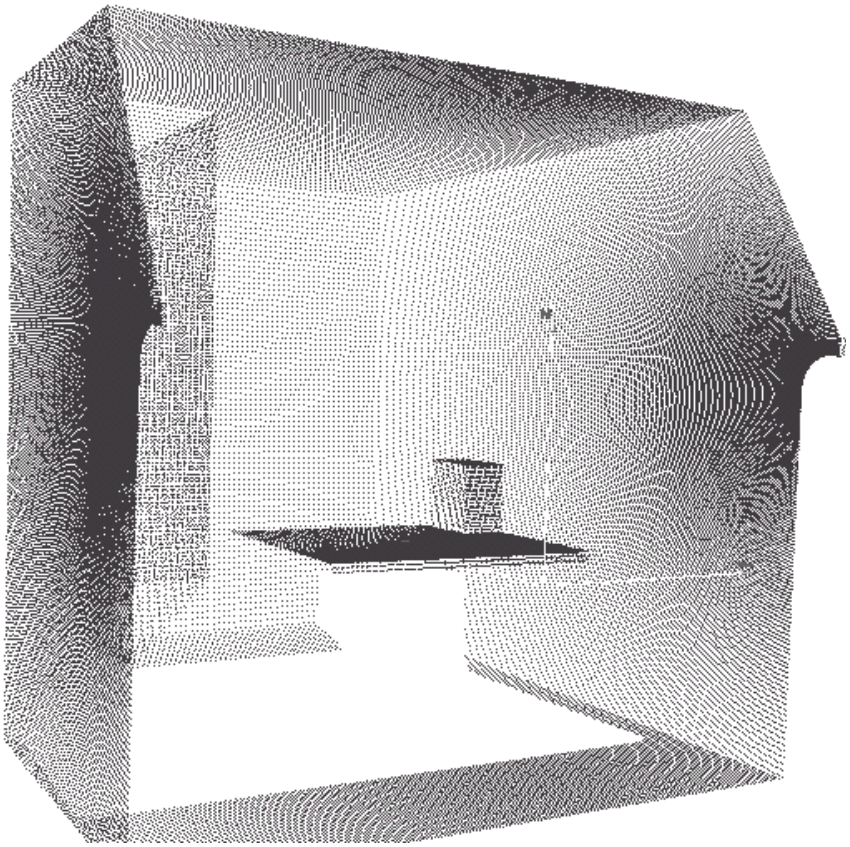
Real Data Experiment



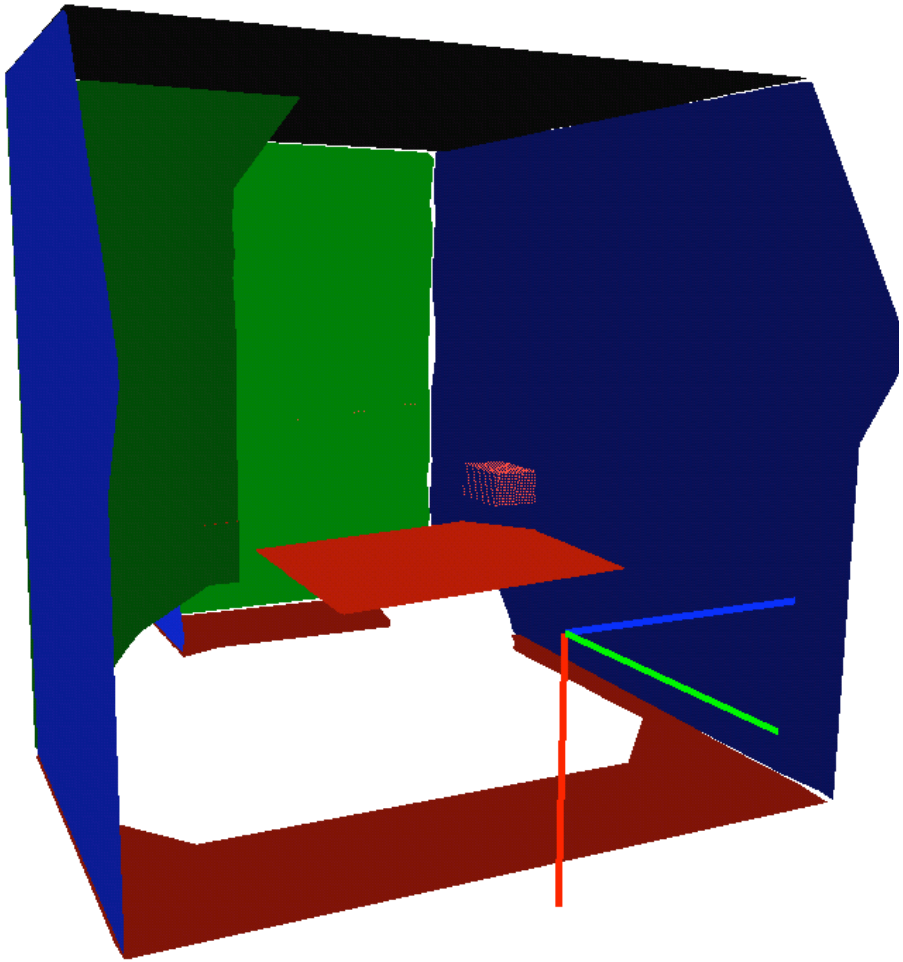


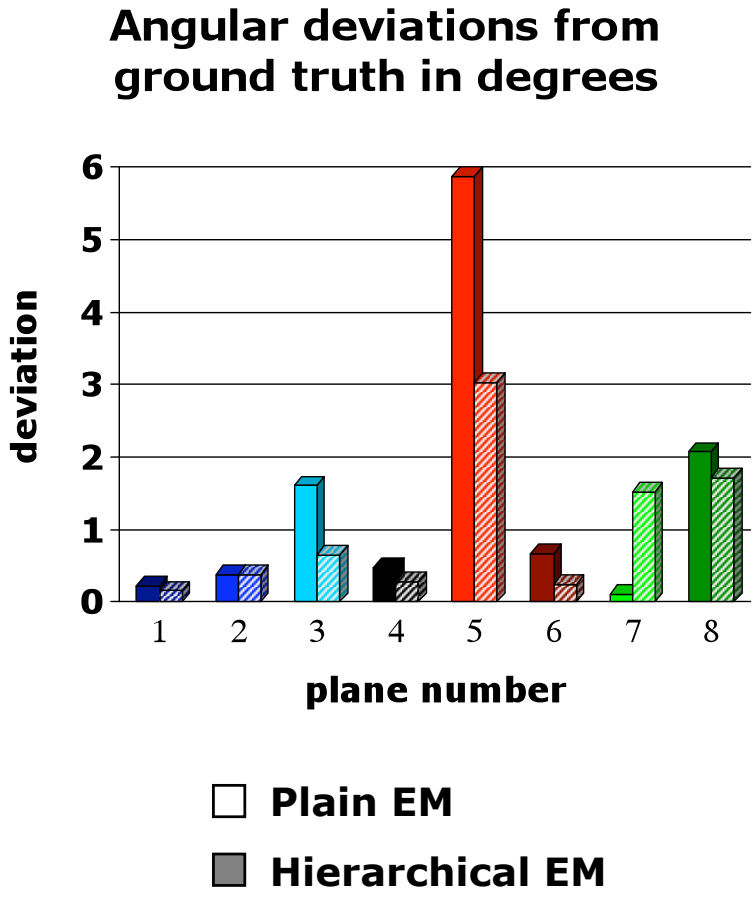
Quantitative Evaluation (1)





Quantitative Evaluation (2)

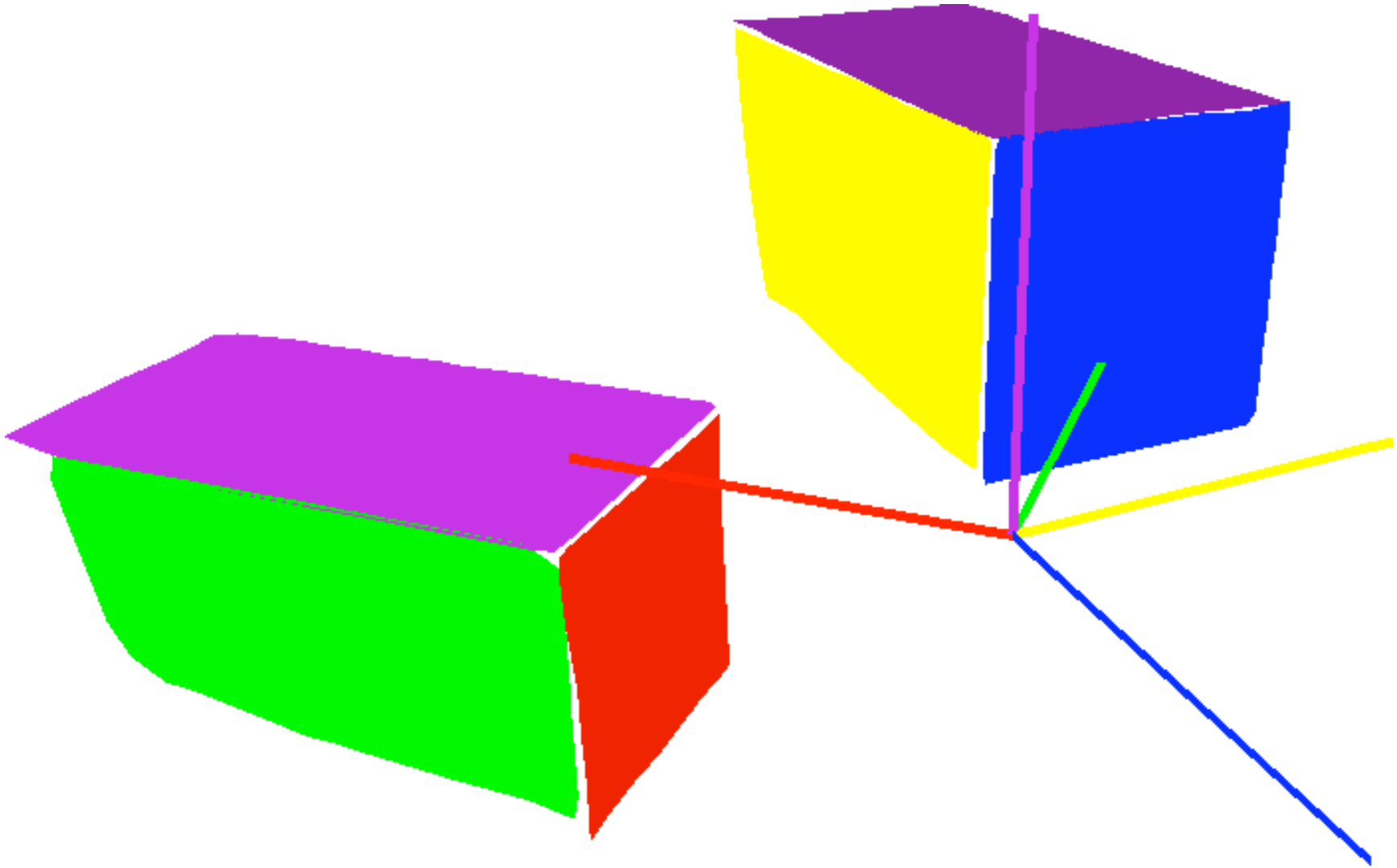




Results

- Hierarchical approach that simultaneously clusters data into planes and planes into main directions
- Expectation Maximization to find most likely model
- Automatic estimation of model complexity using BIC
- Yields more accurate models by utilizing the estimated “global constraints”

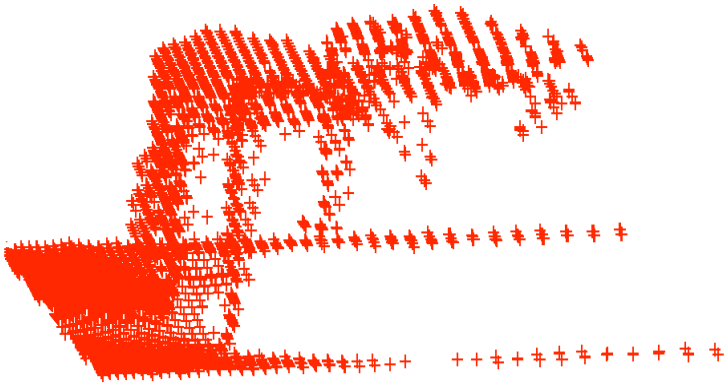
More than 3 Main Directions



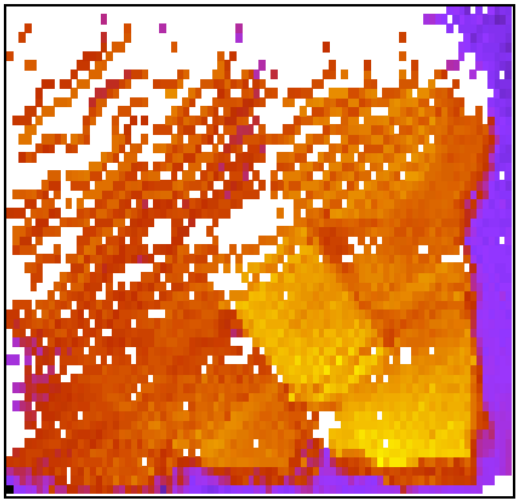
Terrain Modeling (Revisited)

- Common representations for terrain data:

Point clouds



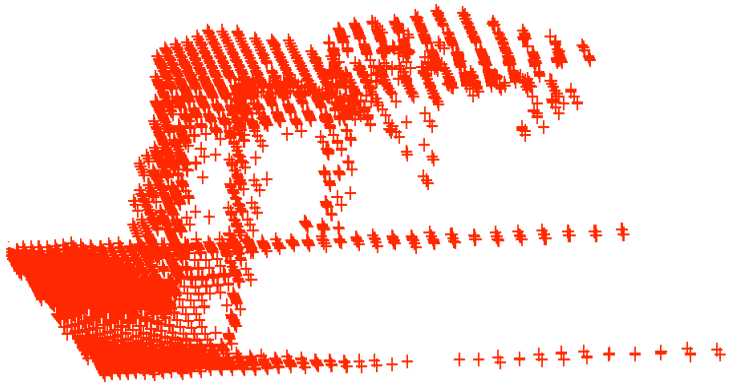
Grid maps



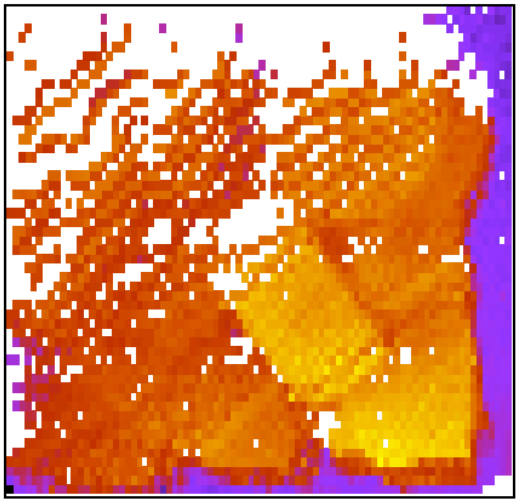
Terrain Modeling (Revisited)

- Common representations for terrain data:

Point clouds



Grid maps

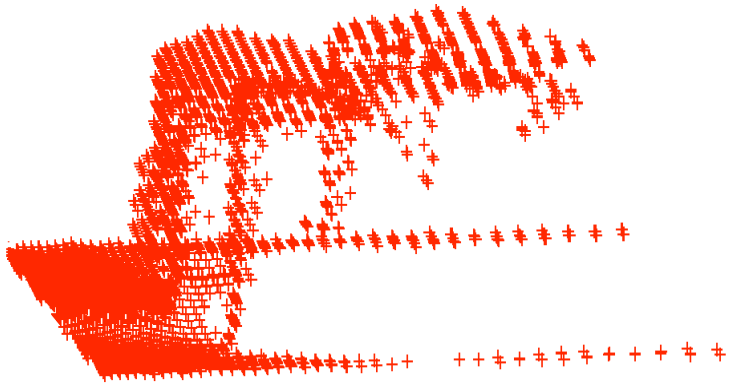


+ in continuous 3D
+ max. resolution
- sparse
- noisy

Terrain Modeling (Revisited)

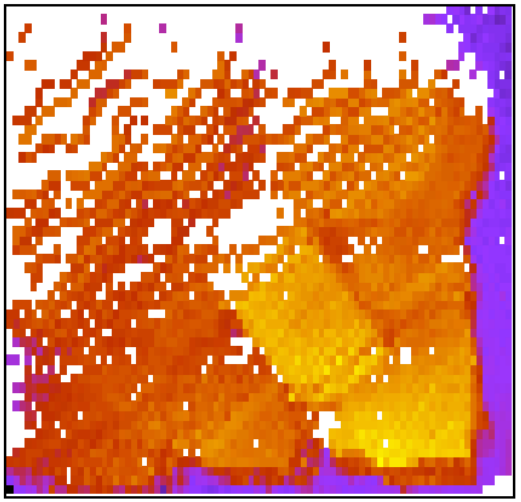
- Common representations for terrain data:

Point clouds



+ in continuous 3D
+ max. resolution
- sparse
- noisy

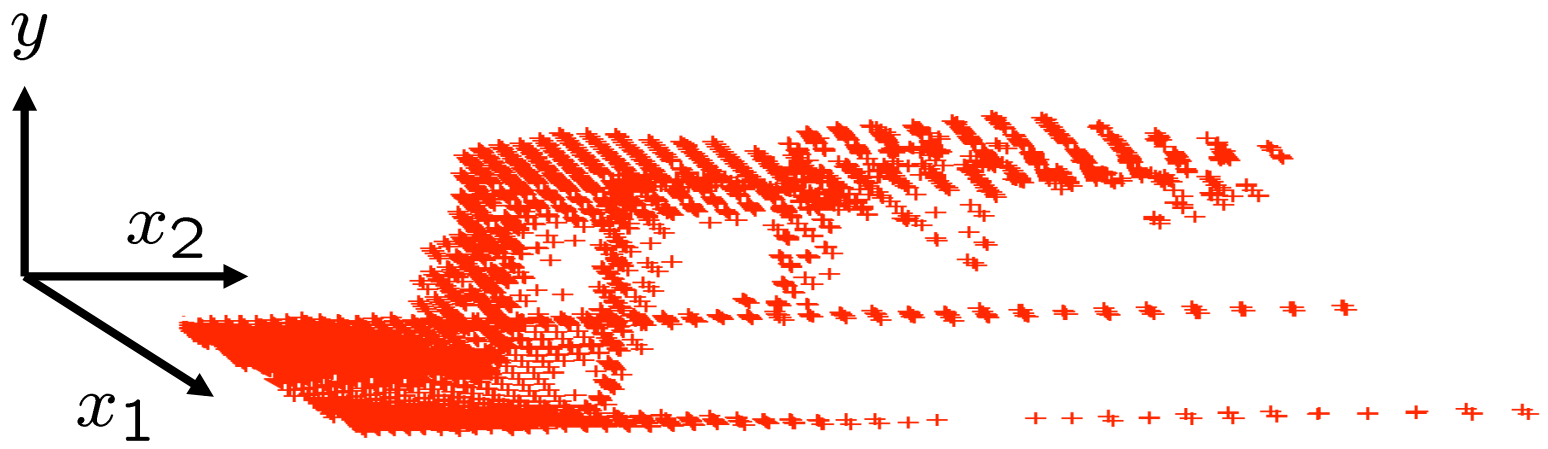
Grid maps



+ compact
+ reduced noise
- sparse
- fixed resolution

Modeling Terrain Elevation

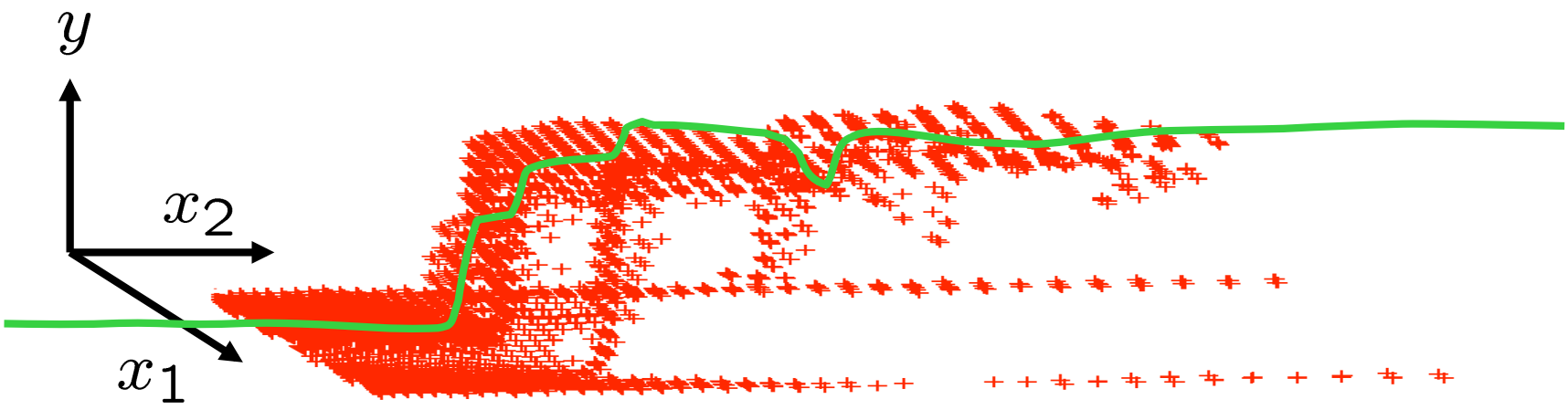
- Terrain mapping seen as a regression problem:



$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Modeling Terrain Elevation

- Terrain mapping seen as a regression problem:

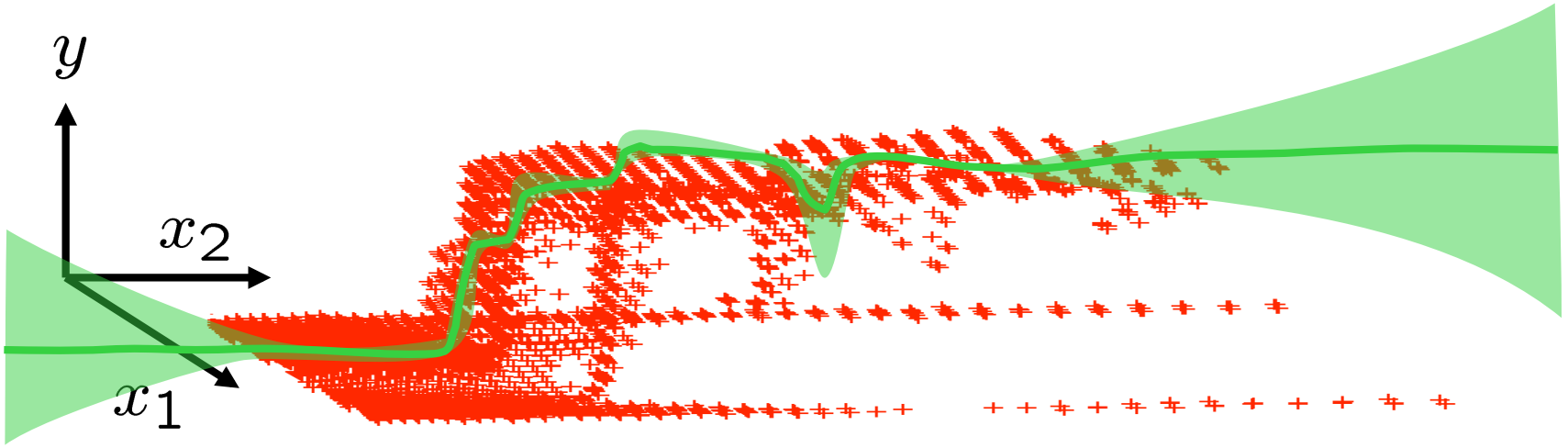


$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

$$y = f(\mathbf{x}) + \epsilon$$

Modeling Terrain Elevation

- Terrain mapping seen as a regression problem:



$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

$$\downarrow$$

$$y = f(\mathbf{x}) + \epsilon$$

$$\downarrow$$

$$p(y_* \mid \mathbf{x}_*, \mathcal{D})$$

LA-GP: Locally Adaptive GPs

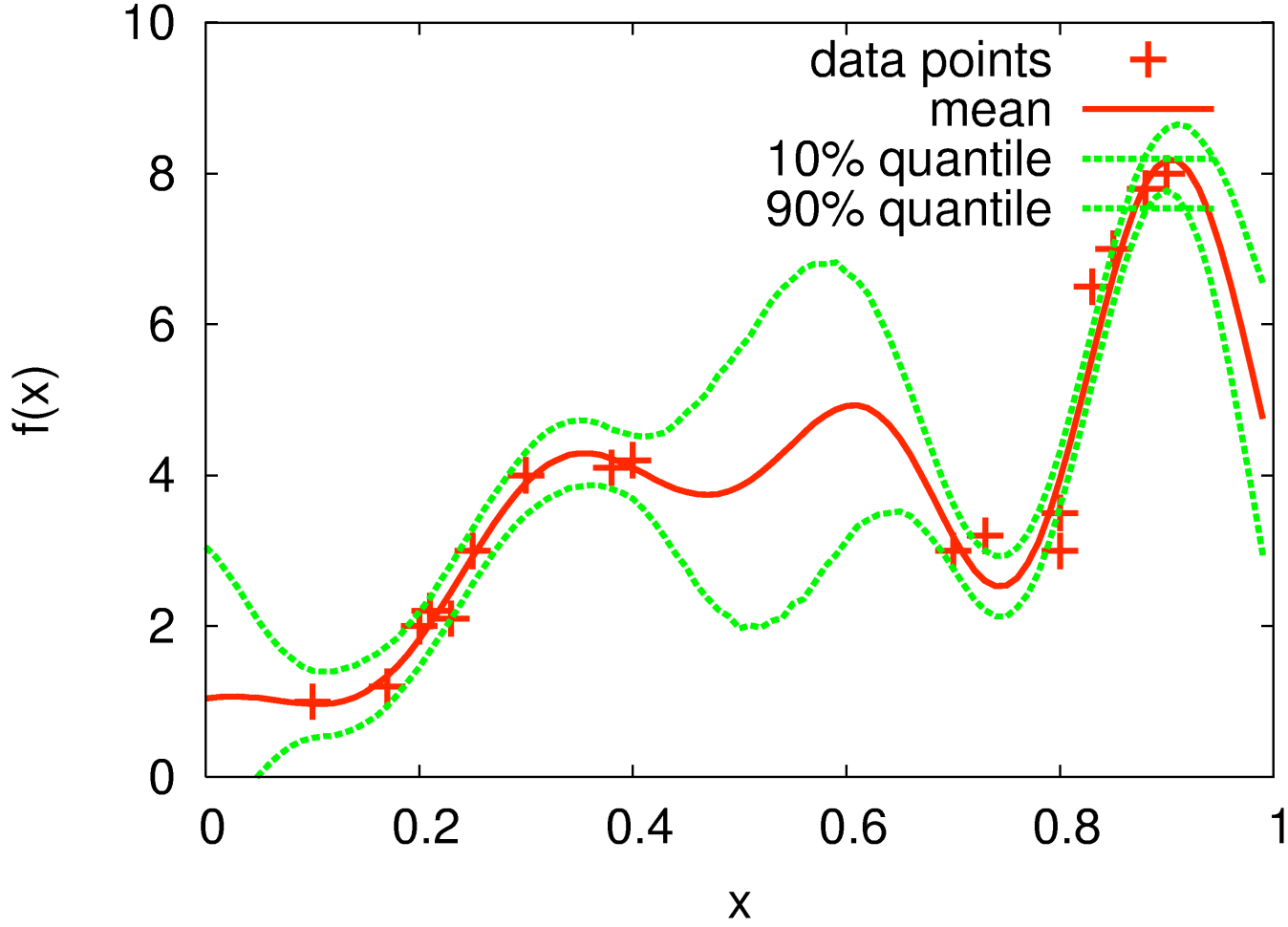
- Approach: Gaussian process regression
- Two extensions:

1. Nonstationary covariances:
   Adapting the smoothness locally

2. Model tiling:
   Sparse approximations for large data sets

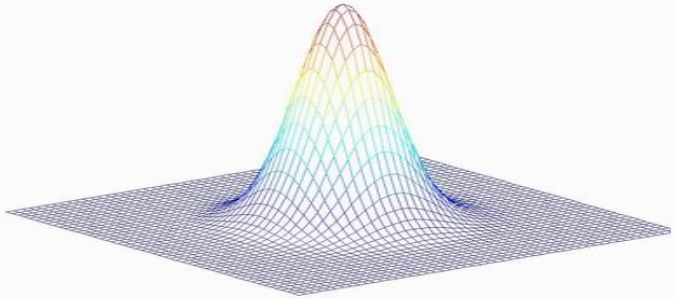
Gaussian Process Regression

- Prediction of a new value y_* at location $\mathbf{x}_*$?

