Advanced Robotics
Mapping

Gerald Steinbauer
Institute for Software Technology
Agenda

• Motivation
• Basic Definitions
• SLAM with Occupancy Grids
• EKF-SLAM with landmarks
• PF-SLAM with landmark
• Grid-based PF-SLAM
• Exploration
• Mapping in ROS
“Probabilistic Robotics”
by Sebastian Thrun, Wolfram Burgard and Dieter Fox
MIT Press
Motivation

• problem of the **acquisition** of a representation of the environment
• generation of maps is a central piece of **navigation**
• maps are commonly used for **localization**
• basis for **decision** making
Mapping is a hot Topic
The General Problem of Mapping

formally, mapping involves, given the sensor data,

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

to calculate the most likely map

\[
m^* = \arg \max_m P(m \mid d)
\]
Why is SLAM hard?

- we mainly talk about Simultaneous Localization and Mapping (SLAM)
- the hypothesis space of all possible maps is huge (easily more than $10^5$ variables)
- it is a chicken-and-egg problem, making a map is easy if we have a pose, localization is easy if we have a map, we have to interleave localization and mapping
- noise in perception and acting - commands and observations are uncertain
- perceptional ambiguities, different places and landmarks may look similar
- cycles in the environment, places reappear
The SLAM Problem

A robot is exploring an unknown, static environment.

**Given:**
- the robot’s controls
- observations of nearby features

**Estimate:**
- map of features
- path of the robot
Maps with Landmarks

• map comprises (distinguishable) landmarks and their position

• landmarks can be very different
  • trees
  • special geometric shapes
  • RFID tags
  • visual tags
  • visual features …

• robot need appropriate sensors
Maps as Occupancy Grids

• represent the **environment** as grid map
  • grid cells are either free or occupied
  • may represent some uncertainty
  • can be in 2d, 2.5d or 3d
Mapping with Known Pose

- we make the problem less hard
- we assume an oracle that tells us the robot's pose
- occupancy grids addresses the generation of consistent maps from noisy observations
- occupancy grids are a rectangular arrangement of random variables
Occupancy Grid Maps

- introduced by Moravec and Elfes in 1985
- represent environment by a grid
- estimate the probability that a location $m_i$ is occupied by an obstacle

**key assumptions**
- occupancy of individual cells ($m[x,y]$) is independent

\[
Bel(m_t) = P(m_t \mid u_1, z_2, \ldots, u_{t-1}, z_t) = \prod_i Bel(m_{t}^{[i]})
\]

- robot positions are known!
Update Occupancy Grid

• treat single cells as **binary** random variable \( p(m_i=1) \)

• use log odd ratio

\[
l(x) := \log \left( \frac{p(x)}{p(-x)} \right) = \log \left( \frac{p(x)}{1 - p(x)} \right)
\]

**occupancy_grid_update**({\(l_{t-1,i}\), \(x_t, z_t\)})

for all cells \( m_i \) do

  if \( m_i \) in perceptual field of \( z_t \) then

    \[
    l_{t,i} = l_{t-1,i} + \text{inverse_sensor_model}(m_i, x_t, z_t) - l_0
    \]

  else

    \[
    l_{t,i} = l_{t-1,i}
    \]

endfor

return \( \{l_{t,i}\} \)
Inverse Sensor Model

- represents the probability of a cell being occupied given a position and sensor readings

\[ inverse\_sensor\_model(m_i, x_t, z_t) = \log \frac{p(m_i | z_t, x_