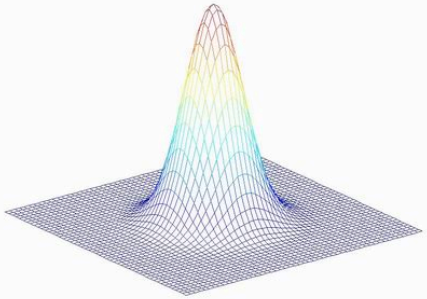


Gaussian Process Regression

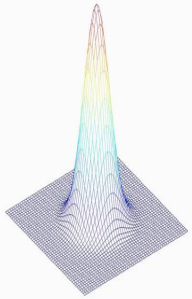
- Stationary covariance functions model *global* function smoothness



broad



medium



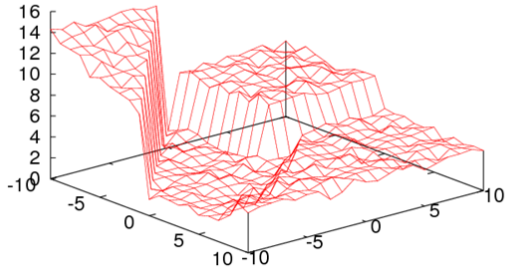
narrow

smooth

medium

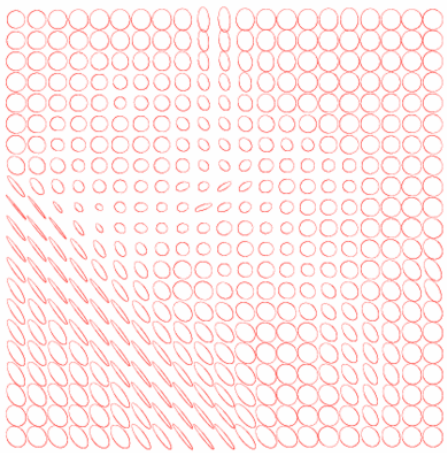
wiggly

Iterative Adaptation

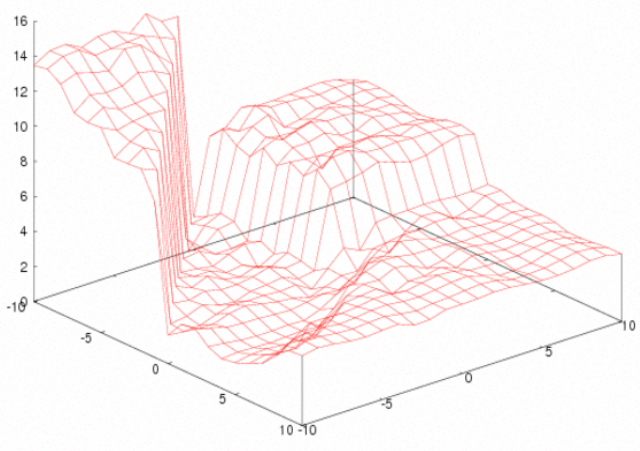


Observation (with white noise σ=0.3)

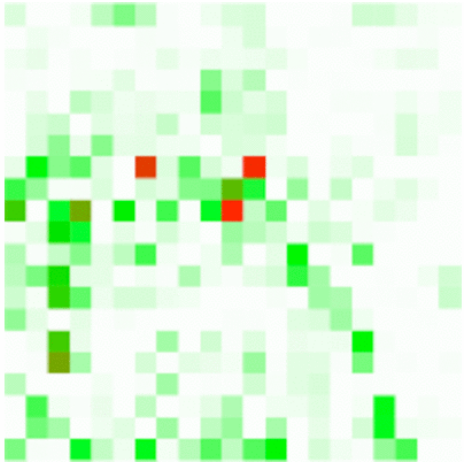
Kernels



Predicted Map

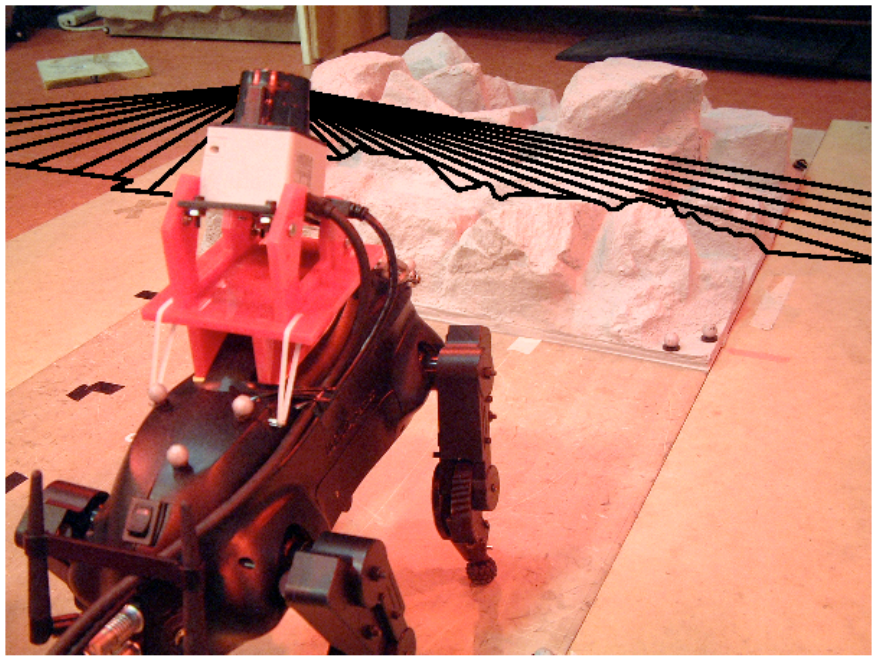


Local errors



Experiments

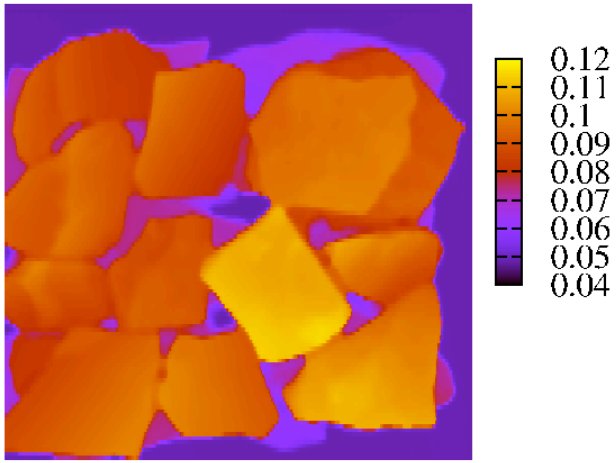
1. We scanned a terrain board, learned an LA-GP model and compared it to the known ground truth.



2. We evaluated the usefulness for motion planning.
3. We used the learned model to traverse a terrain.

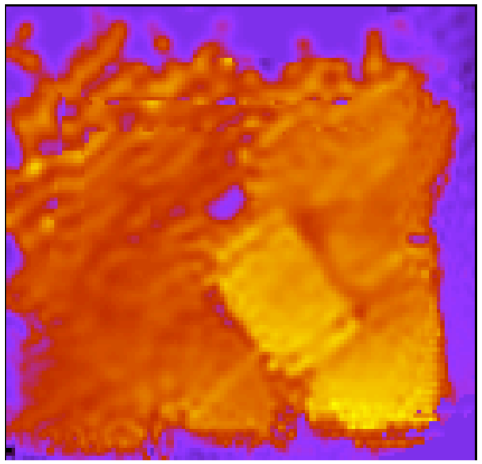
Experiments

True Terrain

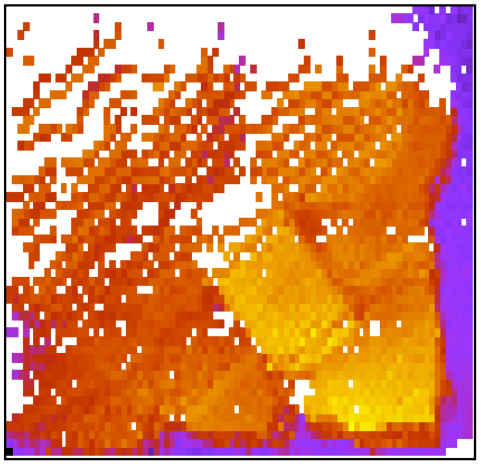


Laser End Points

Adapted GP Model



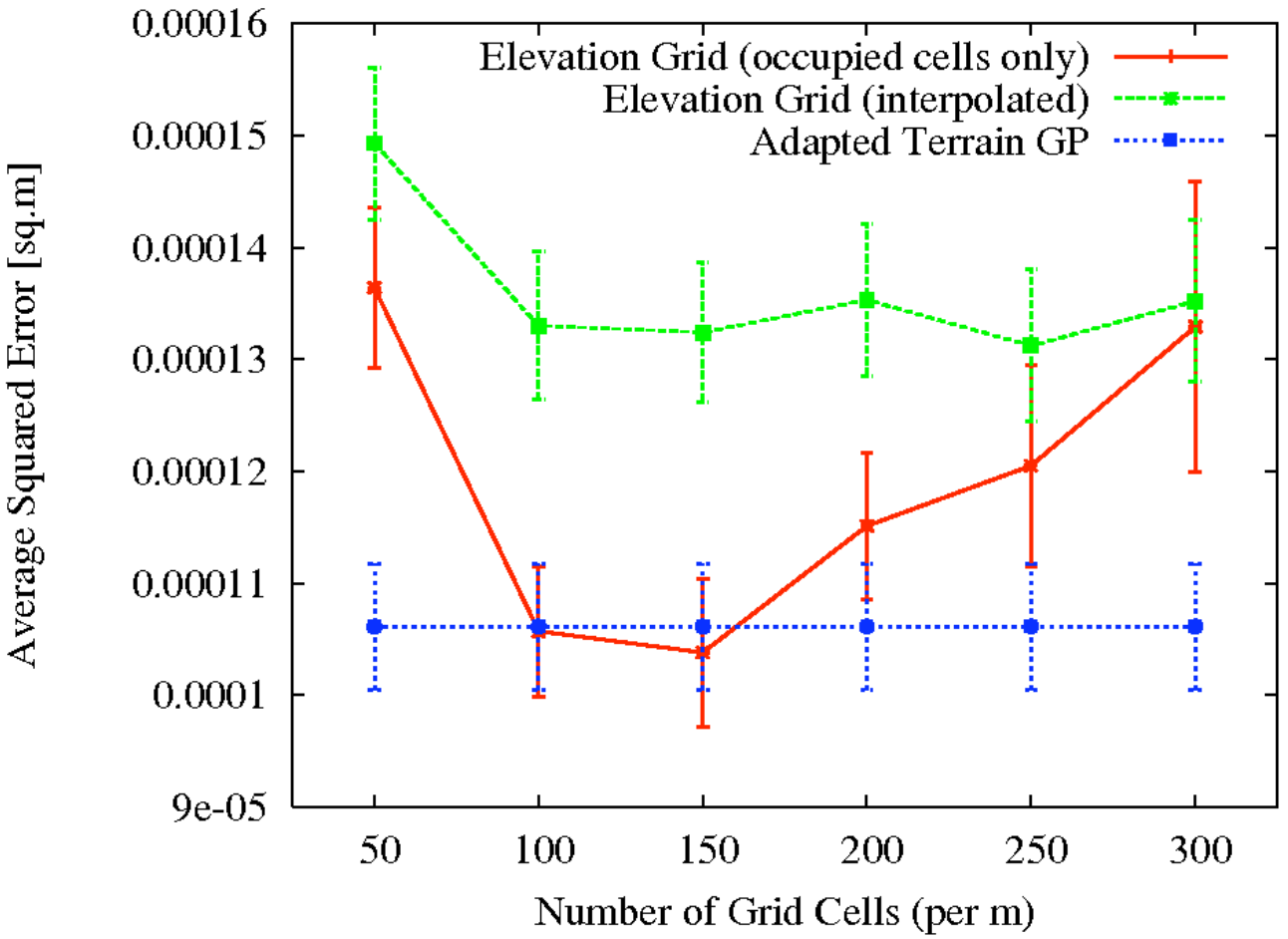
Elevation Grid Map



Experiments

True Terrain

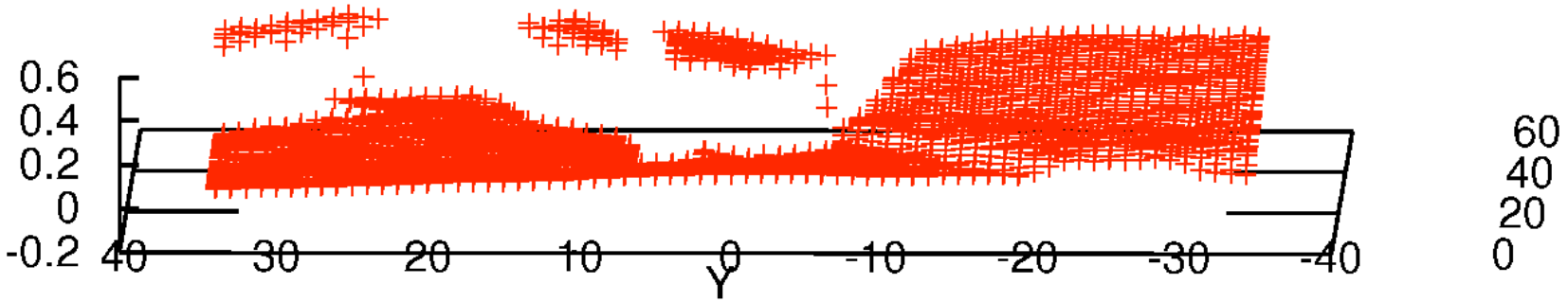
Laser End Points



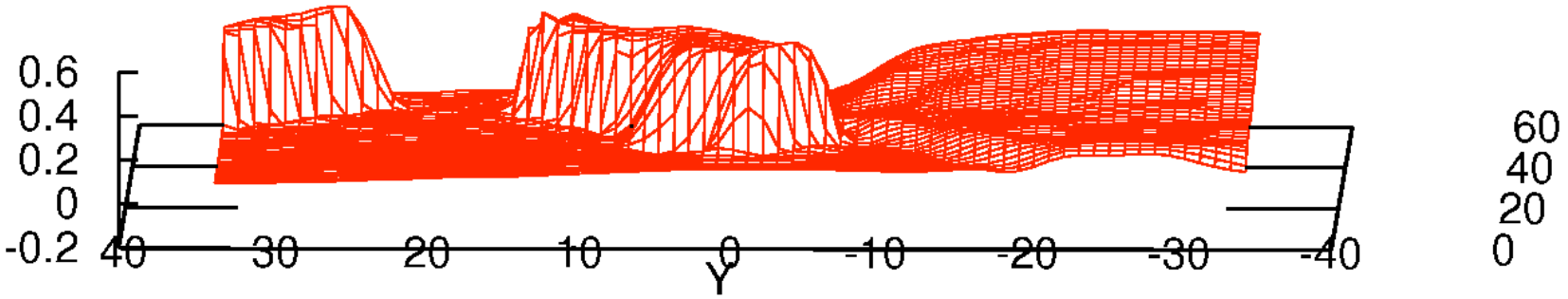
Experiments

Experiments

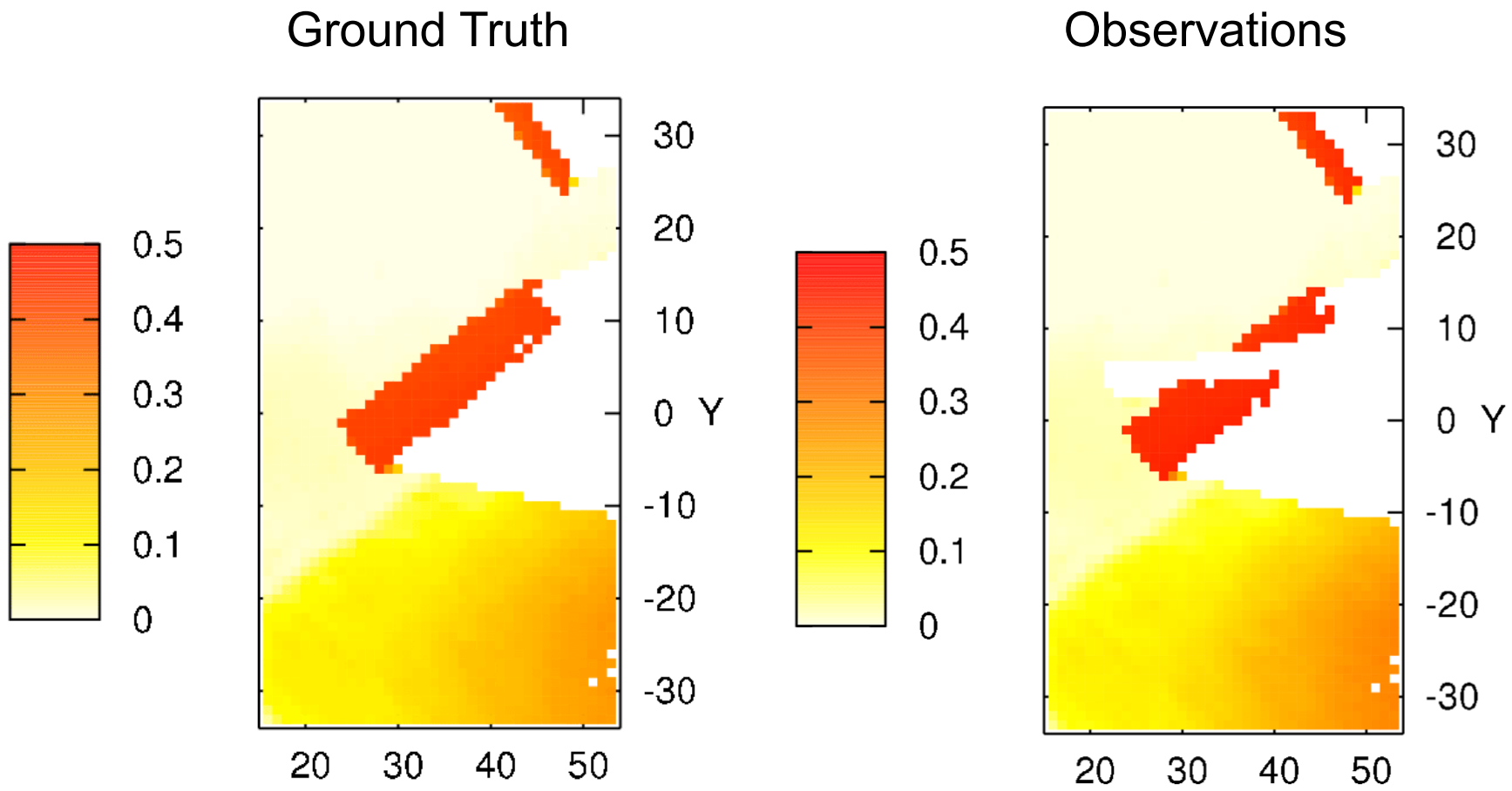
Observations:



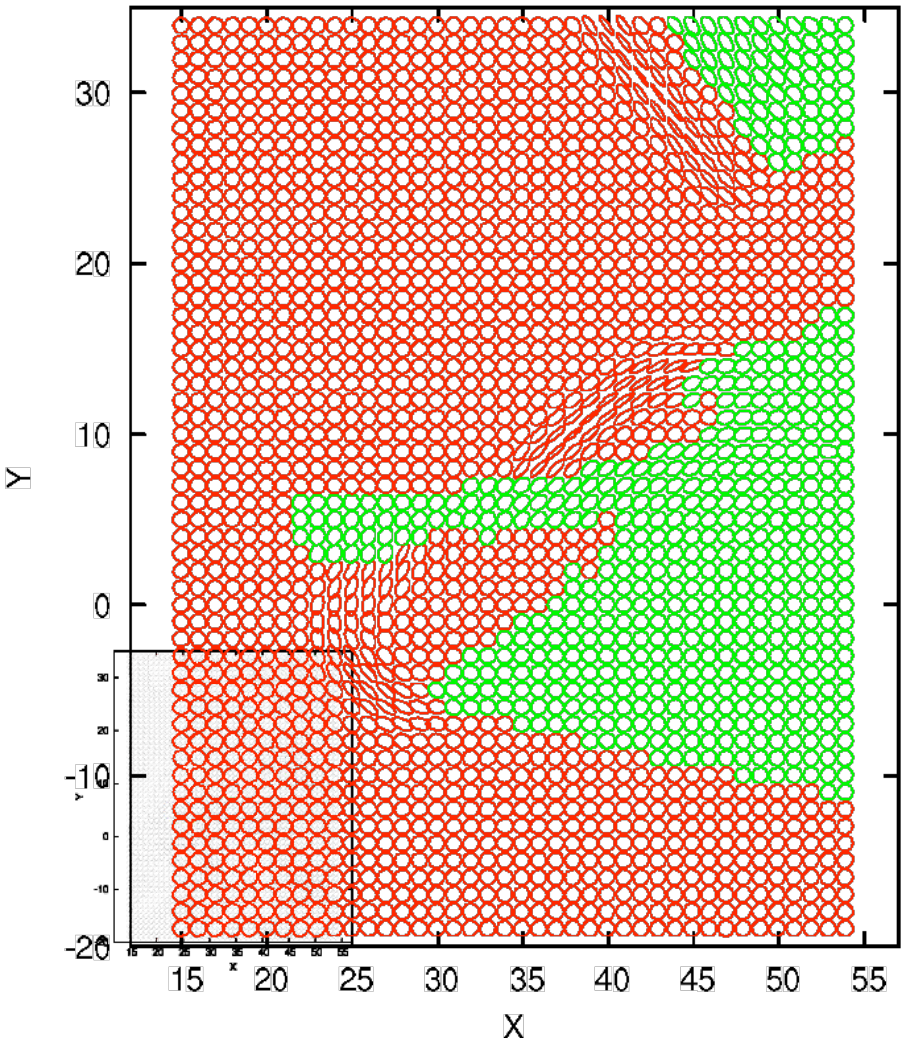
The adapted terrain model:

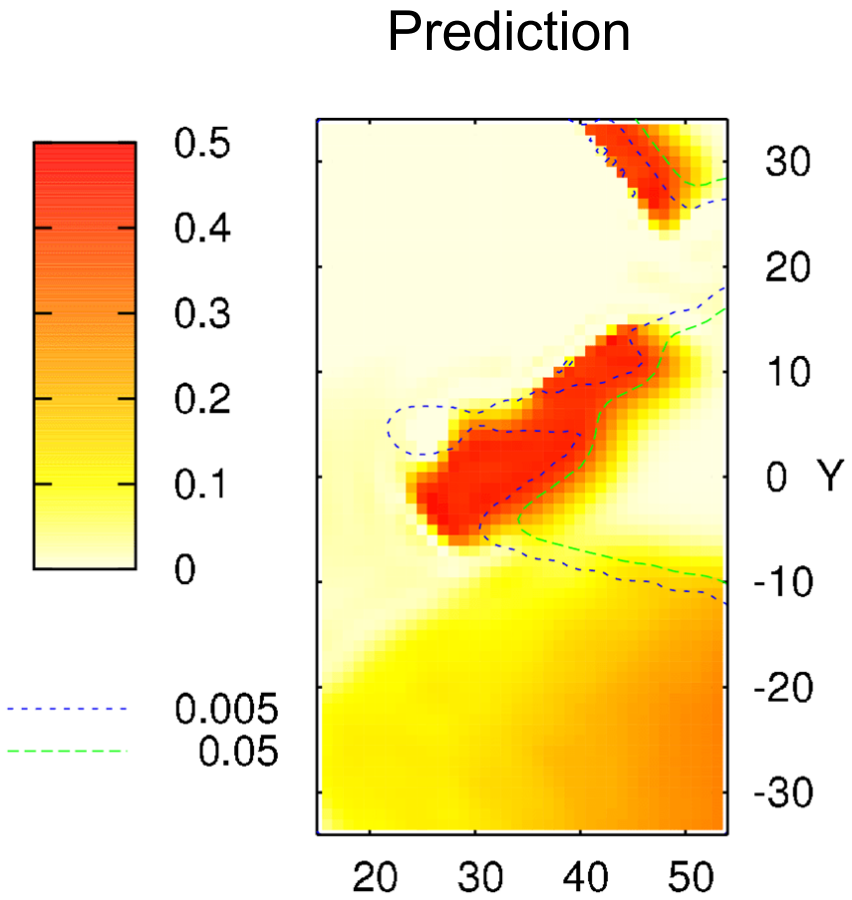


Experiments

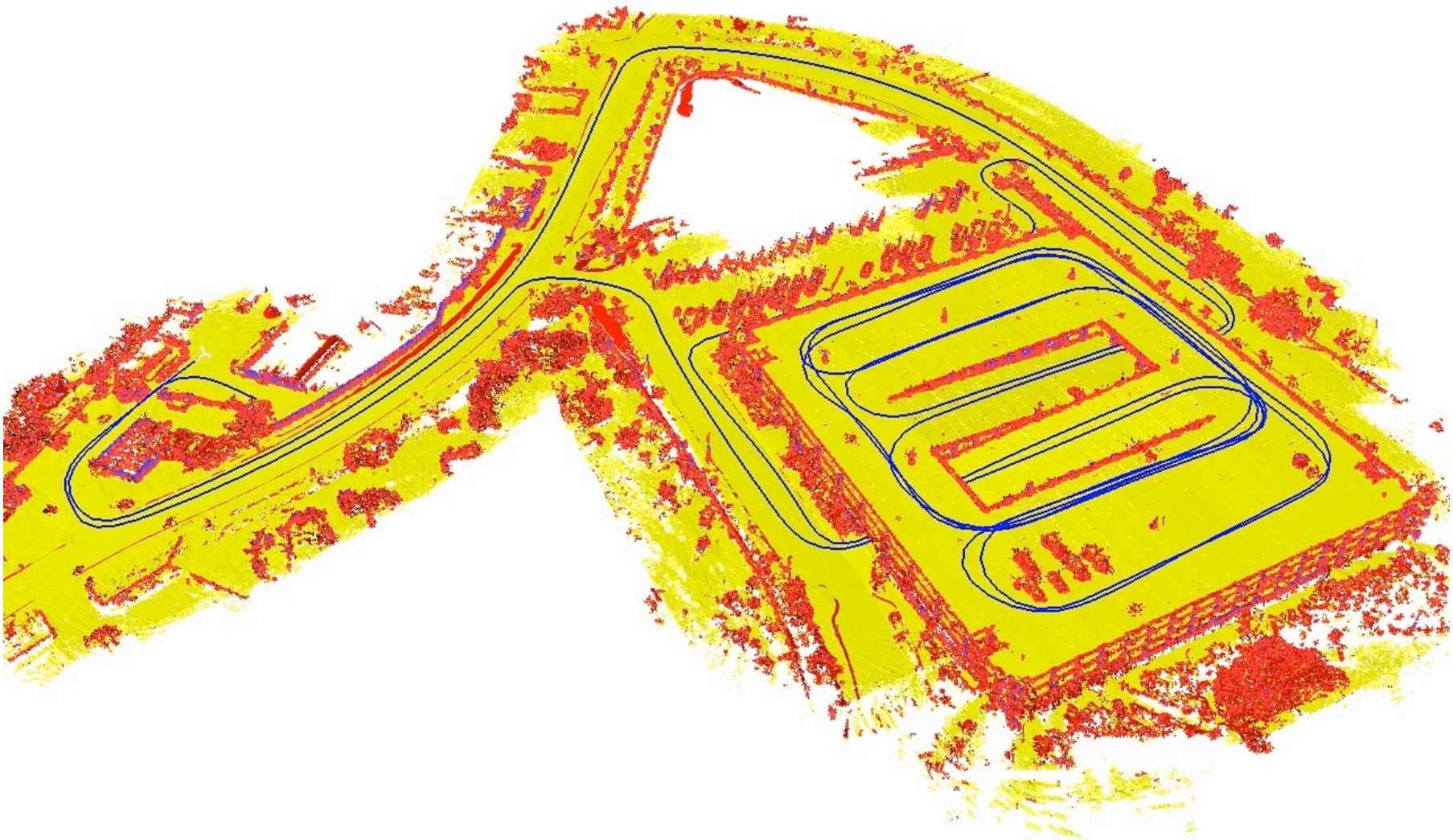


Experiments





Autonomous Parking



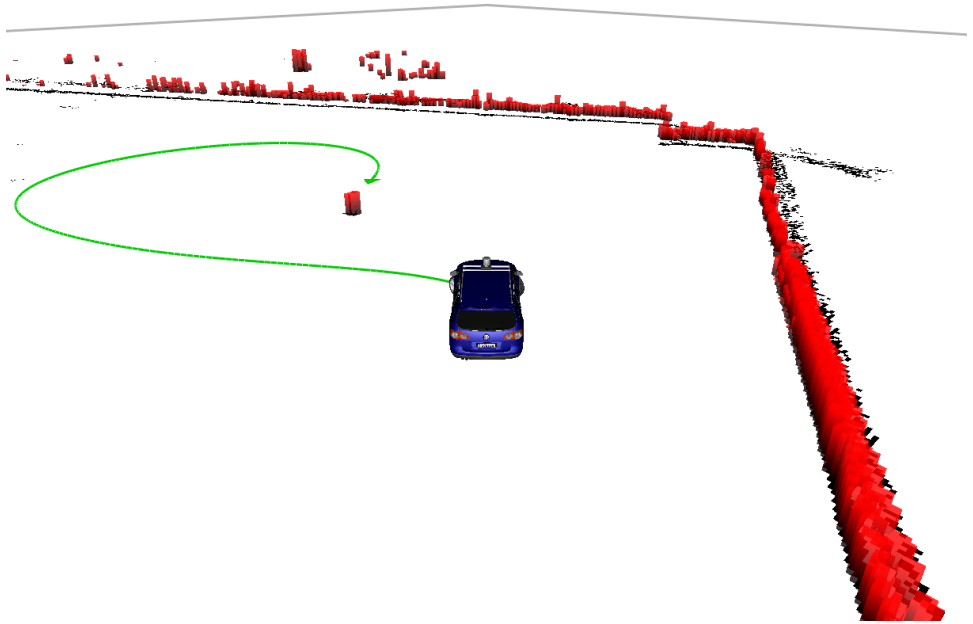
approx. 260MB

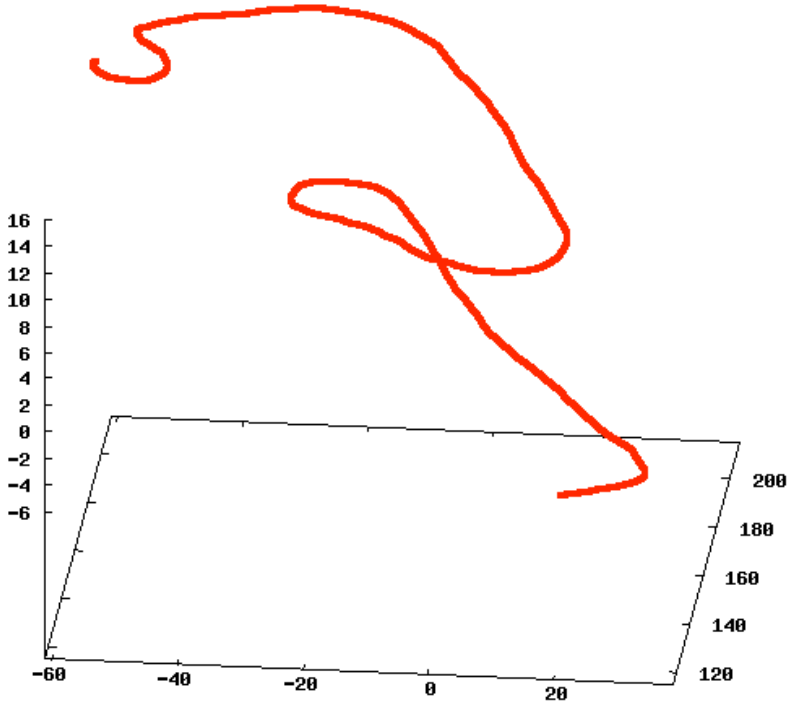
Navigation with the Autonomous Car Junior

- Task: reach a parking spot on the upper level of the garage.

Planned Trajectories

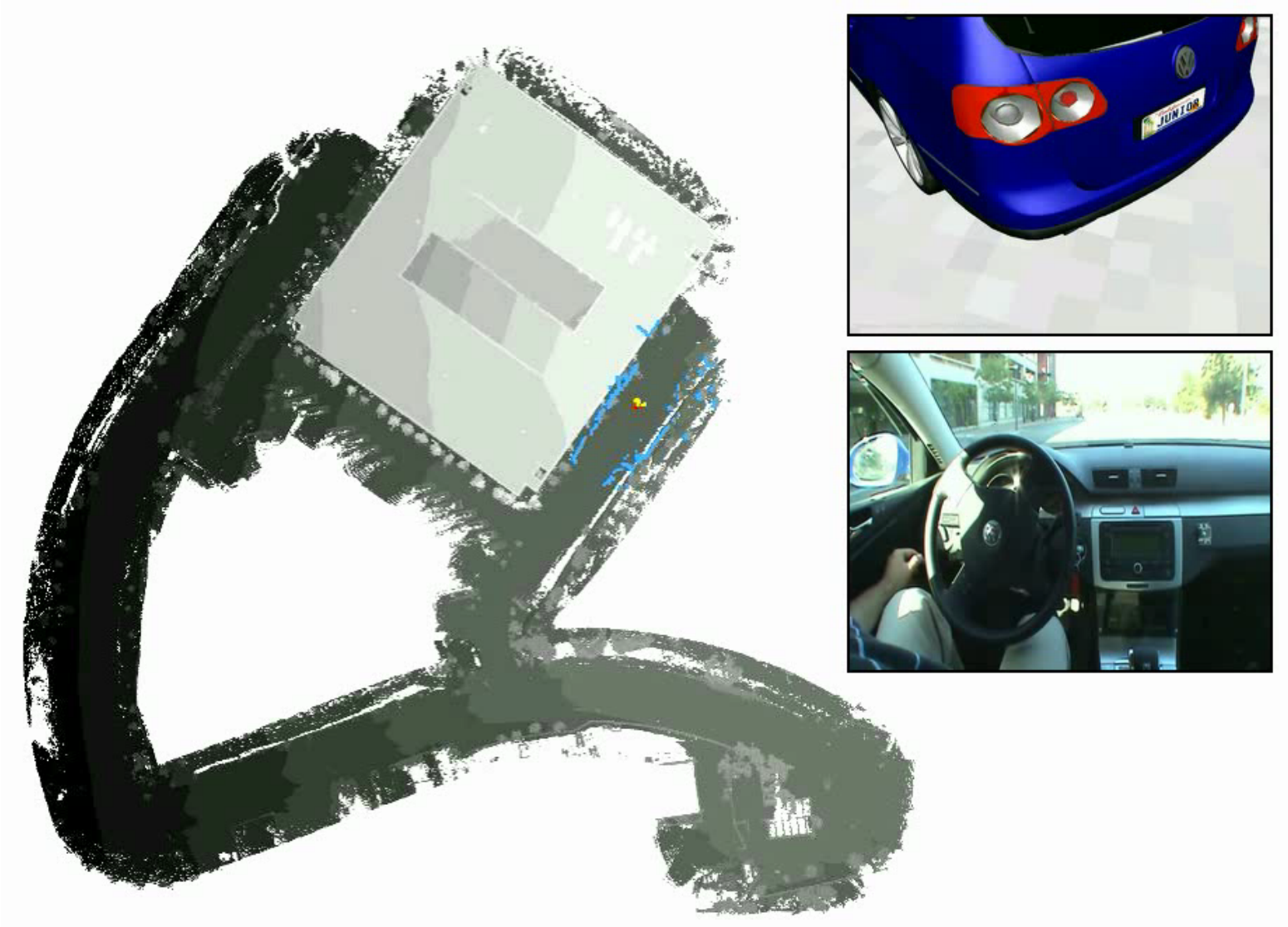
Local plan
towards the goal





Global trajectory

Experiment with Junior



Mapping in Dynamic Environments

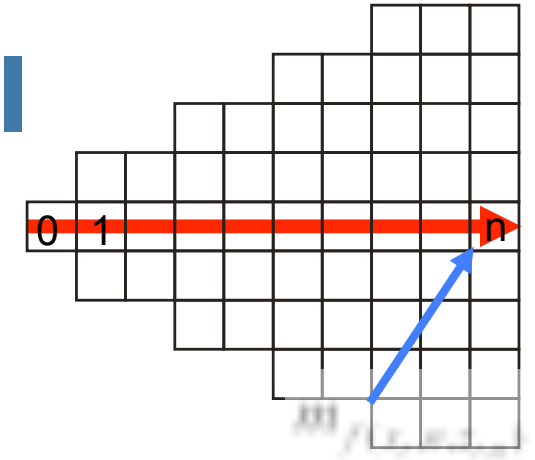
Problem:

- Often models of non-stationary objects are not available.
- Often we cannot assume that there is a separation between non-stationary and static objects.

Solution:

- Using EM to learn beams reflected by dynamic objects.

The Measurement Model



1. pose at time *t*: x_t
2. beam *n* of scan *t*: $z_{t,n}$
3. maximum range reading: $\varsigma_{t,n} = 1$
4. beam reflected by dynamic object: $\varsigma_{t,n} = 0 \text{ and } c_{t,n} = 0$
5. beam reflected by static object: $\varsigma_{t,n} = 0 \text{ and } c_{t,n} = 1$

$$p(z_{t,n} \mid c_{t,n}, x_t, m) = \begin{cases} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \text{if } \varsigma_{t,n} = 1 \\ \prod_{k=0}^{z_{t,n}} (1 - m_{f(x_t,n,k)}) & \text{if } \varsigma_{t,n} = 0 \text{ and } c_{t,n} = 0 \\ m_{f(x_t,n,z_{t,n})} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \text{if } \varsigma_{t,n} = 0 \text{ and } c_{t,n} = 1 \end{cases}$$

Application of EM

$$E_c[\ln p(z, c \mid x, m) \mid z, x, m] = N \cdot \sum_{t=1}^{T} E_c[\ln p(c_t)]$$

$$+ \sum_{t=1}^{T} \sum_{n=1}^{N} \Big[E_c[c_{t,n} \mid z_{t,n}, x_t, m] \cdot (1 - \zeta_{t,n}) \cdot \ln m_{f(x_t,n,z_{t,n})}$$

$$+ (1 - E_c[c_{t,n} \mid z_{t,n}, x_t, m]) \cdot (1 - \zeta_{t,n}) \cdot \ln(1 - m_{f(x_t,n,z_{t,n})}) + \sum_{k=0}^{z_{t,n}-1} \ln(1 - m_{f(x,n,k)}) \Big]$$

E-Step:

$$e_{t,n} = \begin{cases} \dfrac{p(c_{t,n}=1 \mid \zeta_{t,n}=1, x_t, m)}{p(c_{t,n}=0 \mid \zeta_{t,n}=1, x_t, m) + p(c_{t,n}=1 \mid \zeta_{t,n}=1, x_t, m)} & \varsigma_{t,n} = 1 \\ p(c_{t,n}=1) \cdot \dfrac{m_{f(x_t,n,z_{t,n})}}{m_{f(x_t,n,z_{t,n})} + \left(\frac{1}{p(c_{t,n}=1)} - 1\right) \cdot (1 - m_{f(x_t,n,z_{t,n})})} & \text{else} \end{cases}$$

M-Step:

Compute most likely map by considering the expectations $e_{t,n}$ during SLAM.

Computing the Most Likely Map

$$\hat{m}^{[t]} = \arg\max_{m} \left[\sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{n=1}^{N} \Big(I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n}) \cdot (e_{t,n} \ln m_j + (1 - e_{t,n}) \ln(1 - m_j)) + \sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j) \cdot \ln(1 - m_j) \Big) \right]$$

Suppose

$$\alpha_j = \sum_{t=1}^{T} \sum_{n=1}^{N} I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n}) \cdot e_{t,n}$$

$$\beta_j = \sum_{t=1}^{T} \sum_{n=1}^{N} \left[I(f(x_t, n, z_{t,n}) = j) \cdot (1 - \varsigma_{t,n}) \cdot (1 - e_{t,n}) + \sum_{k=0}^{z_{t,n}-1} I(f(x_t, n, k) = j) \right]$$

Computing the Most Likely Map

We assume that all cells m_j are independent:

$$\hat{m}^{[t]} = \arg\max_{m} \left(\sum_{j=1}^{J} \alpha_j \ln m_j + \beta_j \ln(1 - m_j) \right)$$

If we set

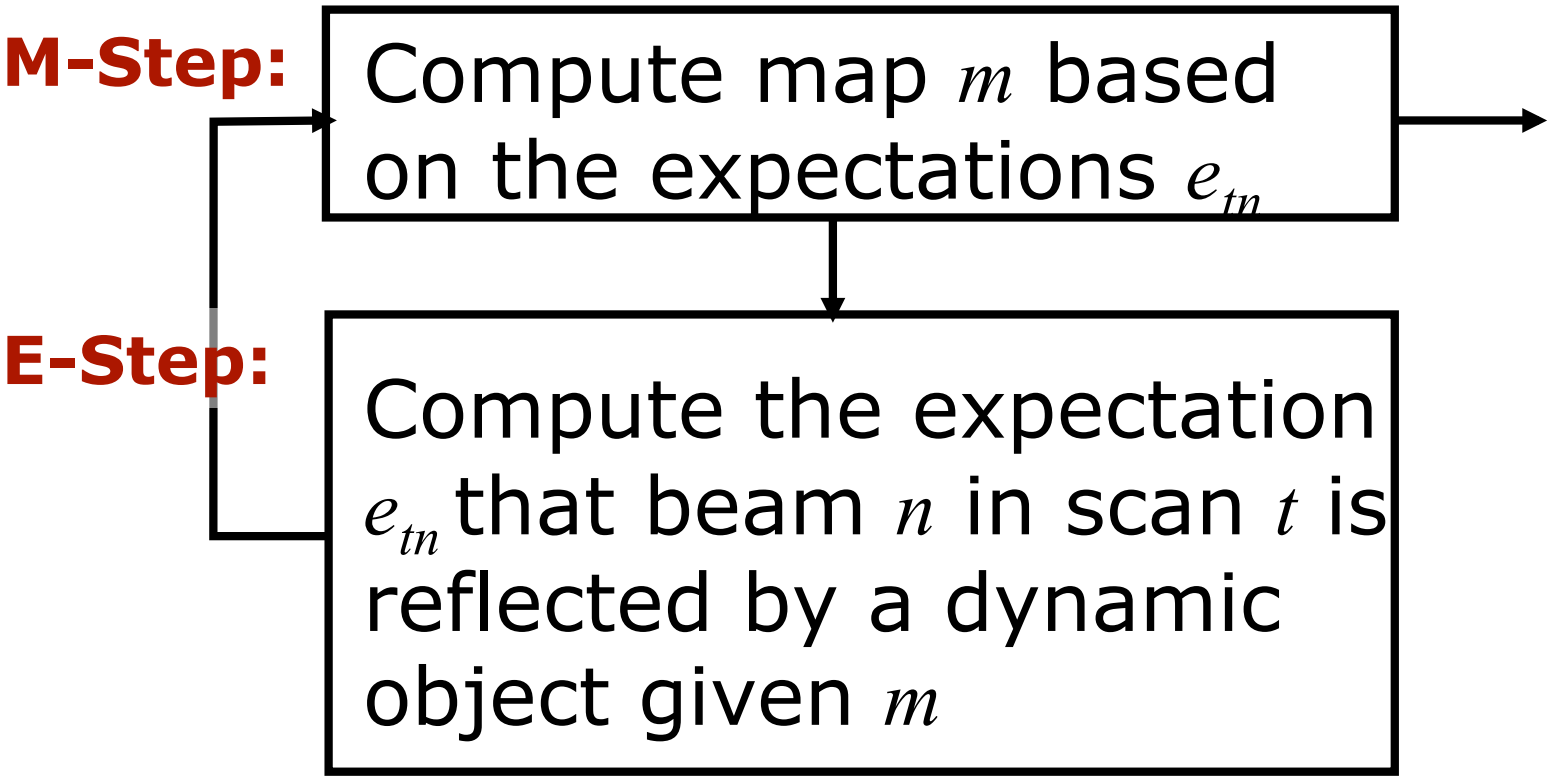
$$\frac{\partial m}{\partial m_j} = \frac{\alpha_j}{m_j} - \frac{\beta_j}{1 - m_j} = 0$$

we obtain

$$m_j = \frac{\alpha_j}{\alpha_j + \beta_j}$$

Computing the most likely map amounts to counting how often a cell has reflected a measurement.

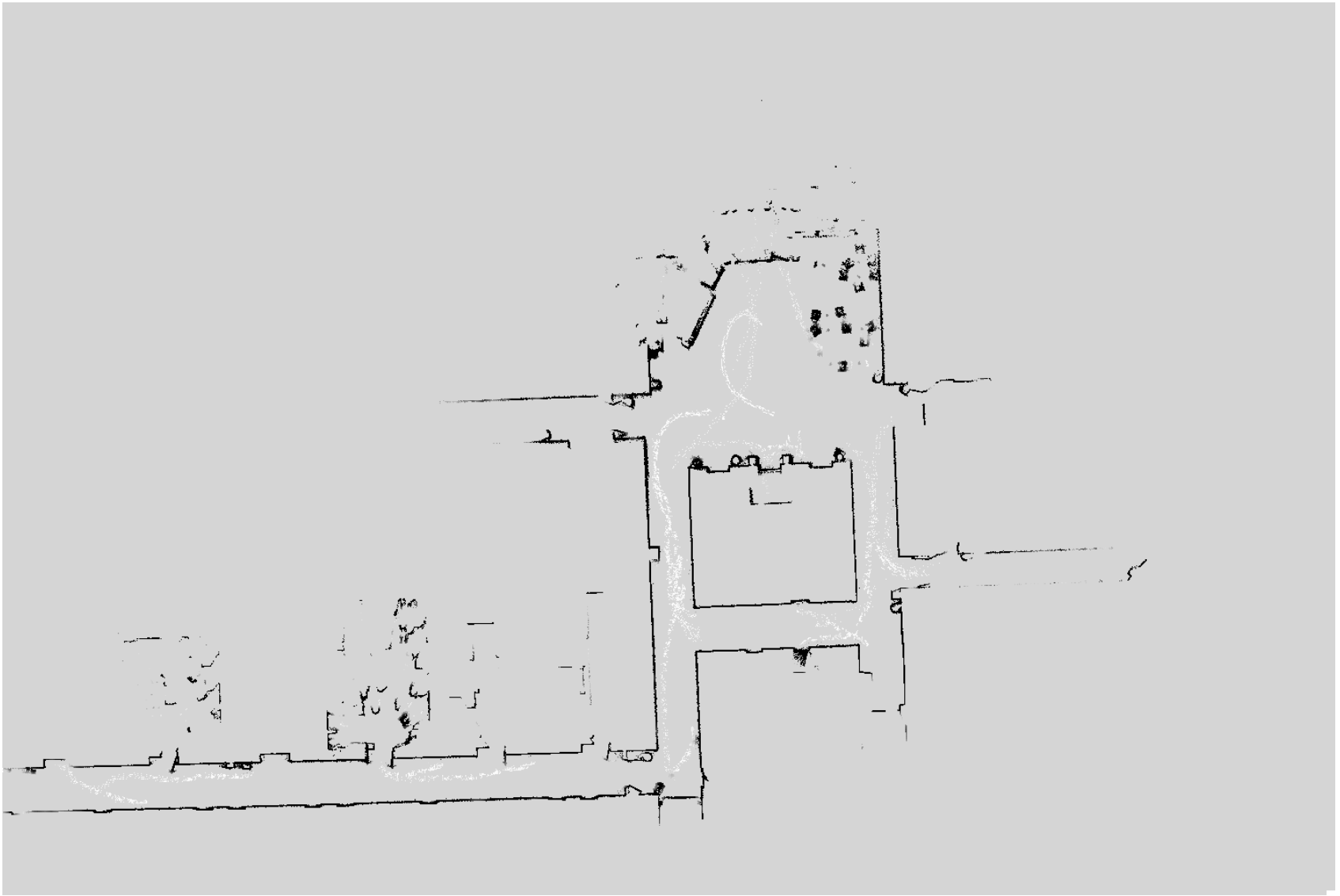
EM-based Estimation of the Most Likely Map



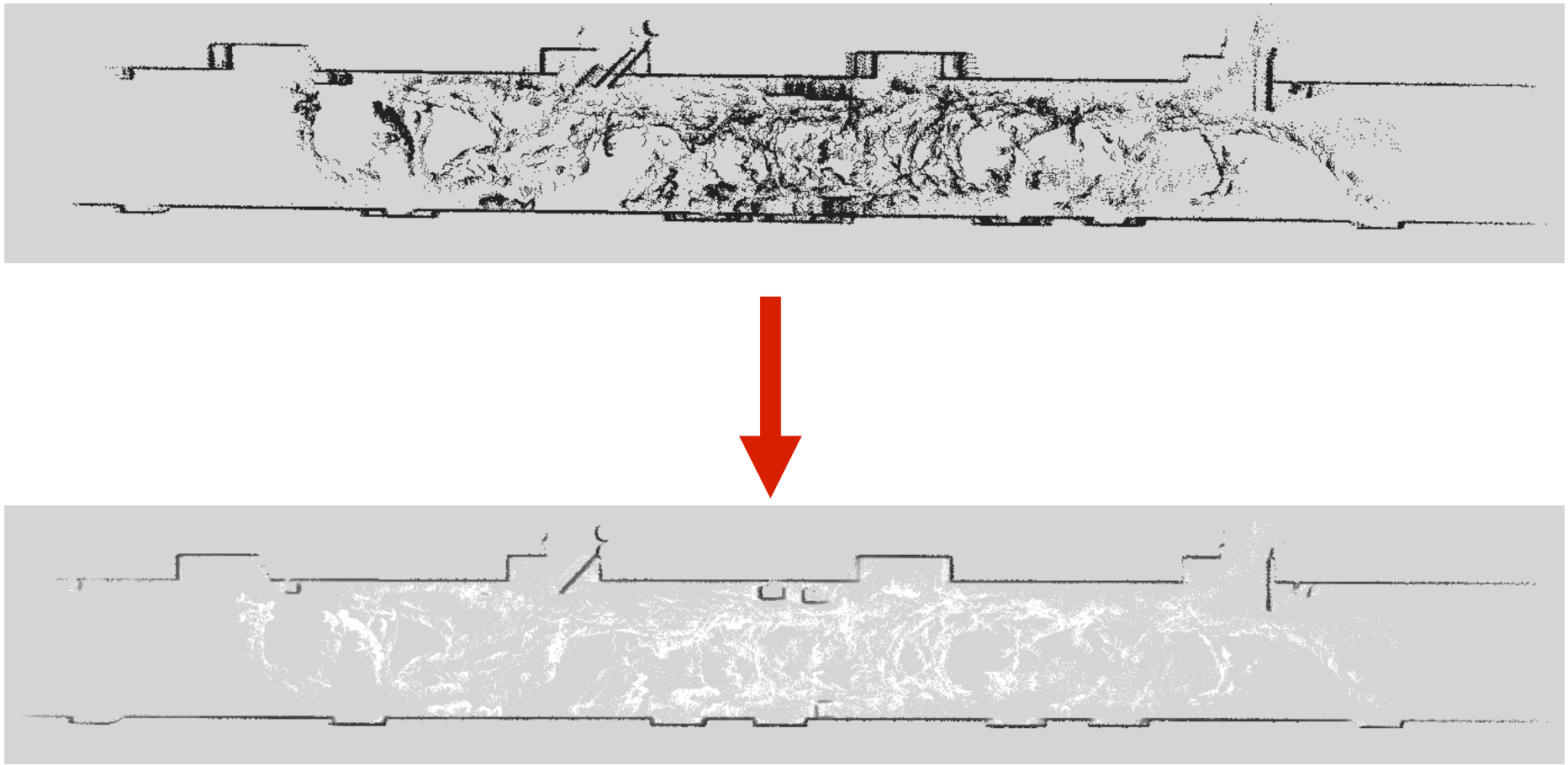
e_{tn}: Expectation that z_{tn} is reflected by a dynamic object

Byzantine Museum, Athens

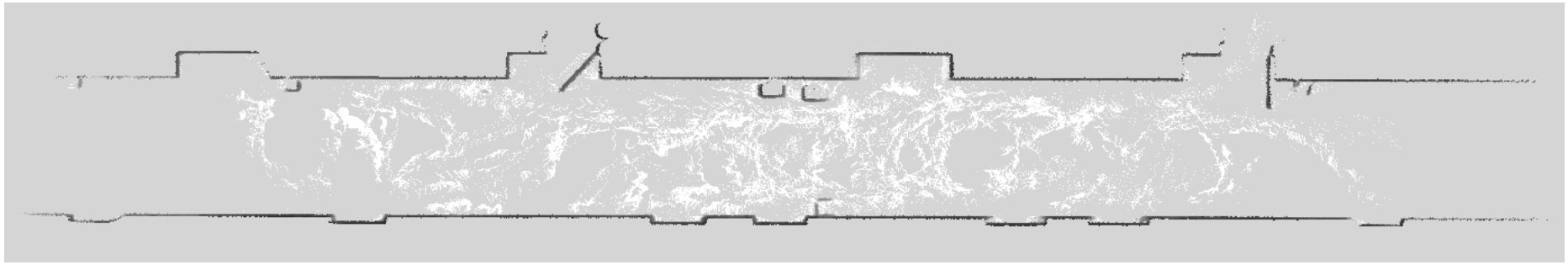
Wean Hall



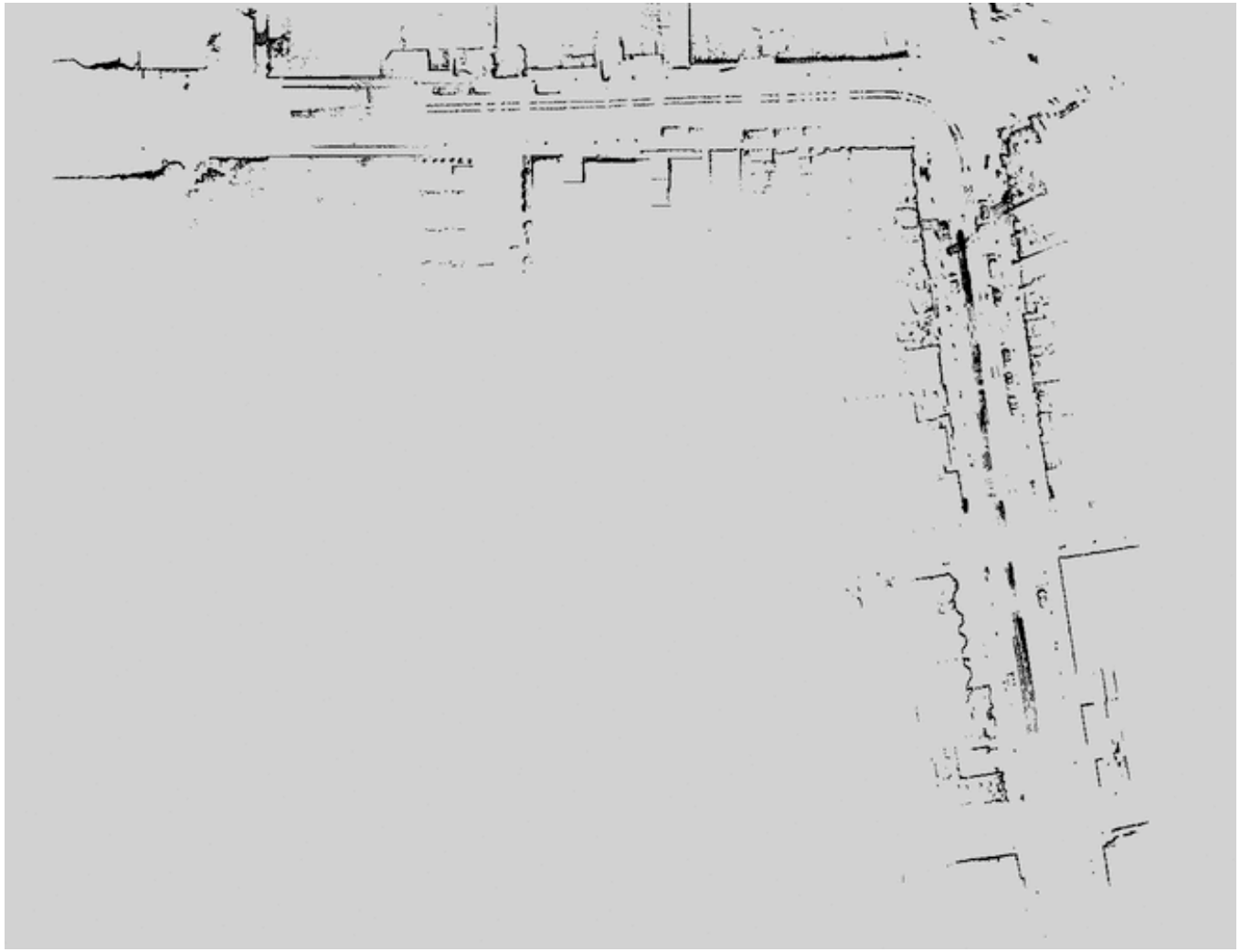
Wean Hall (Hallway)



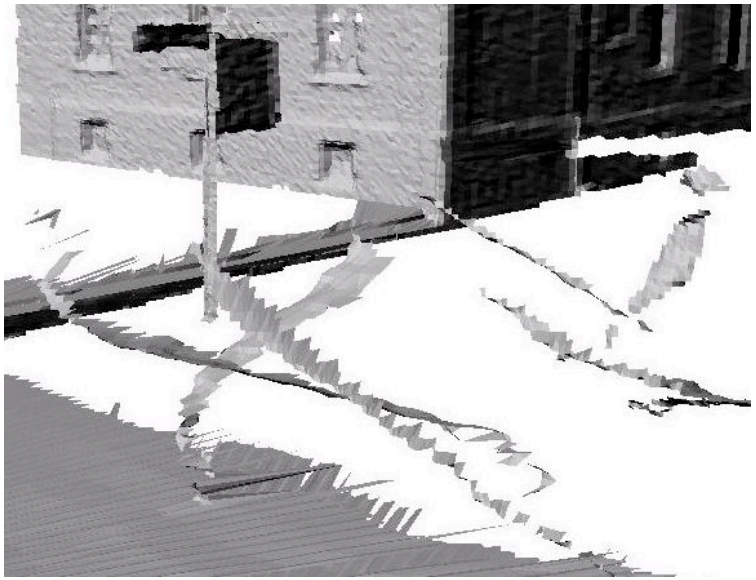
Wean Hall (Hallway)



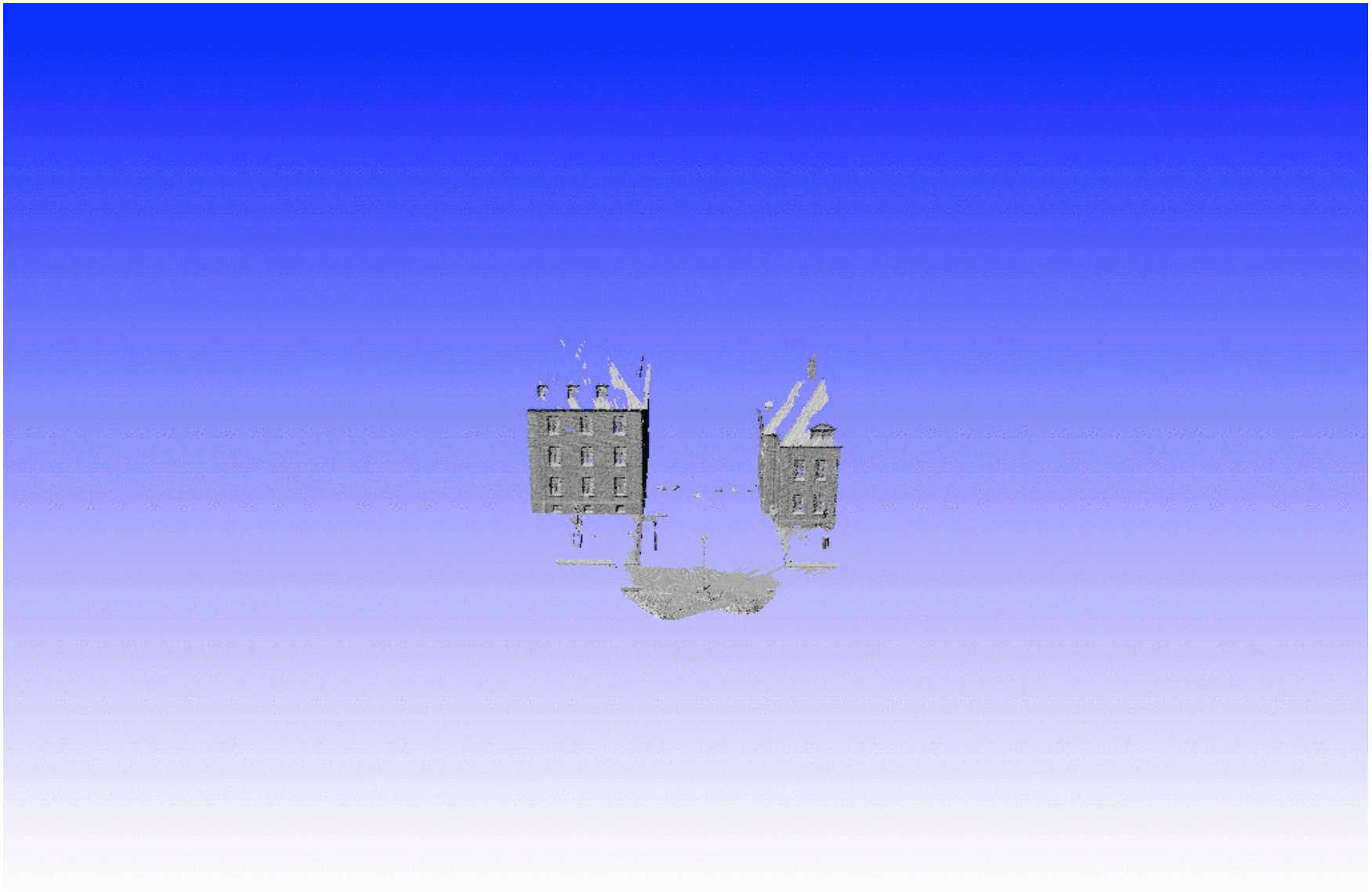
Pittsburgh
Craig Street/Forbes Ave



3D Maps in Dynamic Environments



A Fly-Through



Landmark Selection

Task:

Autonomous navigation in the context of embedded systems.

Problem:

- Simultaneous localization and mapping (SLAM) has high computational and memory demands.
- Embedded devices only provide limited computational power and memory capacity.

EKF-SLAM

- Given:
  - A sequence of observations $z_1, \ldots, z_t$
  - A sequence of actions $u_1, \ldots, u_t$
- Jointly estimate at each point in time t:
  - Pose of the robot: $(x_r, y_r, \theta_r)^t$
  - Positions of *N* landmarks:

$$(x_{l,1}, y_{l,1}, \ldots, x_{l,N}, y_{l,N})^t$$

- Here: Unscented Kalman Filter (UKF)

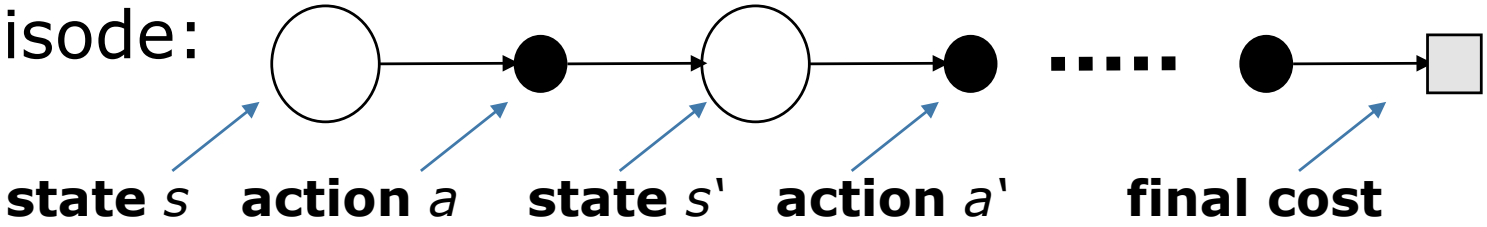
Scenario

- Limited computational resources.
  - Only few landmarks can be integrated.
- Solve navigation task as good as possible.
  - Reach target location as accurate as possible.

- Idea: Learn what to do.
  - Learn which landmark to consider.
- Desired properties:
  - Better than manually tuned heuristic.
  - Good generalization of learned policies.
  - Compress policies so they become applicable on systems with restricted memory capacity.

Reinforcement Learning

- Idea: Learn from interaction with the environment.
- Episode:



Estimate Q function which maps each state action pair to a cost value.

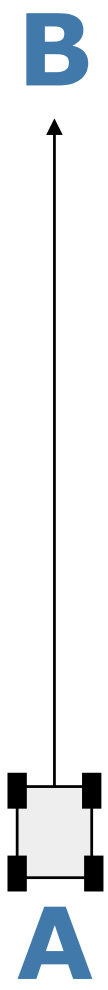
Choose policy π with $\pi(s) = \text{argmin}_a Q(s, a)$

Monte-Carlo Reinforcement Learning

- Estimate the Q-function as the average return over sample episodes.

Single Goal Task

- Robot should move from A to B.
- *N* landmarks in the environment.
- Only *M* landmarks can be used.

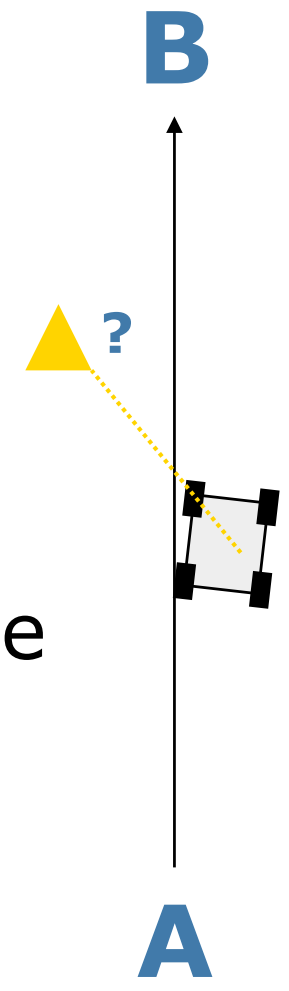


Learning target:

Select the optimal landmarks for the navigation such that the distance between true pose of the robot and the goal B is minimized.

Single Goal Task

- Robot should move from A to B.
- *N* landmarks in the environment.
- Only *M* landmarks can be used.

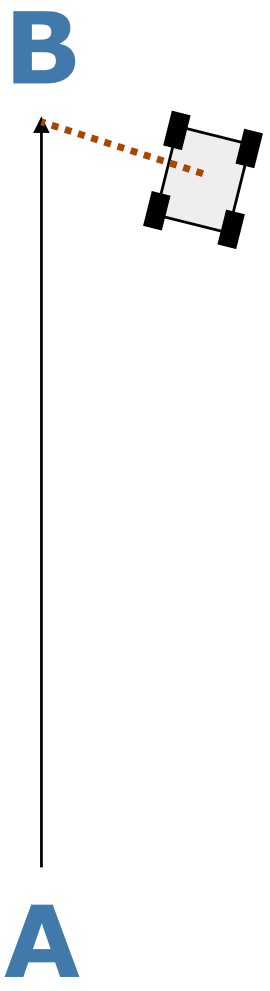


Learning target:

Select the optimal landmarks for the navigation such that the distance between true pose of the robot and the goal B is minimized.

Single Goal Task

- Robot should move from A to B.
- *N* landmarks in the environment.
- Only *M* landmarks can be used.

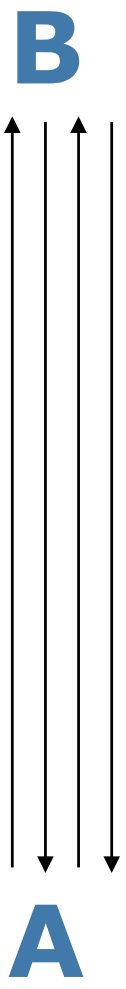


Learning target:

Select the optimal landmarks for the navigation such that the distance between true pose of the robot and the goal B is minimized.

Round Trip Task

- Robot should move from A to B and back to A twice.



Learning target:

Minimize the average distance between pose estimate and true pose over the whole trajectory.

→Focus on loop-closure.

State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

1. Estimated distance to the sub-goal B
2. Number of landmarks integrated in UKF

3. Yaw angle to potential new landmark
4. Distance of potential new landmark to closest landmark already integrated

5. (Entropy of the robot pose)

Binary action:

- Accept landmark / reject landmark

State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

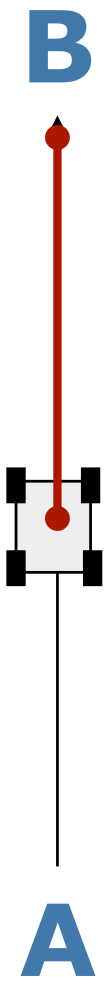
1. **Estimated distance to the sub-goal B**
2. Number of landmarks integrated in UKF

3. Yaw angle to potential new landmark
4. Distance of potential new landmark to closest landmark already integrated

5. (Entropy of the robot pose)

Binary action:

- Accept landmark / reject landmark



State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

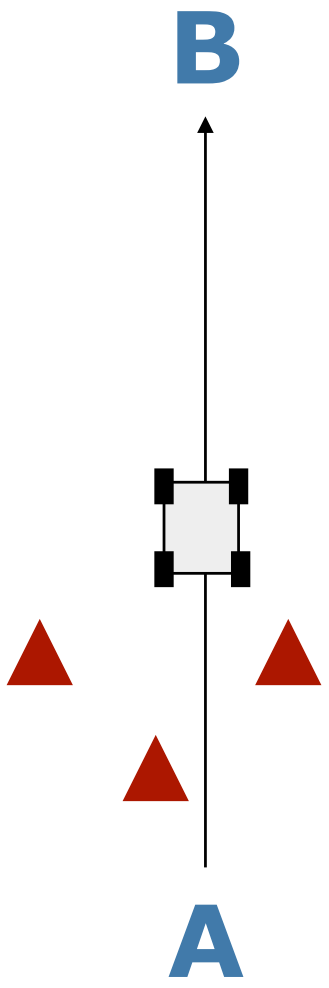
1. Estimated distance to the sub-goal B
2. **Number of landmarks integrated in UKF**

3. Yaw angle to potential new landmark
4. Distance of potential new landmark to closest landmark already integrated

5. (Entropy of the robot pose)

Binary action:

- Accept landmark / reject landmark



State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

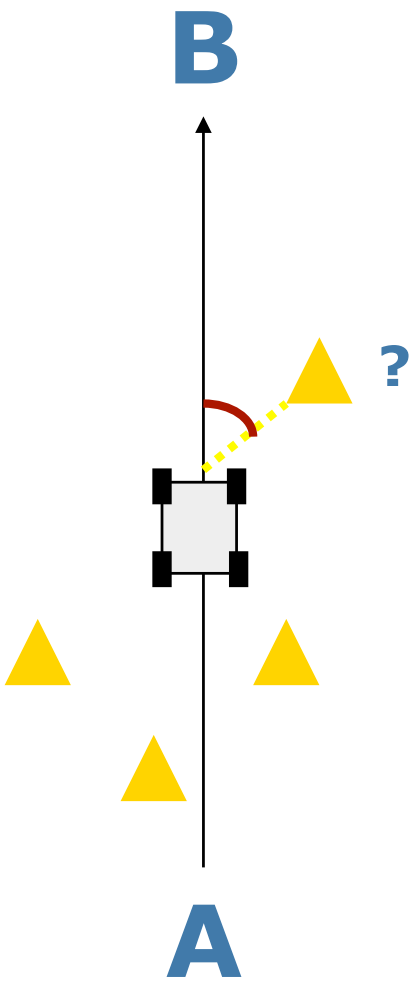
1. Estimated distance to the sub-goal B
2. Number of landmarks integrated in UKF

3. **Yaw angle to potential new landmark**
4. Distance of potential new landmark to closest landmark already integrated

5. (Entropy of the robot pose)

Binary action:

- Accept landmark / reject landmark



State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

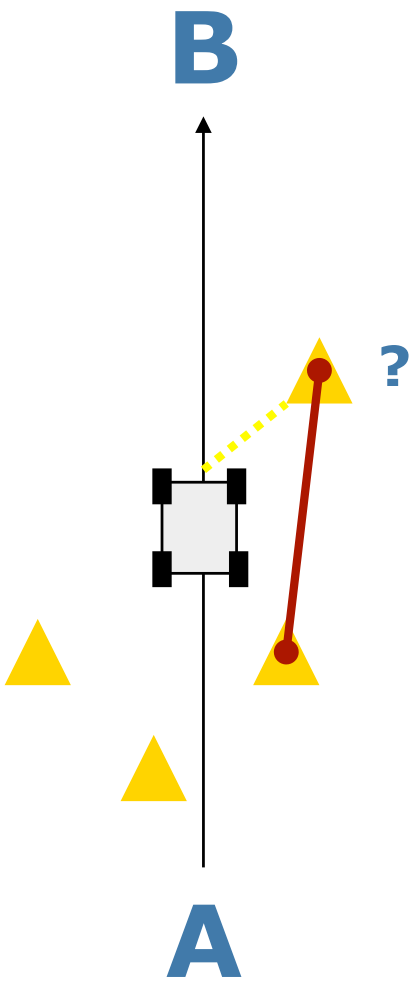
1. Estimated distance to the sub-goal B
2. Number of landmarks integrated in UKF

3. Yaw angle to potential new landmark
4. **Distance of potential new landmark to closest landmark already integrated**

5. (Entropy of the robot pose)

Binary action:

- Accept landmark / reject landmark



State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

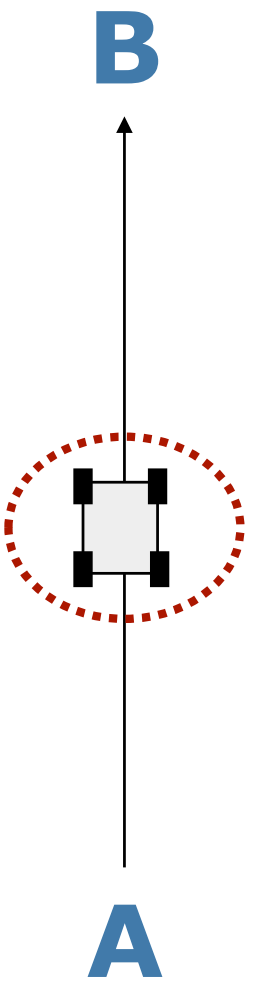
1. Estimated distance to the sub-goal B
2. Number of landmarks integrated in UKF

3. Yaw angle to potential new landmark
4. Distance of potential new landmark to closest landmark already integrated

5. **(Entropy of the robot pose)**

Binary action:

- Accept landmark / reject landmark



State and Action Space

Potential state space:

- Full UKF state (high-dimensional)

Relevant features:

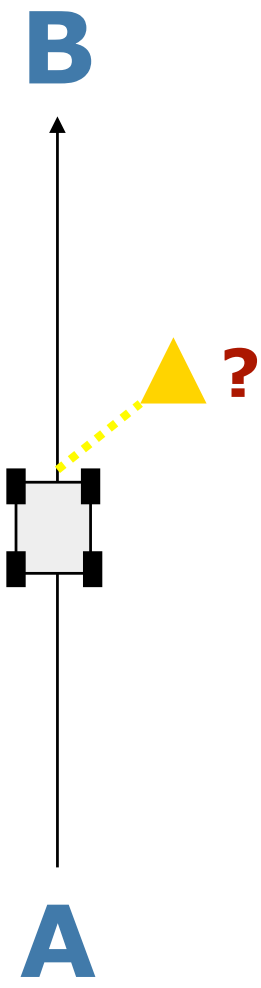
1. Estimated distance to the sub-goal B
2. Number of landmarks integrated in UKF

3. Yaw angle to potential new landmark
4. Distance of potential new landmark to closest landmark already integrated

5. (Entropy of the robot pose)

Binary action:

- **Accept landmark / reject landmark**



Representation of the Q-Function

Q: (s,a) → cost

k-Nearest Neighbor Regression:

- Store training data in a KD-tree.
- Find up to k nearest neighbors of query point (s,a) within a fixed search radius.
- Return mean of neighbors.

Properties:

→ Generalizes in sparse areas, and is accurate in dense areas.

→ No overfitting.

→ Model is expressed explicitly by the data (as Gaussian Processes), but very fast.

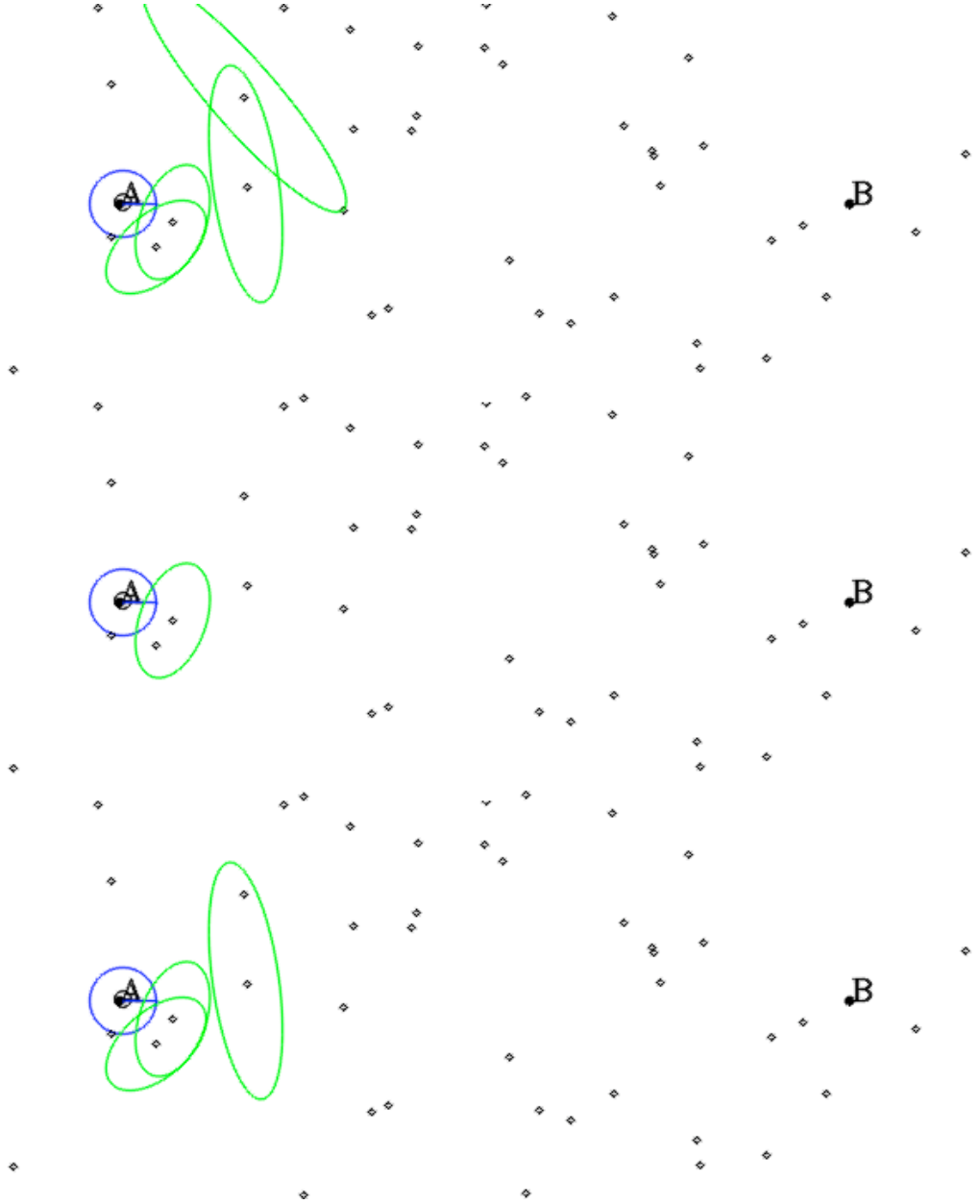
Qualitative Evaluation
Single Goal Task

Policies

M-First heuristic

Equidistant heuristic

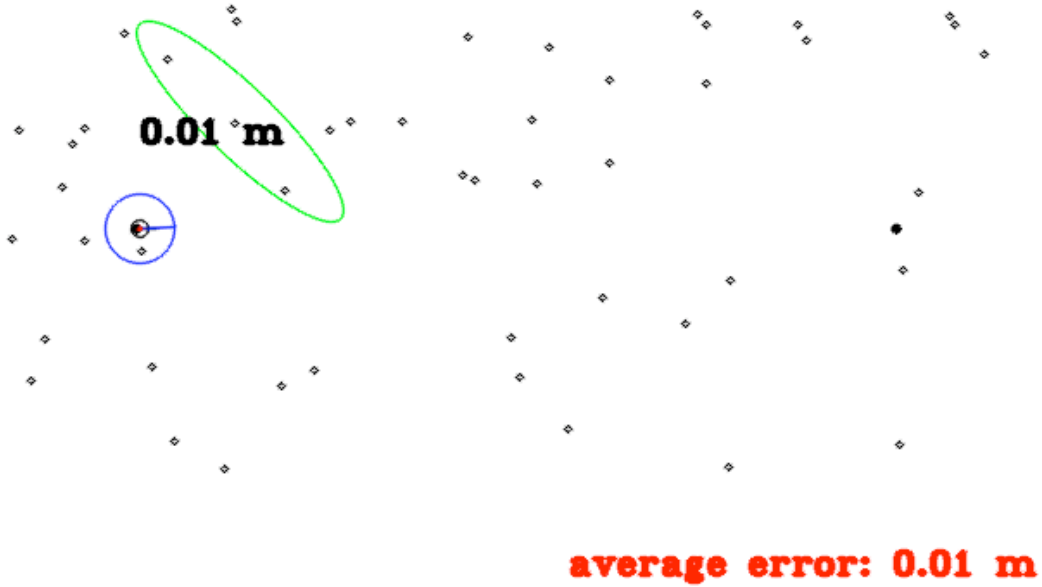
Learned policy



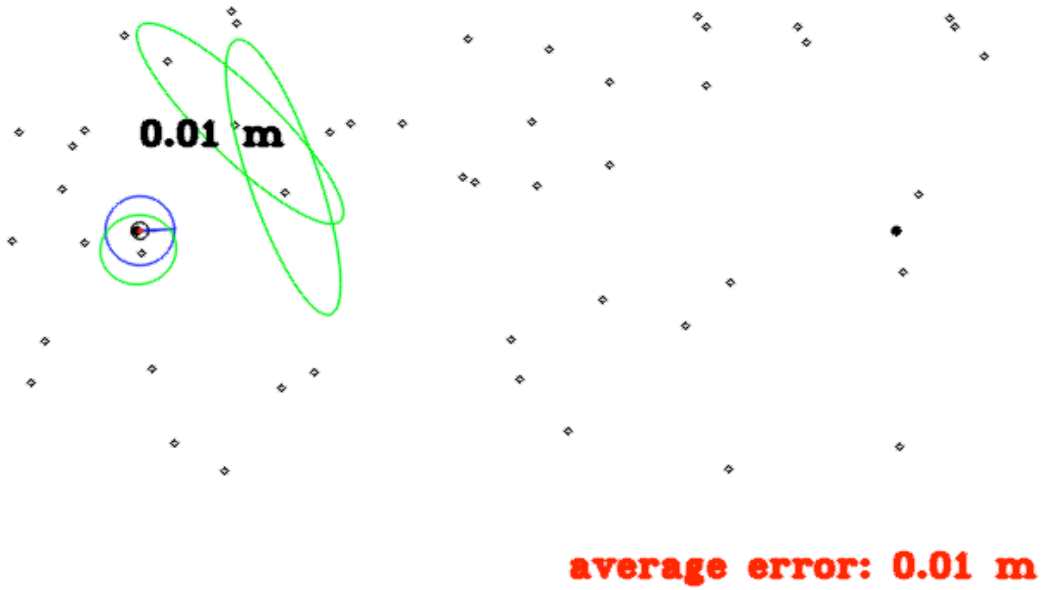
Qualitative Evaluation
Round Trip Task

Policies

Equidistant heuristic

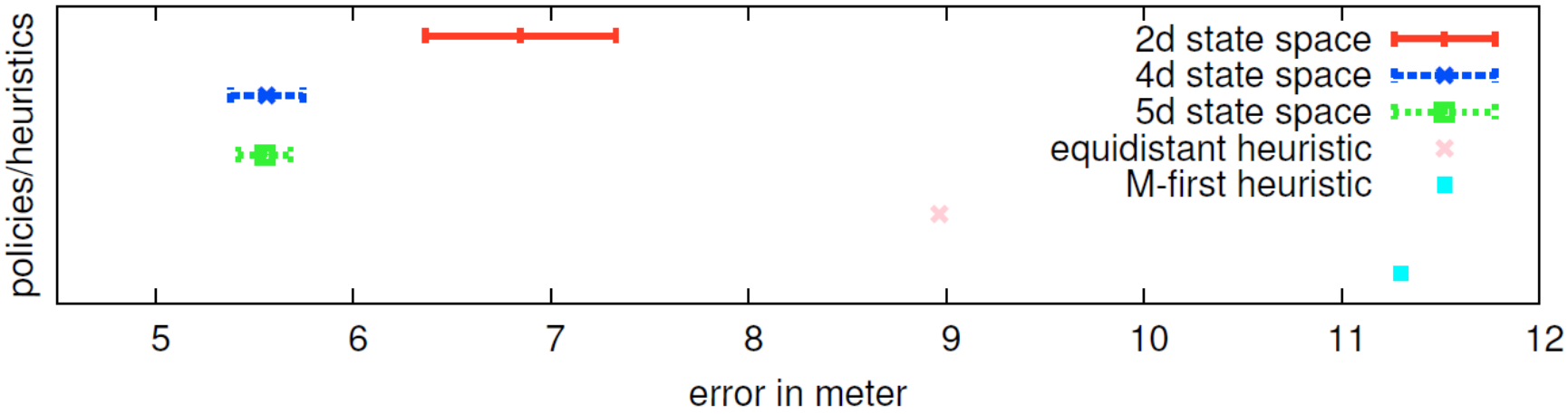


Learned policy



Quantitative Results

- 10 independent trainings run consisting of 2000 episodes.
- For each episode a new random environment is created.
- Each trained policy is tested in 1000 episodes.
- **Single goal task**: Learned policies lead to significantly smaller average errors compared to hand-tuned heuristics.



- **Round trip task**: Learned policies (± m) are significantly better than equidistant heuristic (m).

Generalization

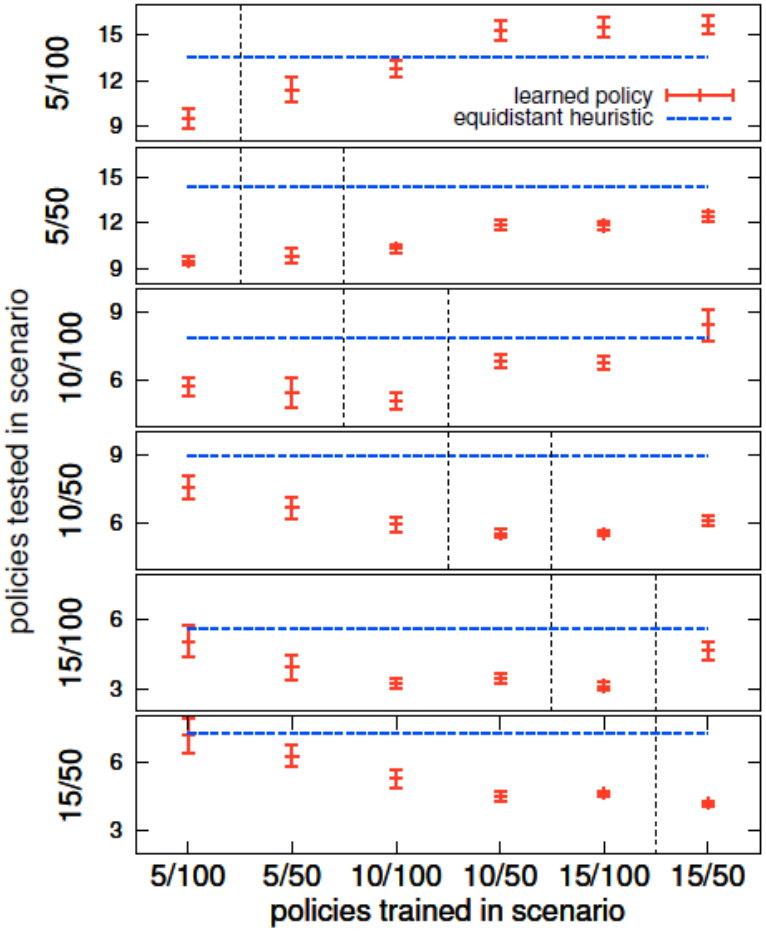
- So far we did learning and testing using the same setting.
- Desired property:
  - Learn the landmark selection policy in one training scenario.
  - Apply the learned policy successfully in other scenarios.

Crucial Parameters:

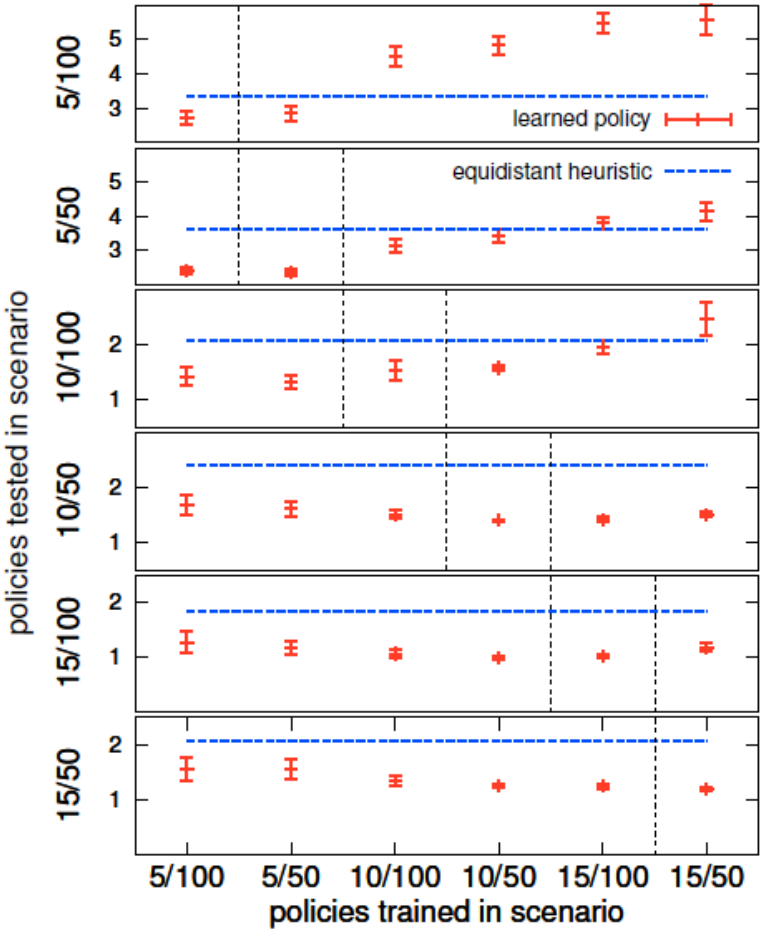
- *M* - the number of landmarks to select. (5-10-15)
- *N* - the landmark density in the environment. (50-100)

- Adjust state space accordingly:
  - E.g., percentage of landmarks already integrated.

High Degree of Generalization



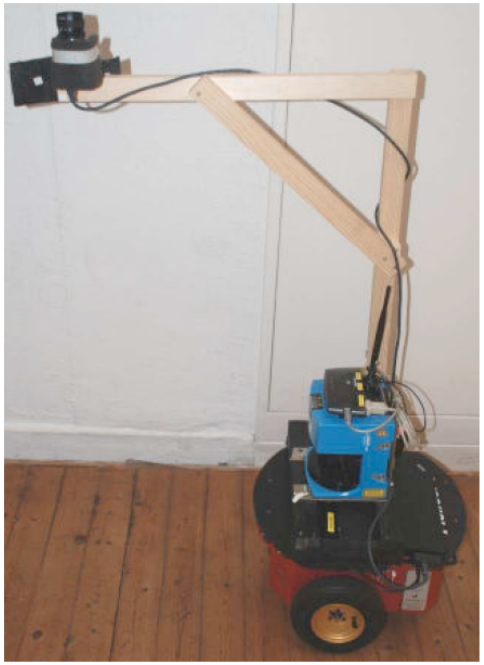
(a) Single-goal task



(b) Round-trip task

- Each policy (e.g., 5/50) is trained 10 times.
- Each trained policy is evaluated using 1000 episodes.

Real-Robot Experiment



Pioneer robot with upward looking camera

View of the robot: detected visual landmarks on the ceiling

Policy trained in simulation, tested on robot.

→Learned policy is significantly better than equidistant.

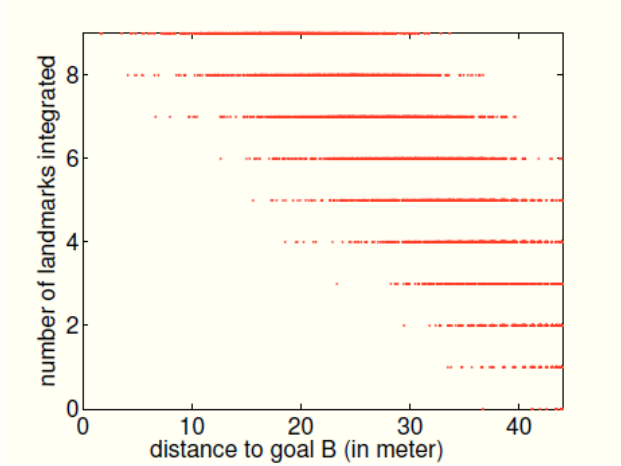
Average error: 0.50 m (learned) versus 0.66 m (equidistant)

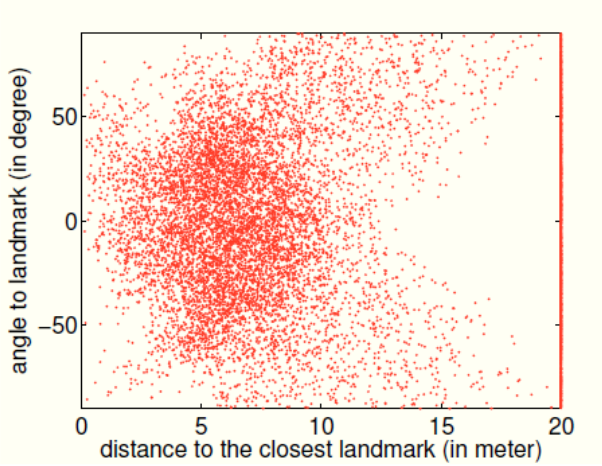
Uncompressed Policy

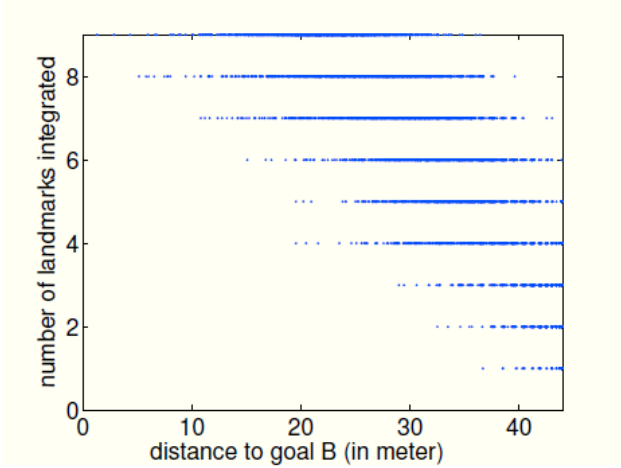
20.000 labelled data points of the uncompressed policy:

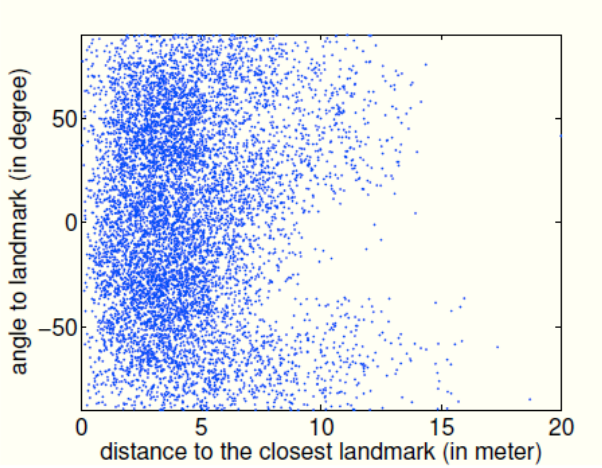
accept:

reject:







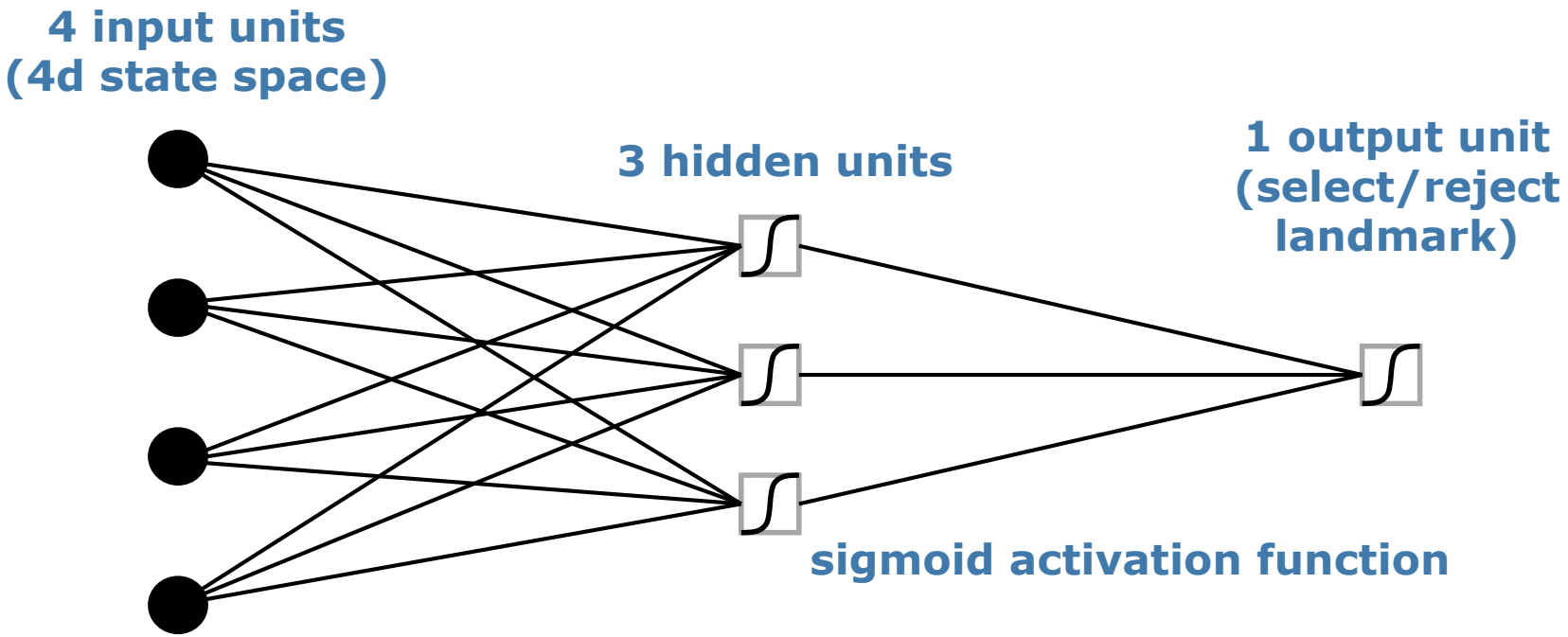


1st and 2nd dim.

3rd and 4th dim.

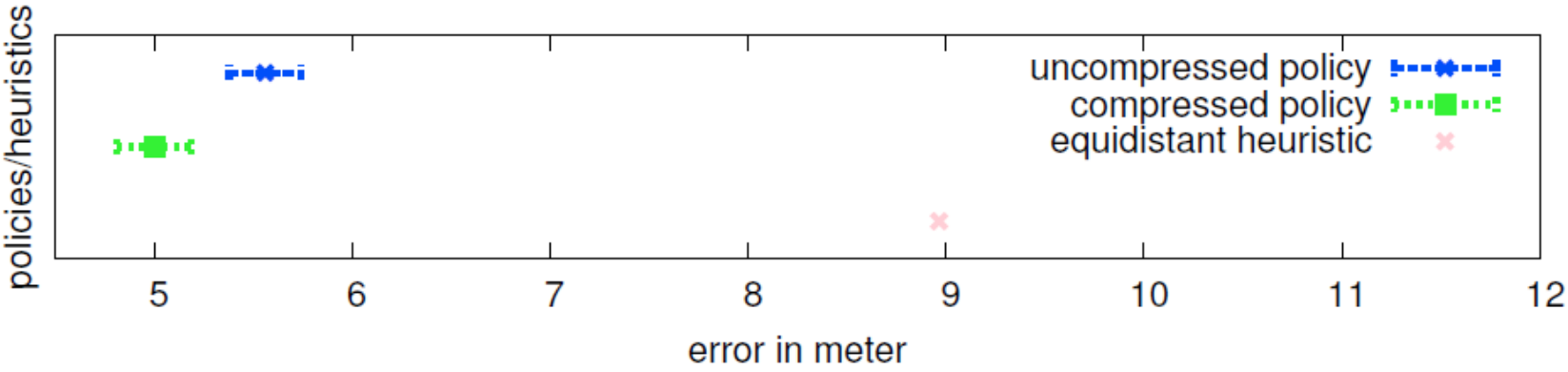
Policy Compression Using Neural Network Classification

- Neural network structure:

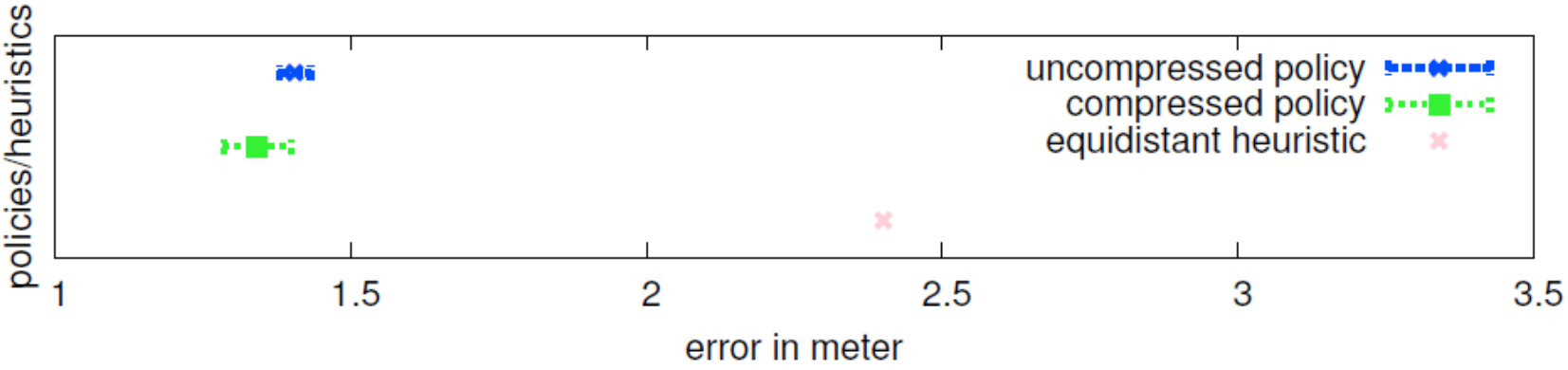


- 15 weights, 4 bias parameters

Comparison: Compressed Versus Uncompressed Policies



Single-goal task

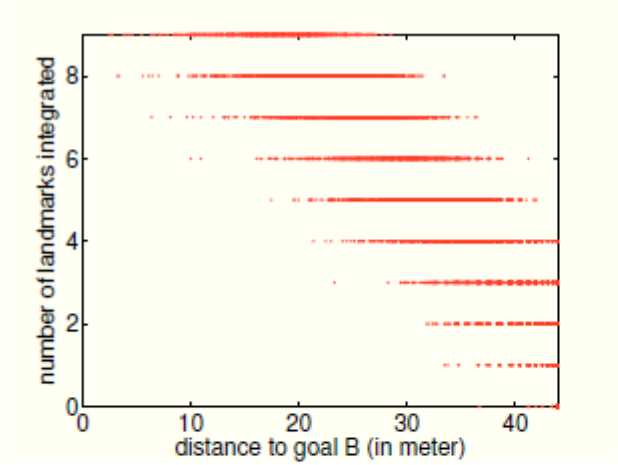


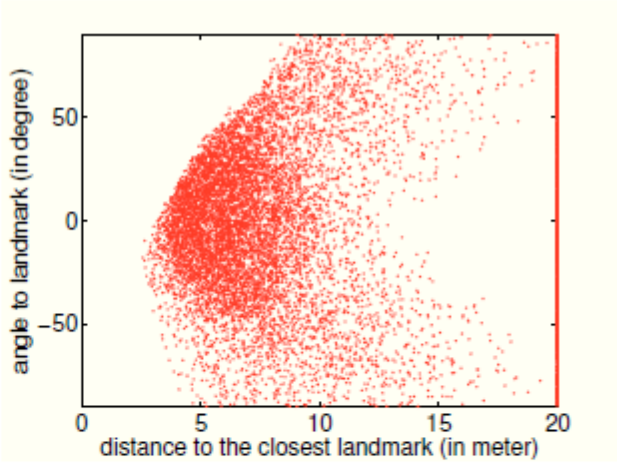
Round-trip task

Compressed Policy

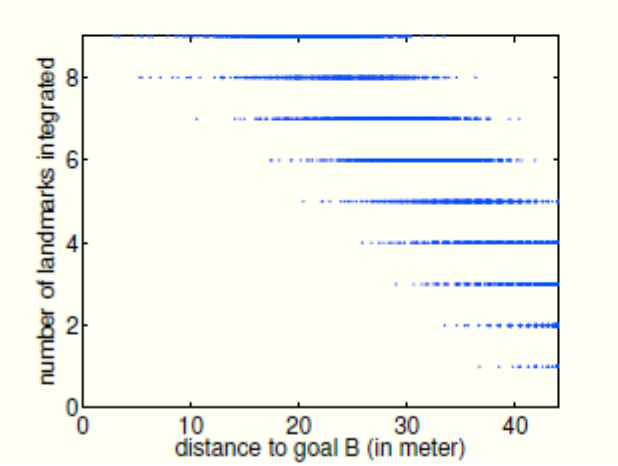
20.000 labelled data points created by the neural network:

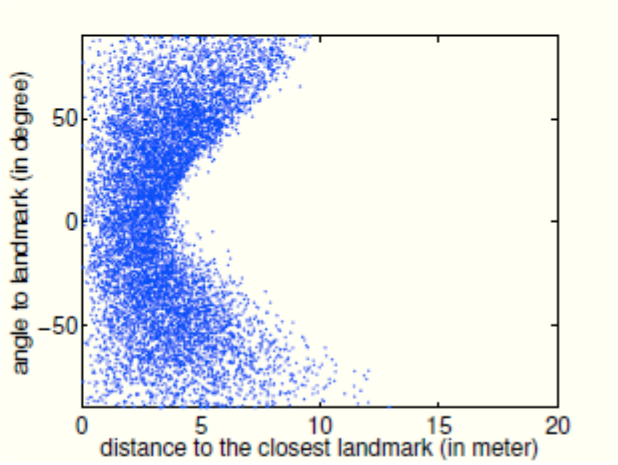
accept:





reject:





1st and 2nd dim.

3rd and 4th dim.

Deletion of Landmarks?

- Idea: Marginalize out old landmark in order to cope with limited memory.
- Deletion leads to a more complex problem since the required state and action space increases significantly.
  - Which old landmark should be replaced by the new one?
- First investigations reveal that the following selection policy is promising:
  - Keep some landmarks fixed for re-localization.
  - Perform incremental pose correction with set of frequently replaced landmarks.
- More research has to be done.

Conclusion

- Novel approach for landmark selection in the context of autonomous navigation.
- Important for robots with limited resources.
- The learned policies are significant better then manually designed strategies such as the equidistant heuristic.
- Learned policy generalizes to new environment (with different properties).
- Landmark selection policy learned in simulation has successfully been applied on a real robot.

Robustness towards Shorter/ Longer Paths?

- We can mimic longer/shorter trajectories with by the help of the landmark capacity *M* and landmark density *N*.
  - If the path length is doubled the, we simple double *M* and *N*.
- Limitation: We did not adapt the sensor and motion noise accordingly.
  - How robust is the approach towards different sensor/motion noise?
  - More research has to be done here!

Robustness towards Changes in Sensor and Motion Noise?

- We applied policy learned in simulation successfully on a real robot.
- We estimated the motion and sensor noise parameter only very roughly.

→Small indication that the approach is somehow robust towards changes in noise.

- But: More research has to be done here!
  - Inclusion of noise specific features in the state space might increase the robustness…

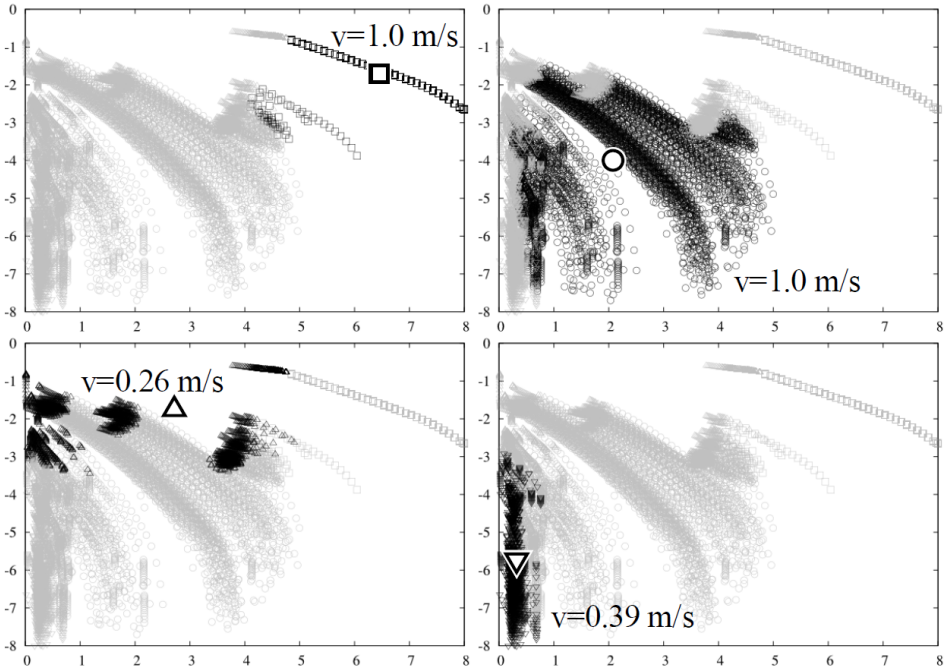
Action Selection

- Variant of ε–greedy to boost the training:

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) & \text{if } Q(s,a_{accept}) \geq Q(s,a_{reject}) \\ & \text{and } \hat{A}_1 < 1 - \varepsilon \\ a_{accept} & \text{if } [Q(s,a_{accept}) = Q(s,a_{reject}) \text{ or } \hat{A}_1 \cdot \varepsilon] \\ & \text{and } \hat{A}_2 < \frac{M}{N_{visible}} \\ a_{reject} & \text{else} \end{cases}$$

Compact Representation

- K-NN is very fast, but the memory requirements is linear to the number of training data.
- Use approach that Armin Hornung, Maren Bennewitz, and I applied for a similar learning task: Efficient visual navigation in known environment.
  - Problem: Some areas in the state space/action are visited very rarely (the values are “nonsense”.)
  - Idea: Express state space by the state/action values visited in 1000 episode using the learned policy.
  - Use clustering technique.



Localization Using a Camera

Advantages of cameras:

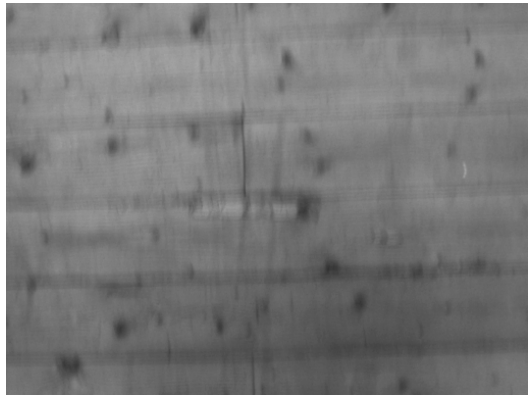
- Provide valuable information
- Compact
- Lightweight

⇒ Ideal sensor for humanoid robots and UAVs

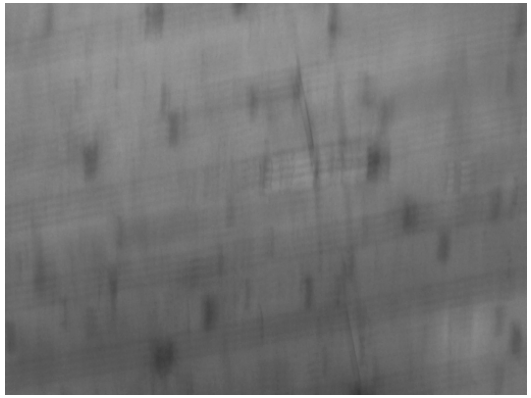
Problem of Vision-based Localization

- Fast movements of the robot introduce motion blur in the images

v=0.05 m/s



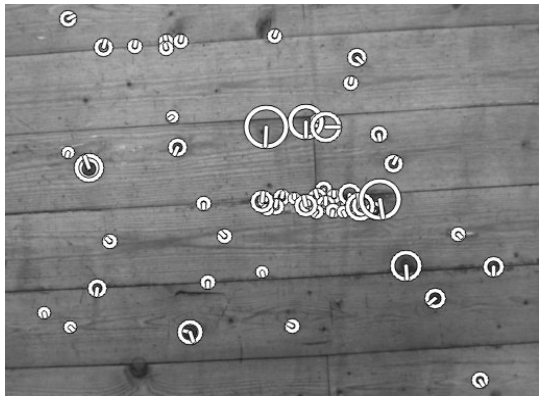
v=0.4 m/s



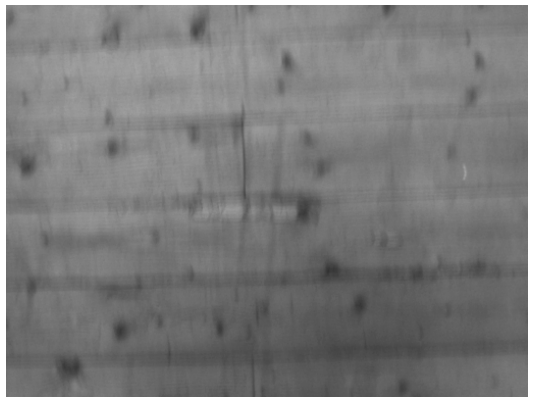
v=1.0 m/s

Problem of Vision-based Localization

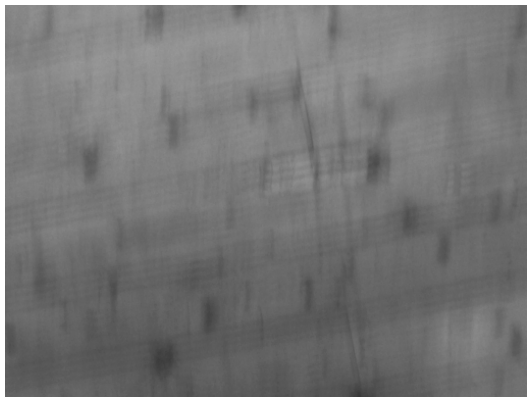
- Fast movements of the robot introduce motion blur in the images



v=0.05 m/s



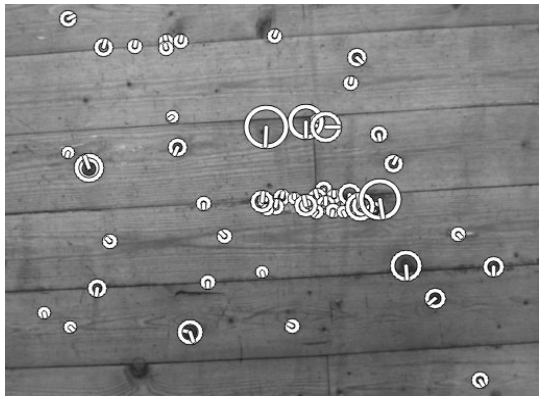
v=0.4 m/s



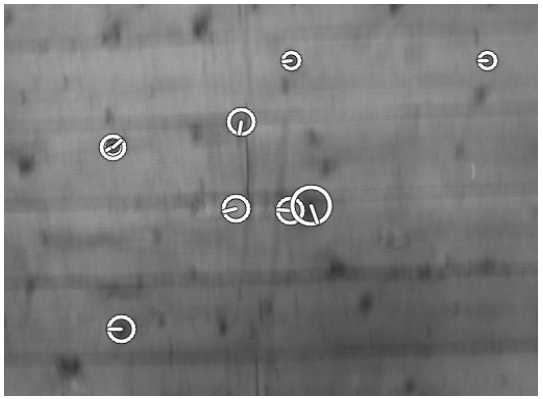
v=1.0 m/s

Problem of Vision-based Localization

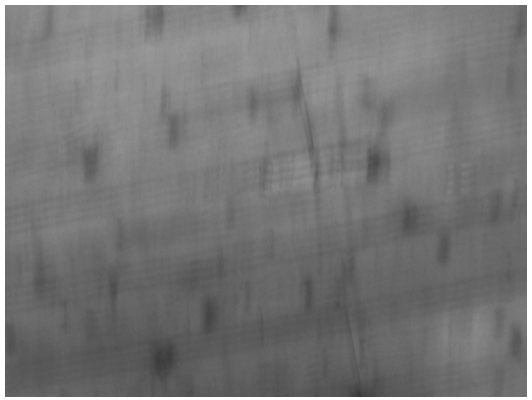
- Fast movements of the robot introduce motion blur in the images



v=0.05 m/s



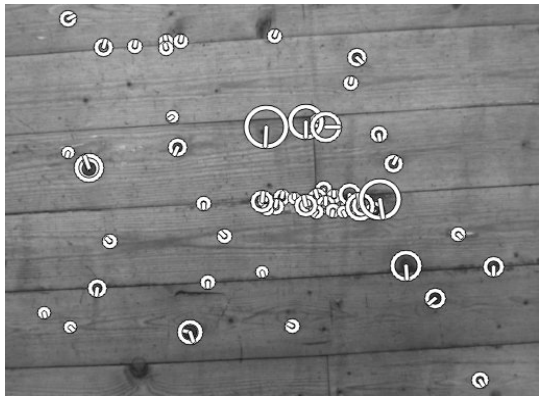
v=0.4 m/s



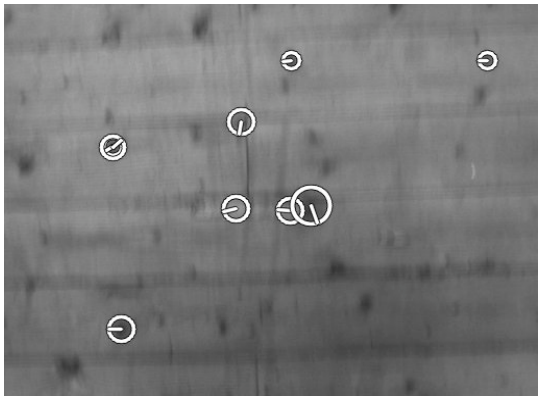
v=1.0 m/s

Problem of Vision-based Localization

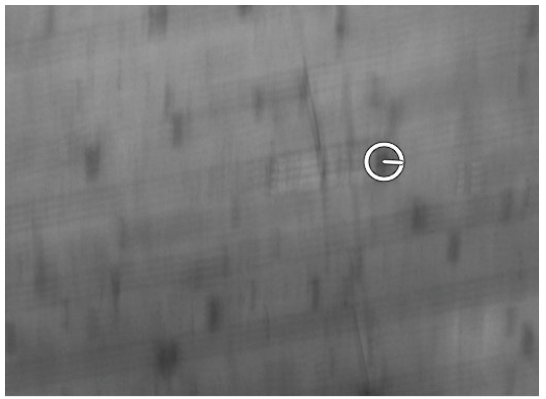
- Fast movements of the robot introduce motion blur in the images



v=0.05 m/s



v=0.4 m/s



v=1.0 m/s

Problem of Motion Blur

- Features cannot be extracted and matched reliably anymore
- Degradation depends on
  - Camera quality, shutter speed
  - Lighting conditions
  - Velocity of the robot
- Image preprocessing possible, but does not completely restore the image

Approach

- Let the robot learn how to control its motion in order to reach the destination fast and reliably

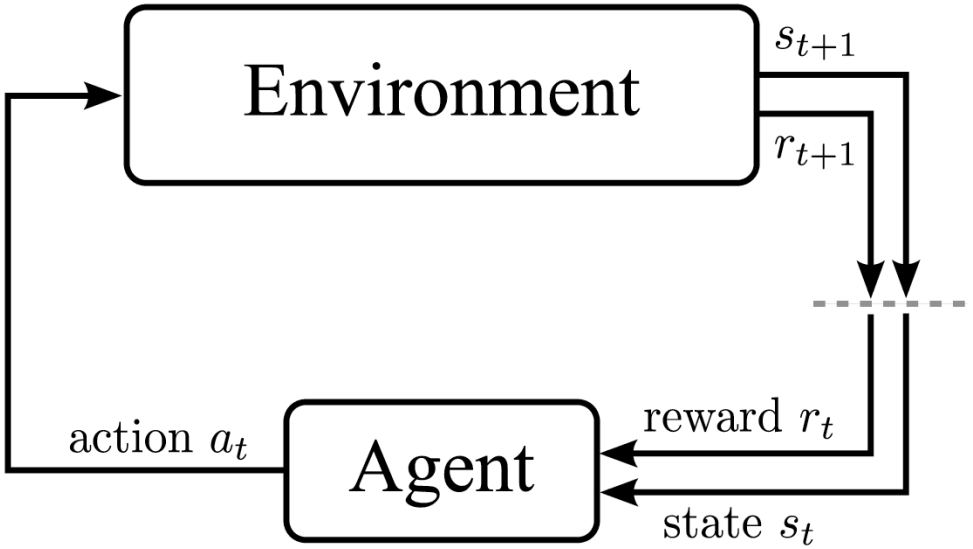
- Take the influence of motion blur into account to generate an efficient navigation policy

Task Description

- Navigation from the start to a goal point (or to an intermediate waypoint)
- Localization based on visual features
- UKF to track the robot’s pose
- Task is finished when reaching the destination (physically)
- Goal: Learn a navigation strategy which minimizes the time to the destination

Learning Navigation Policies

- Formulation as reinforcement learning task
- Defined as Markov decision process (MDP) using the states *S*, the actions *A*, and the rewards *R*



- In our case: Augmented MPD

State Space

- Use relevant features to represent the complete state
  - Uncertainty of UKF, entropy over the Gaussian

    $h = \frac{1}{2} \ln \left((2\pi e)^3 \cdot |\det(\Sigma)| \right)$
  - Estimated Euclidean distance to the goal (g_x, g_y)

    $d = \sqrt{(g_x - x_t)^2 + (g_y - y_t)^2}$
  - Estimated relative angle to the goal

    $\varphi = atan2(g_y - y_t, g_x - x_t) - \theta_t$
- Based on the most-likely pose estimate

Actions

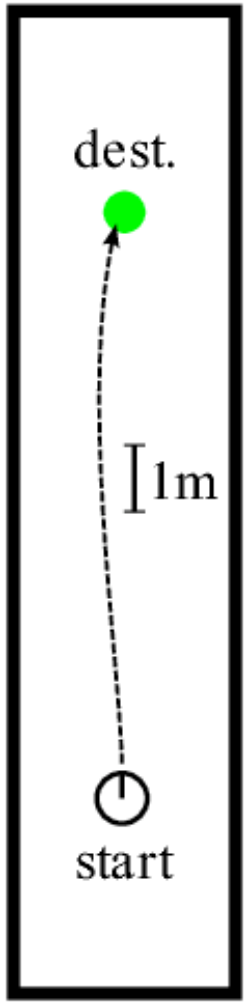
- Choosing the velocity *v*
- Possible actions
  A={0.1,0.2,0.3,0.4,1.0} in m/s
- Discretization determined according to the effect of motion blur
- The selected velocity *v* is sent to the navigation controller
- Note: The collision avoidance stops the robot in dangerous cases

Rewards

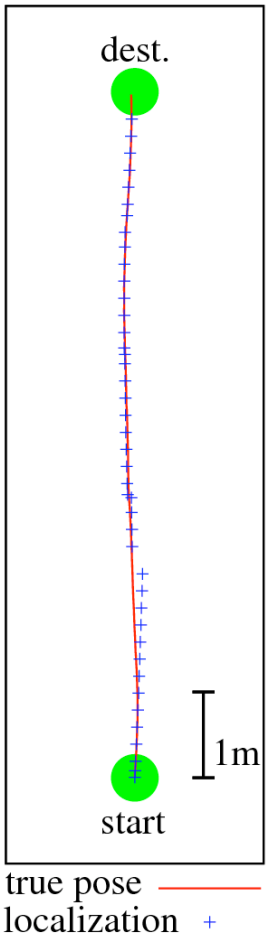
- Reward at t: $$r_t = \begin{cases} 100 & \text{if } t = T \\ -\Delta_t & \text{otherwise} \end{cases}$$
- Drives the robot to reach the destination as fast as possible
- No explicit punishment for delocalization
- Implicit punishment: Time to stop, re-localize, and accelerate again

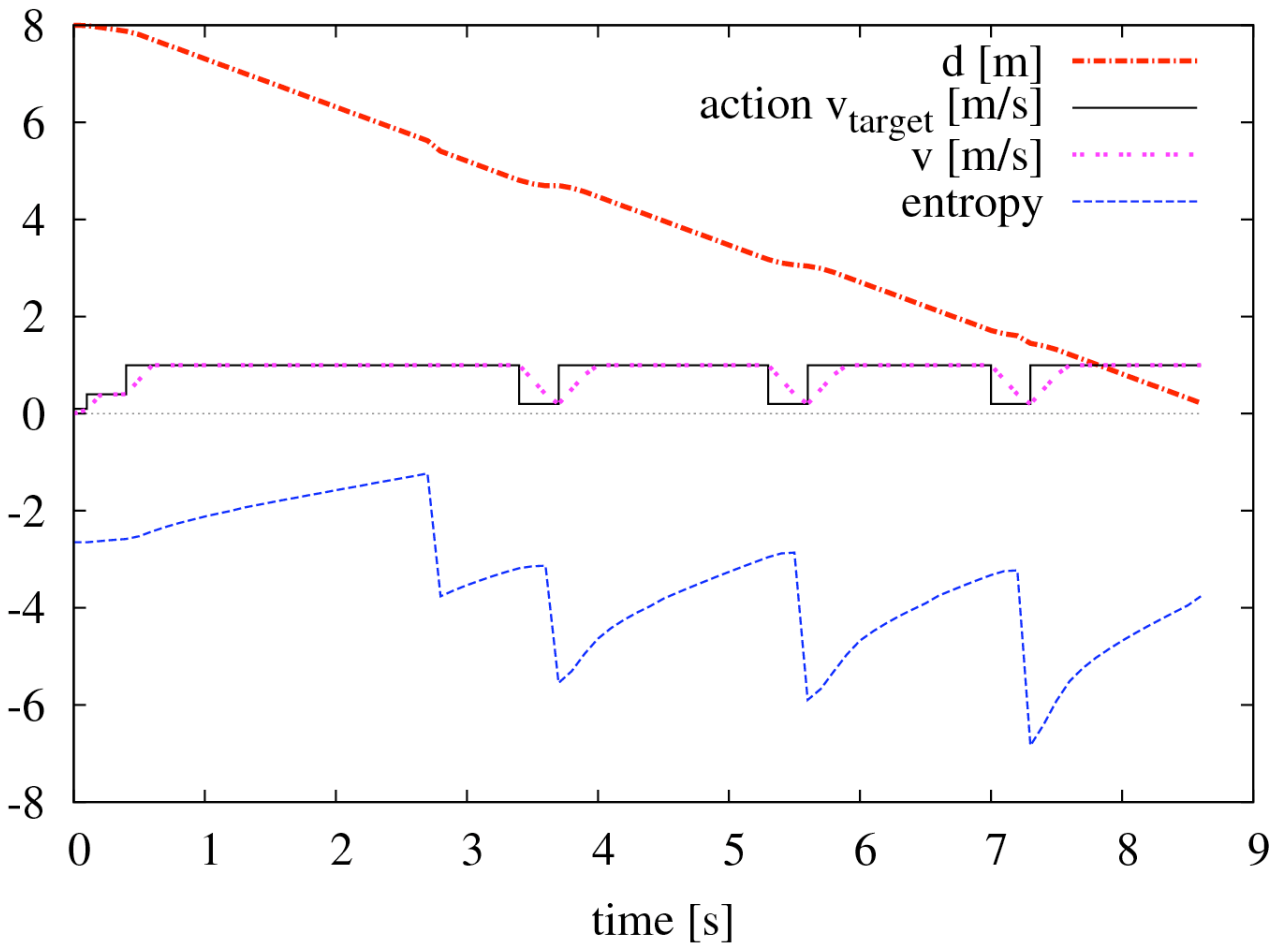
Experiments

- Learning and parameter evaluation in simulation
- The policy is learned in a scenario with 2 waypoints
- The landmark observation probability is estimated from real data
- Each learning run: 500 episodes
- The landmark positions are randomized in each episode (landmark density: 40 L/ m^2)

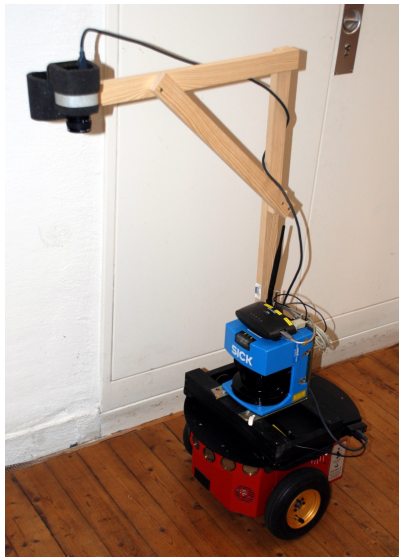


Trajectory of the Learned Policy (in Simulation)

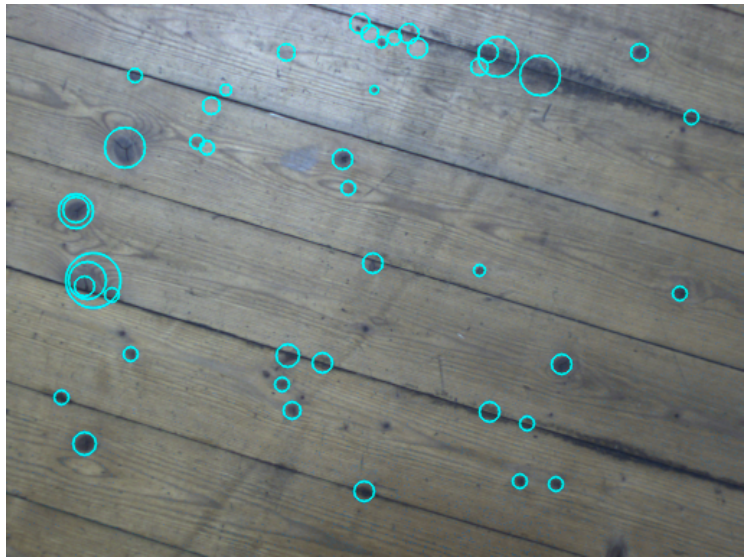




Indoor Robot



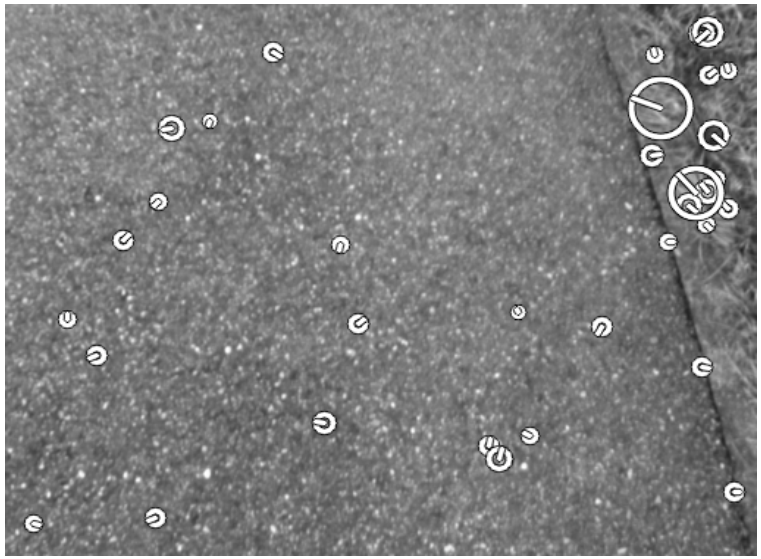
Pioneer 2 robot with downlooking camera



Observed floor patch with SURF as visual landmarks

Outdoor Robot

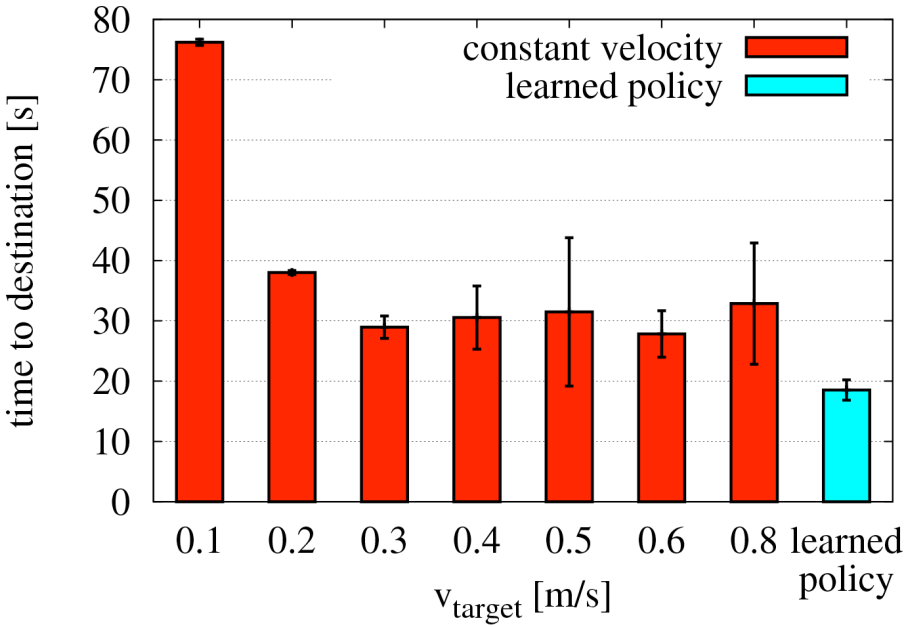
Powerbot robot with downlooking camera



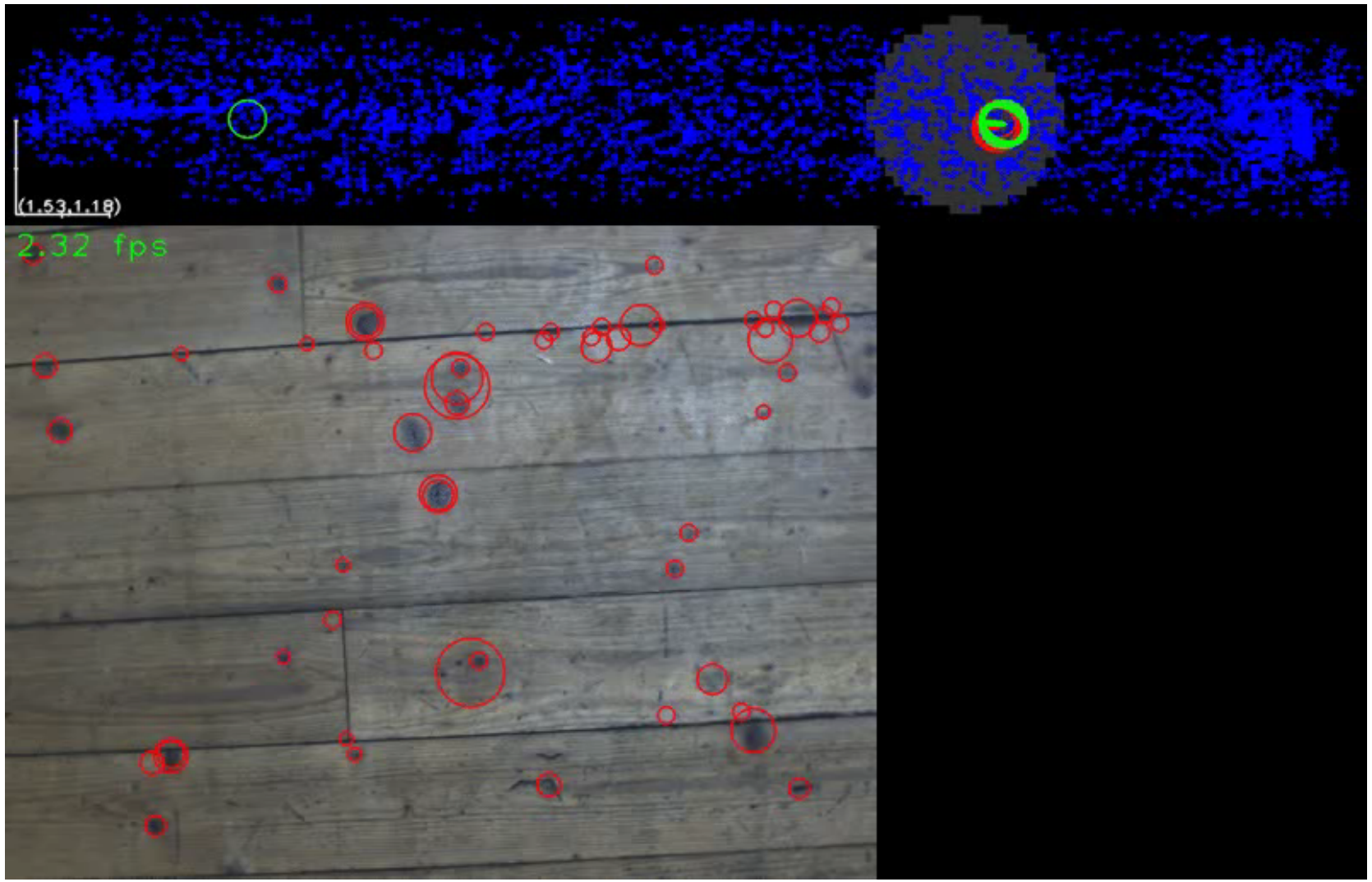
Observed floor patch with SURF as visual landmarks

Evaluation - Indoor Robot

- Two-waypoint scenario
- Evaluation: 10 runs of the learned policy
- Compared to driving at constant velocity (10 runs for each velocity)
- **Our learned policy is significantly faster!**

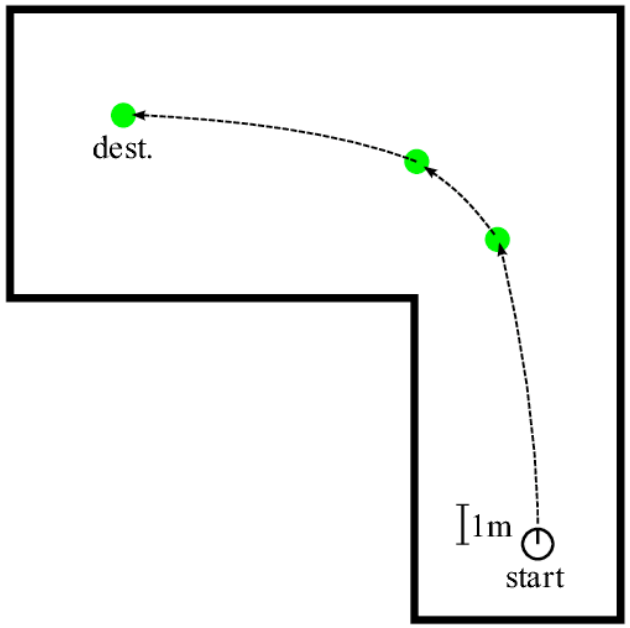


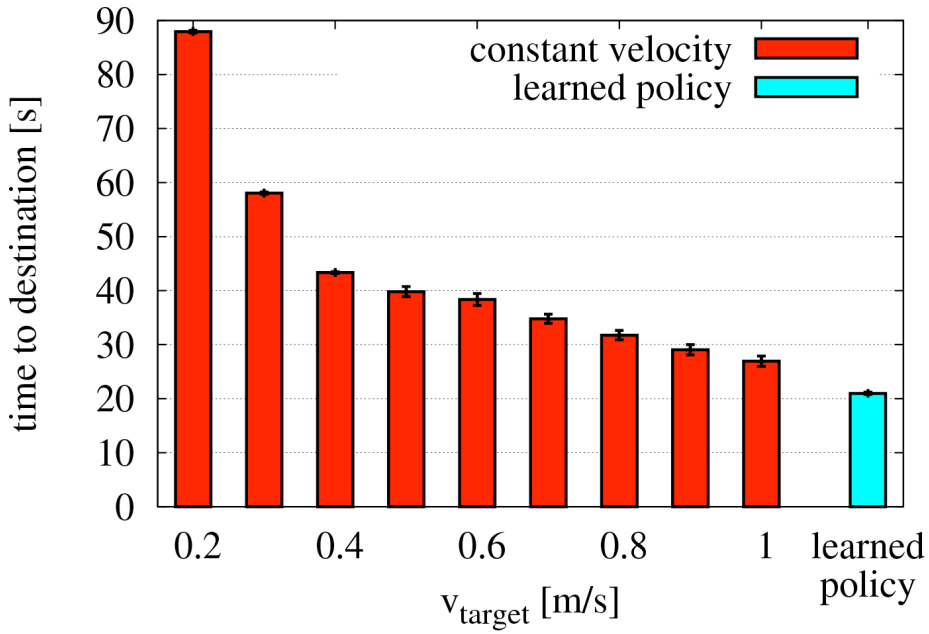
Learned Policy – Indoor Robot



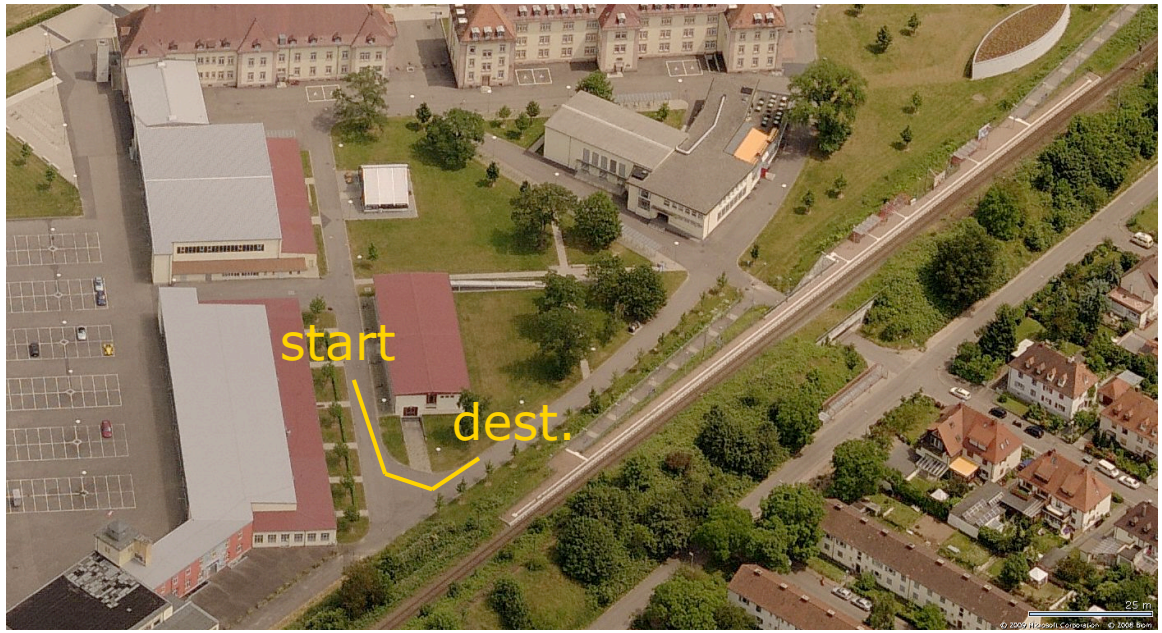
Scenario with Multiple Waypoints Simulation Results

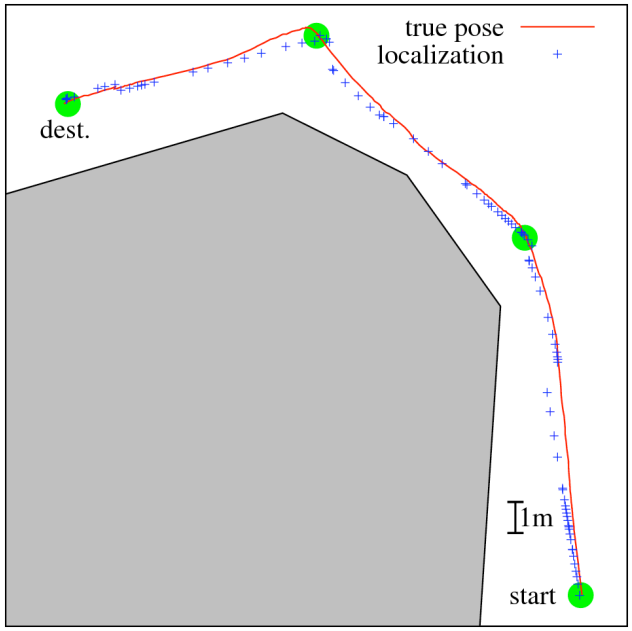
- When the pose estimate is sufficiently close to the waypoint, the next waypoint is regarded as goal



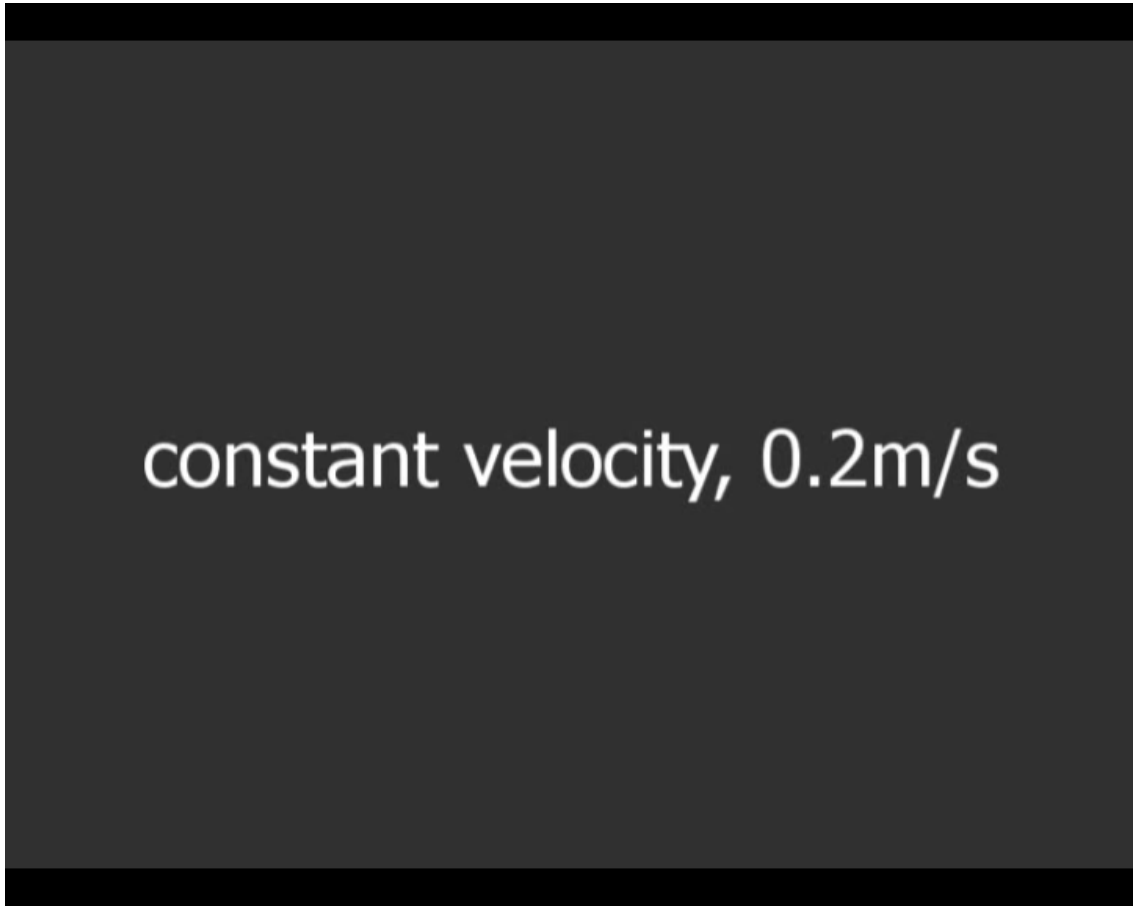


Learned Policy – Outdoor Robot

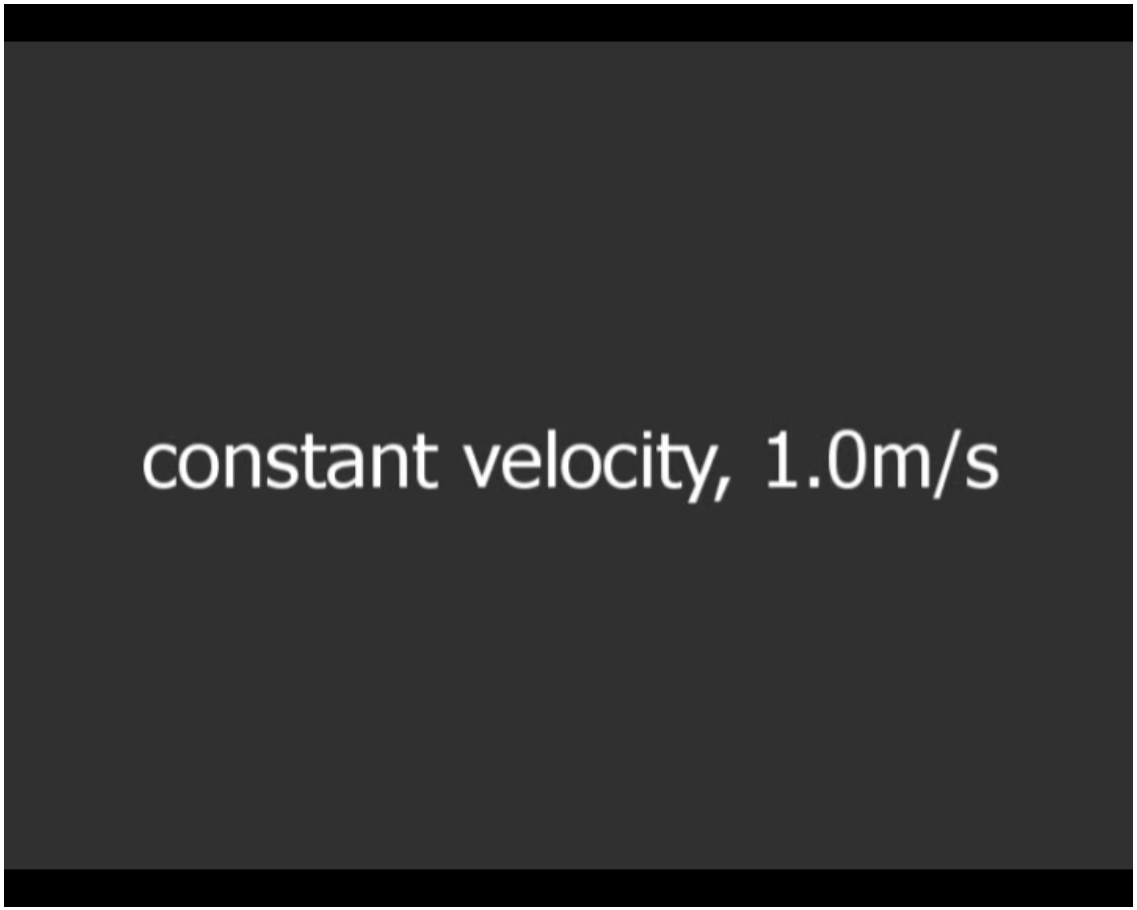




Learned Policy – Outdoor Robot



Learned Policy – Outdoor Robot

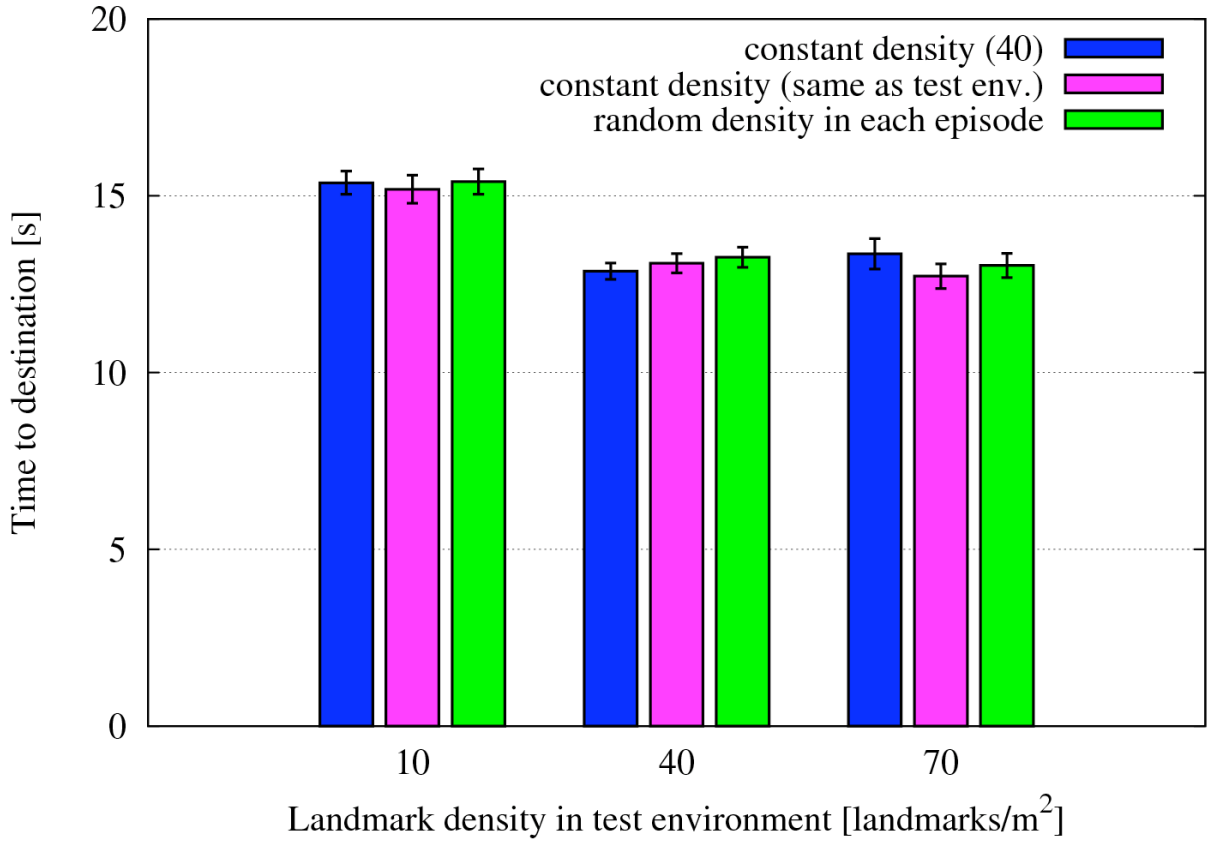


Learned Policy – Outdoor Robot

Generalization over the Landmark Density

- Evaluate the policy learned in an environment with a landmark density of 40 L/m^2 in
  - sparser (10 L/m^2) and
  - denser (70 L/m^2) environments
- Compare the performance to a policy learned in an environment matching the respective test environment

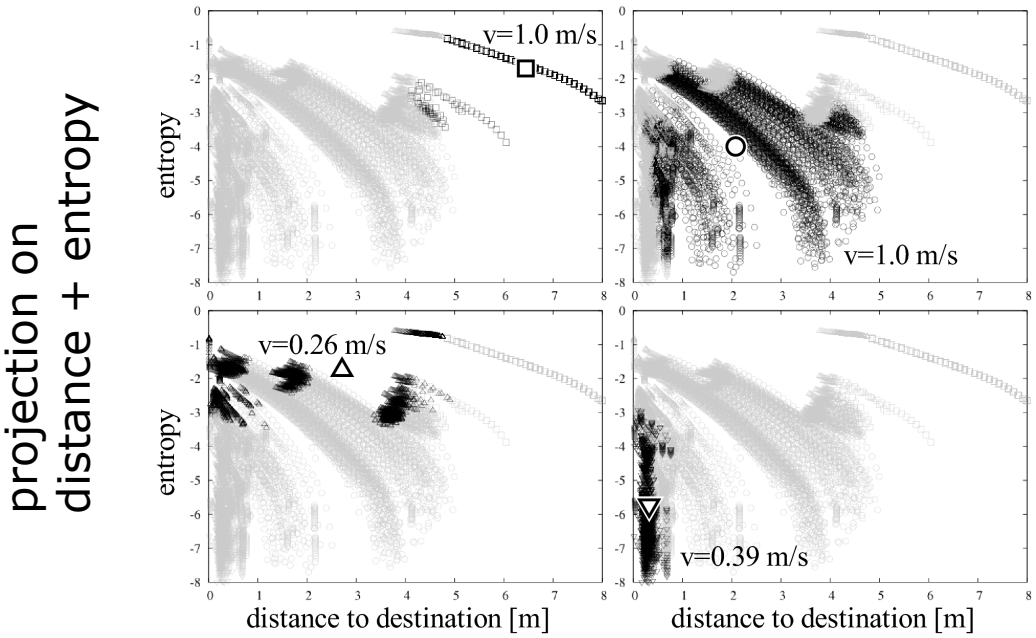
Generalization over the Landmark Density



- No need to explicitly account for environmental conditions in the learning scenario since no significant difference!

Policy Compression

- Learned policy is represented by a table whose size depends on the discretization
- For each visited state (d,φ,h), the chosen velocity v is regarded as classification
- Apply X-Means (k-Means+BIC) to find the number of clusters and their centers



- **Significantly more compact representation**
- **Yields a similar performance**

Conclusions (1)

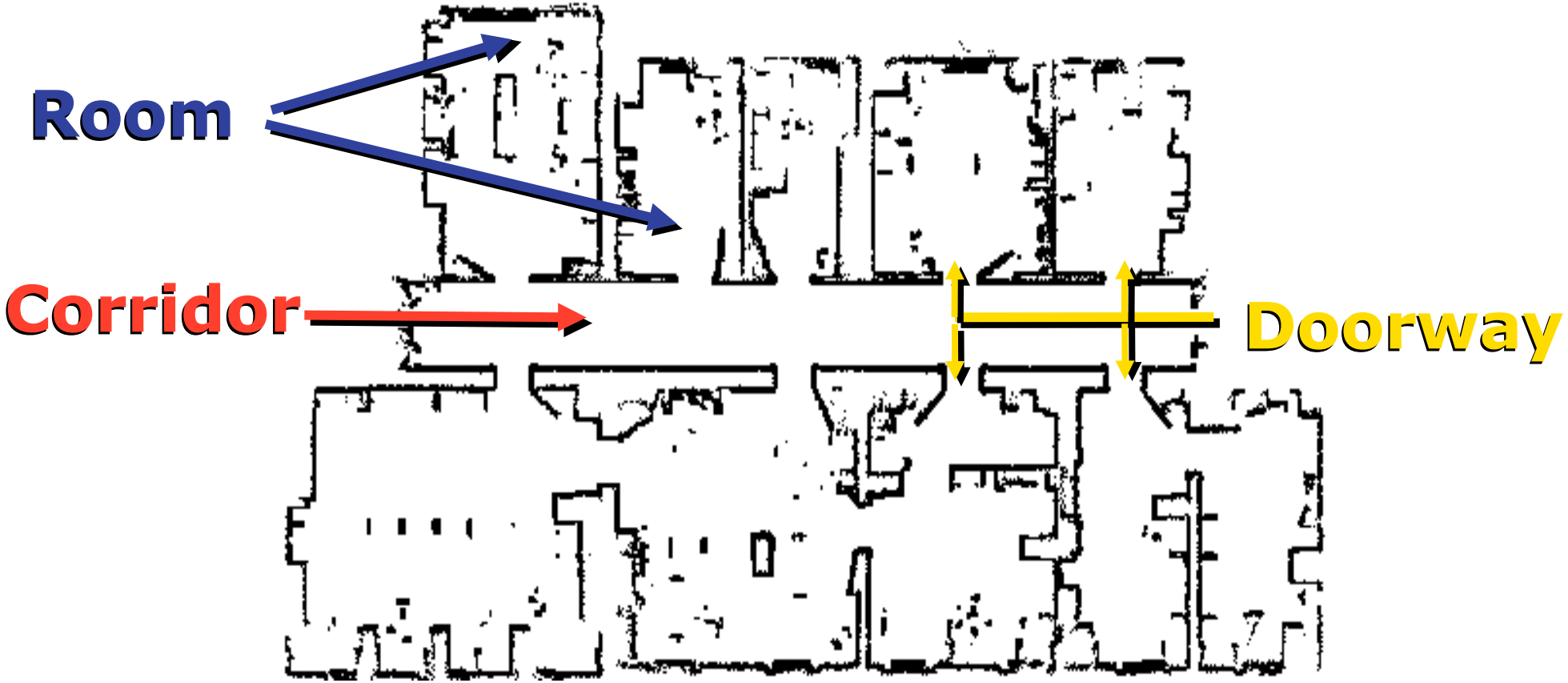
- New approach to learning an efficient policy for vision-based navigation
- Via reinforcement learning, the robot learns to reach its destination as fast as possible
- The impact of motion blur on the feature detection is implicitly taken into account
- Our learned policy outperforms any strategy of driving at a constant velocity

Conclusions (2)

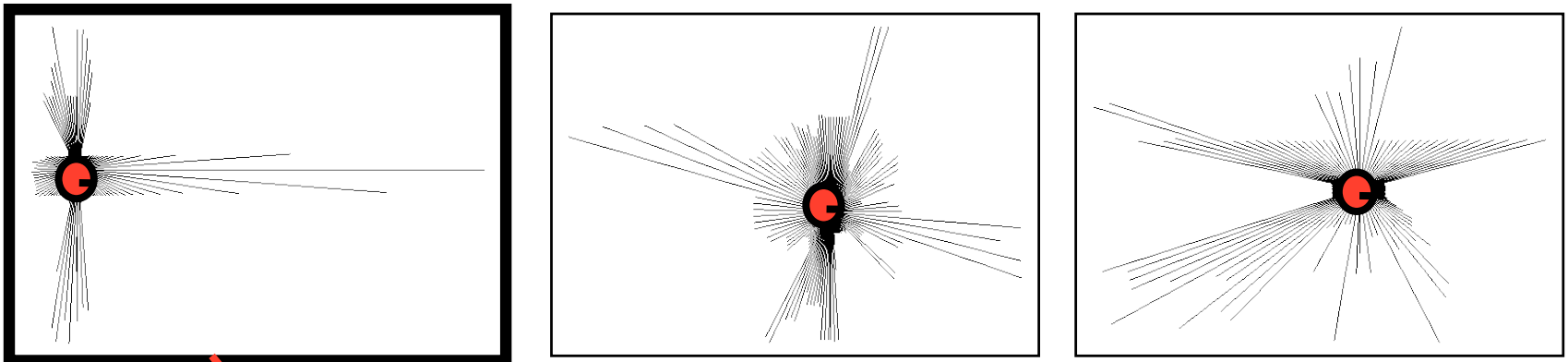
- The learned policy generalizes over different environments
- X-means clustering on the visited state space for compressing the learned policy
- Compressed policy is an order of magnitude smaller without a loss of performance

Place Classification

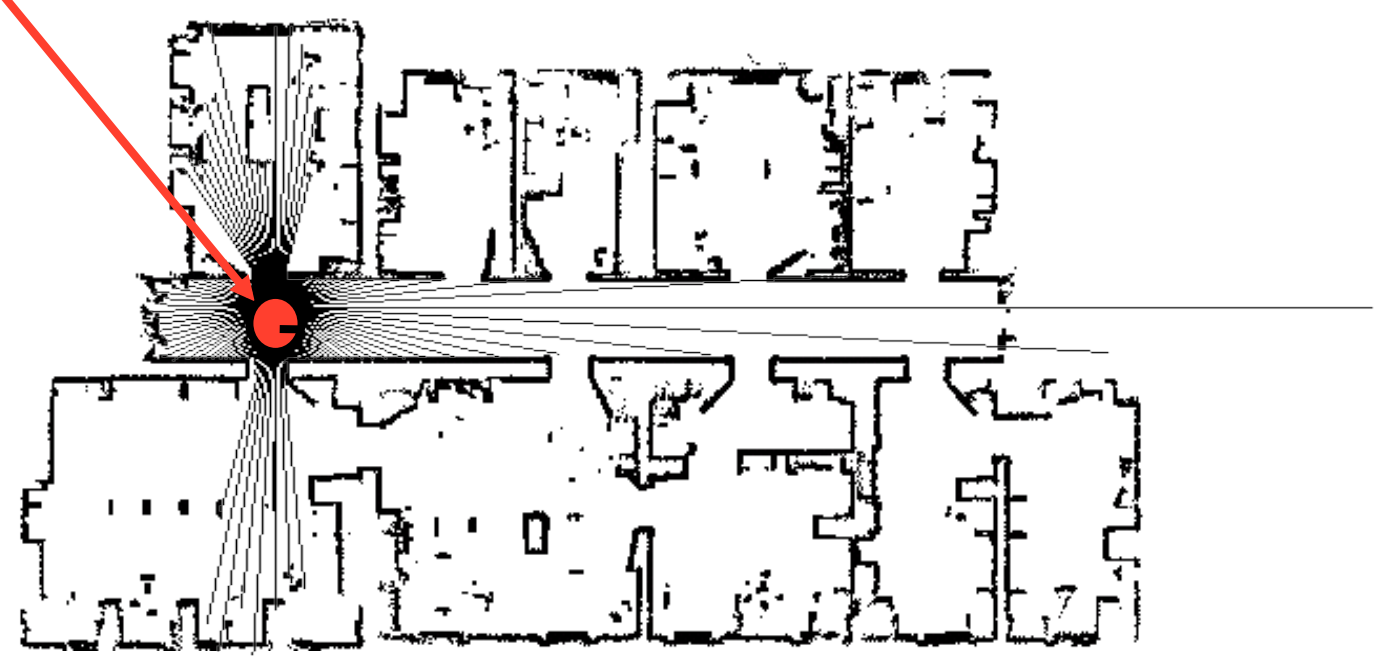
- Indoor environments can be decomposed into different places
- How can we **determine** the type of place a robot is at and how can we **utilize** it to improve navigation tasks?



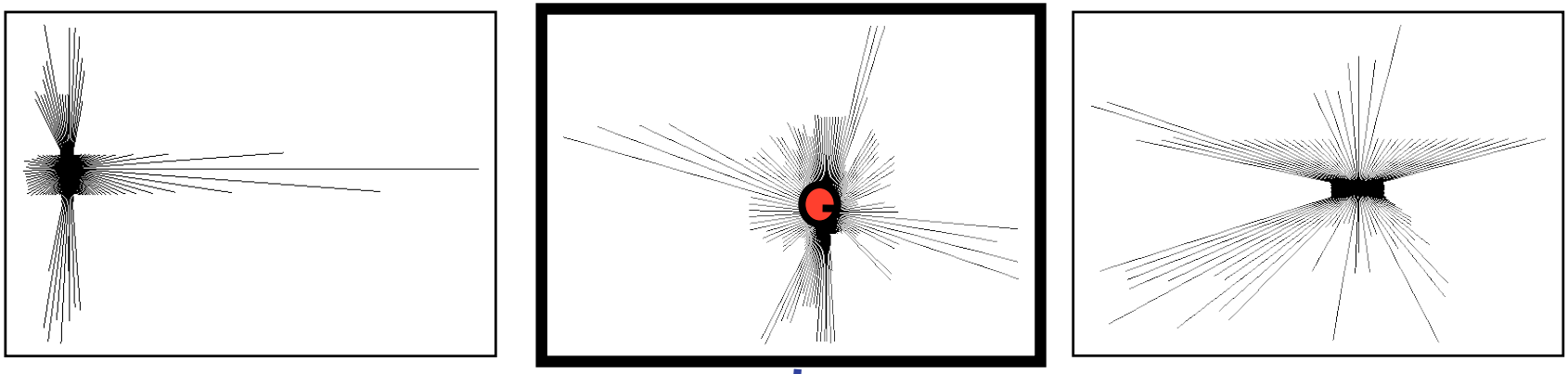
Observations



Corridor



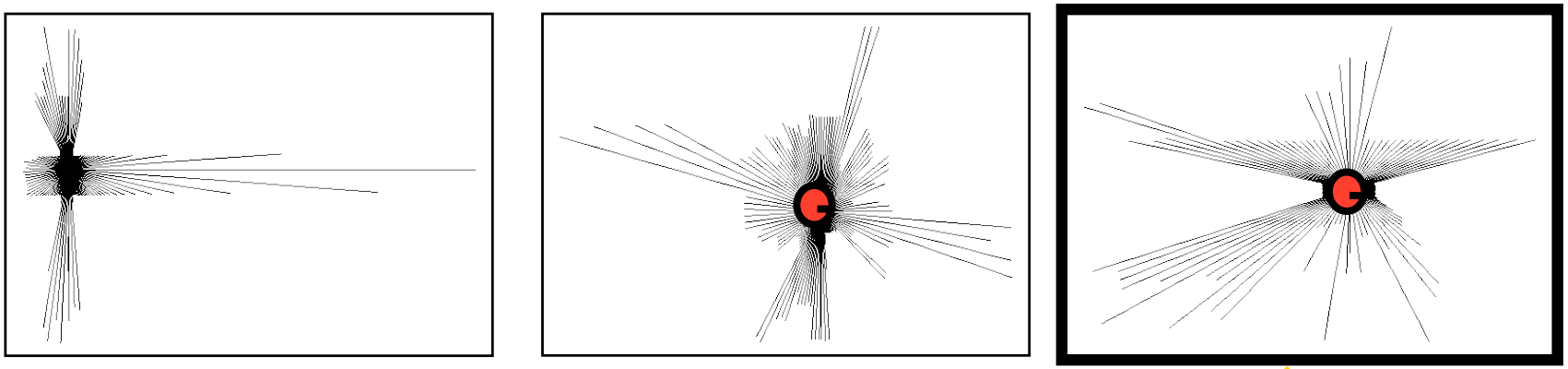
Observations



Corridor

Room

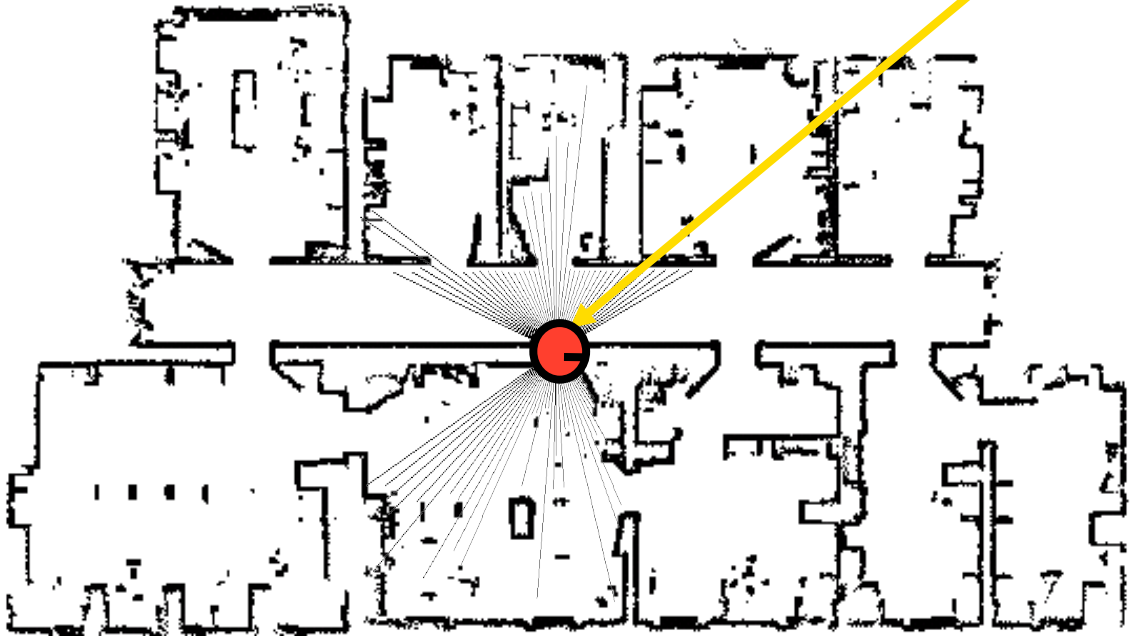
Observations



Corridor

Room

Doorway



Place Classification using Boosting

Observation:

There exists a variety of simple features f_i we can define on laser range scans.

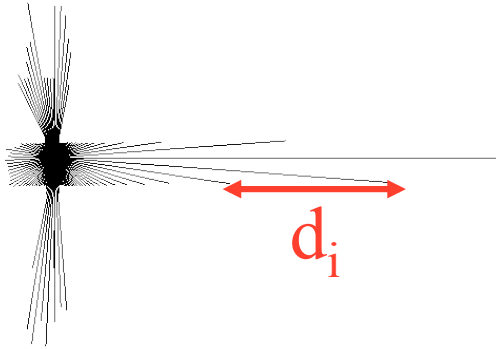
Problems:

Each single feature f_i gives poor classification rates.

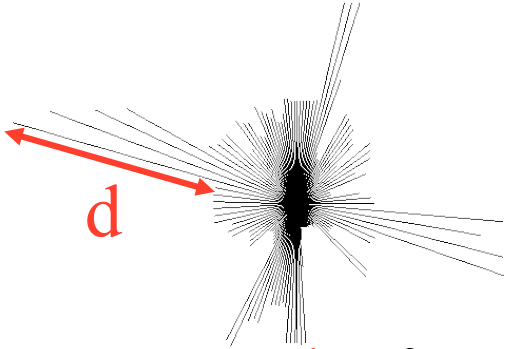
Idea:

Combine multiple simple features to form a strong classifier using AdaBoost.

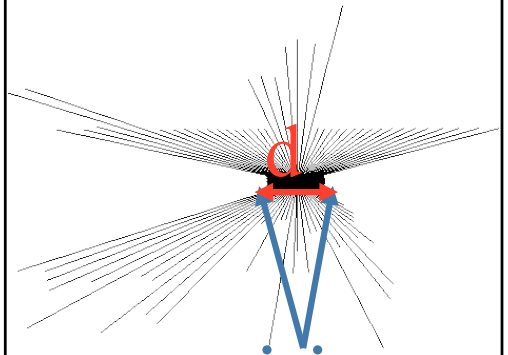
Simple Features



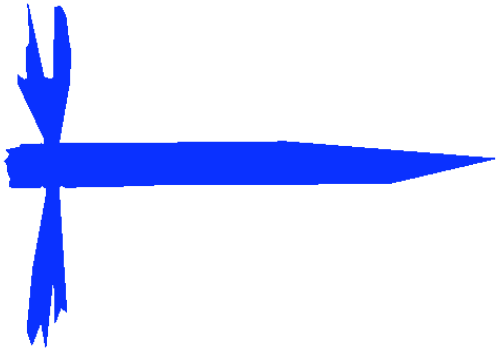
- $f = \frac{1}{N} \Sigma\, d_i$



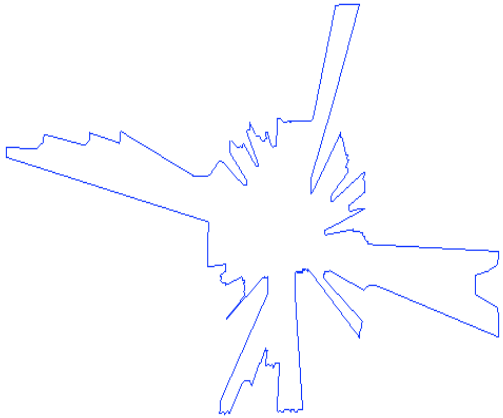
- gap = d > θ
- f = #gaps



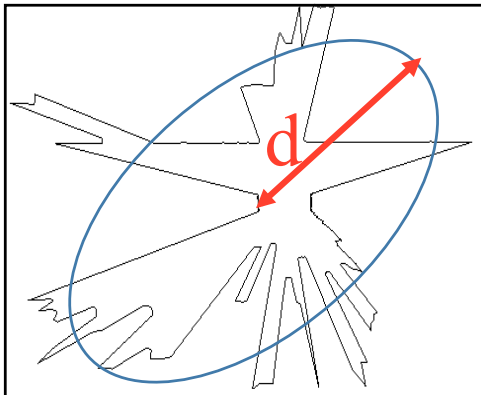
- f = d



- f = area

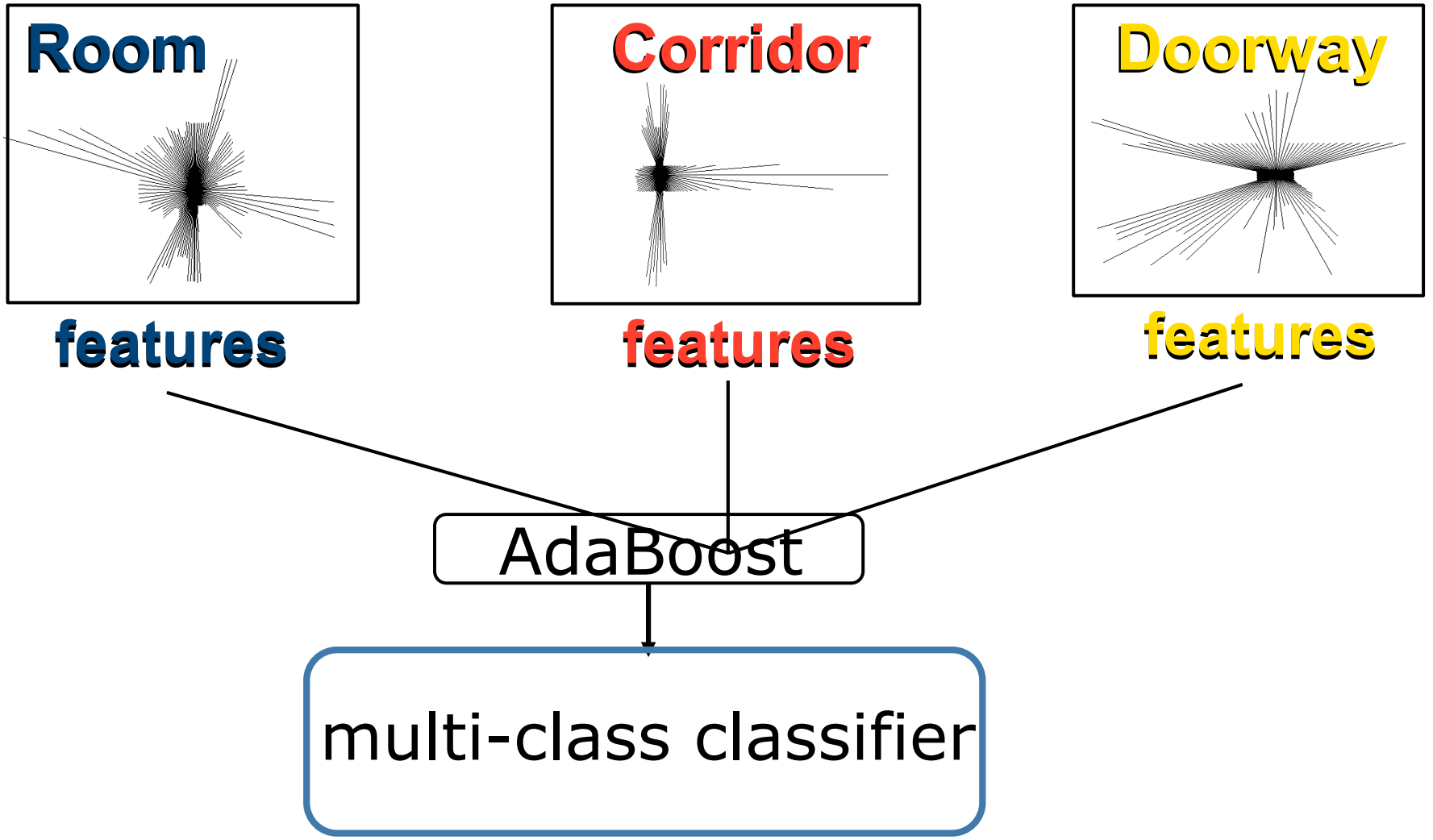


- f = perimeter

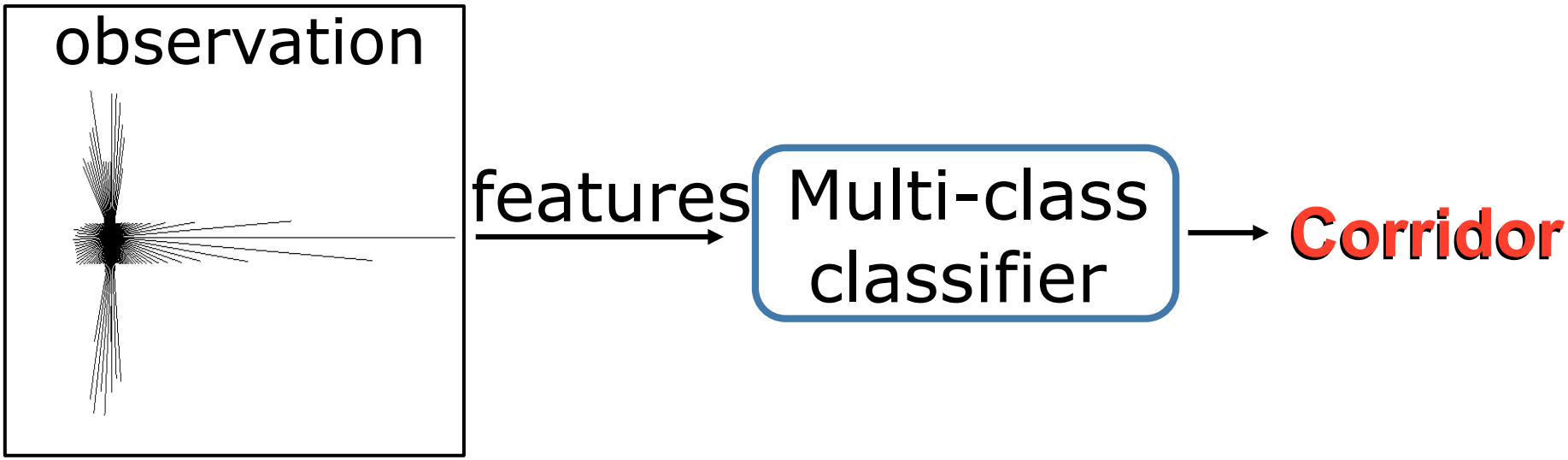


- f = d

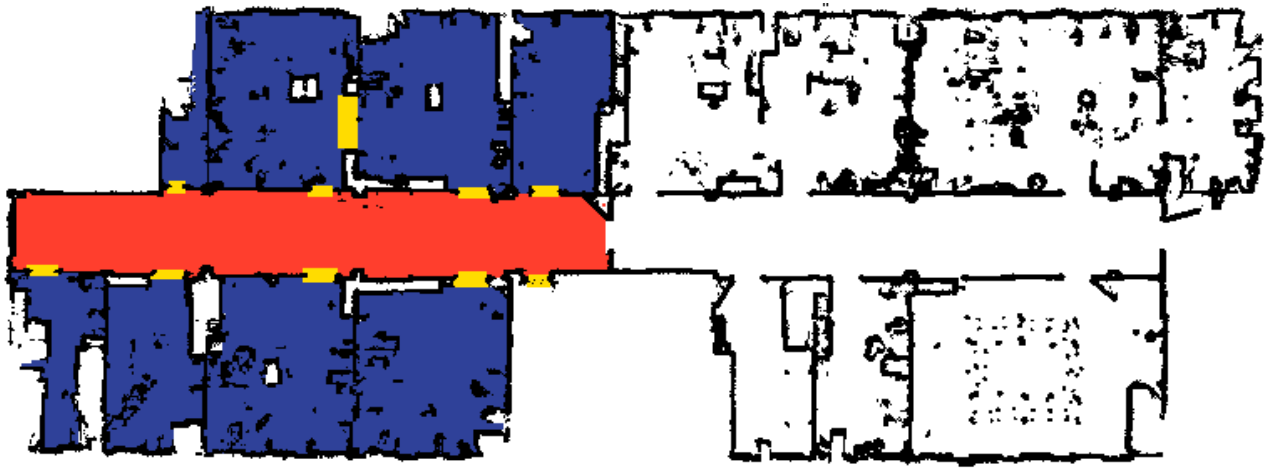
Learning


Room
Corridor
Doorway
features
features
features
AdaBoost
multi-class classifier

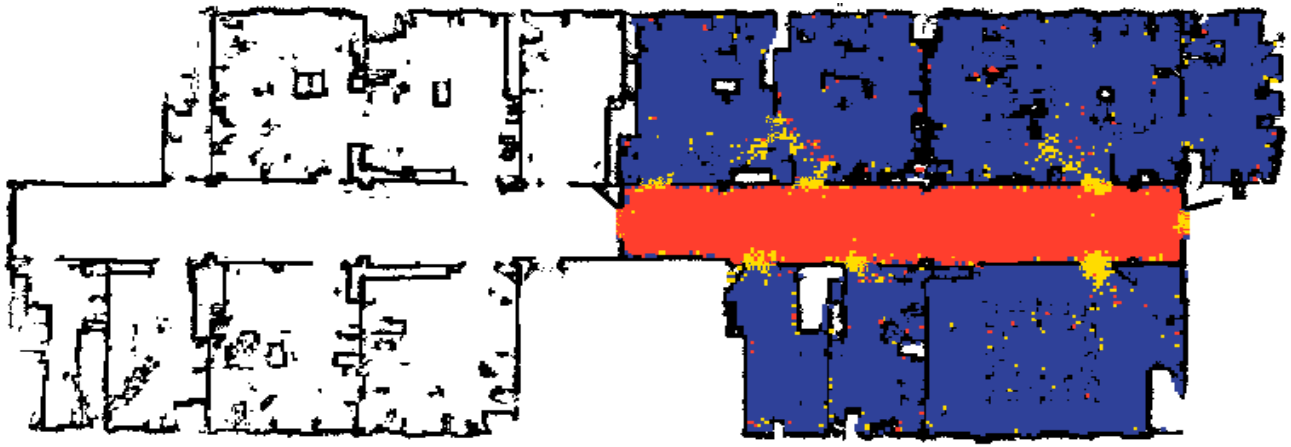
Classification



Application Result



Training
examples:
16045

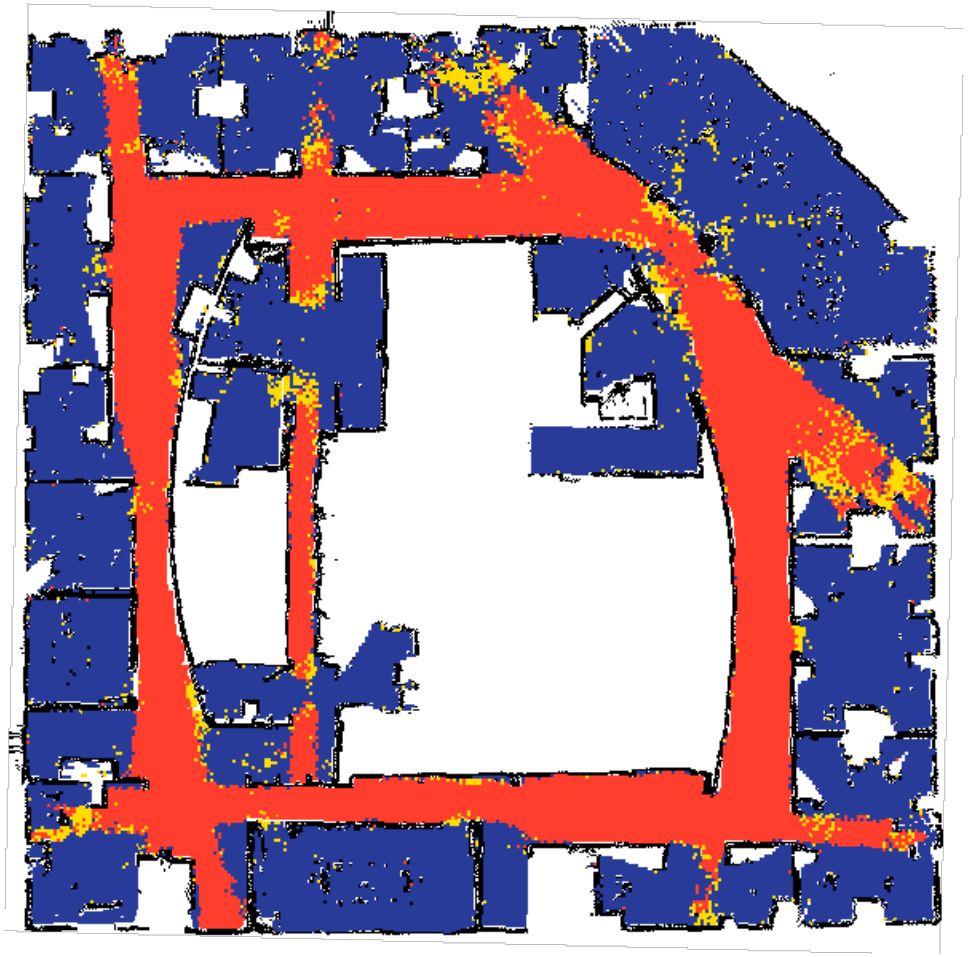


Test
examples:
18726
classification:
94%

Corridor **Room** **Doorway**

Application to a New Environment

Application to a New Environment



Corridor **Room** **Doorway**

Related Work

- Minimizing the uncertainty in the belief distribution during SLAM or navigation
  [Kollar&Roy ‘06, He et al. ’08, Roy et al. ‘99]

- Consider shaking movements of the head, acquire only images during stable phases
  [Ido et al. ’09]

- Image preprocessing to restore an image and reduce the effects of motion blur
  [Pretto et al. ’09]

Thank you!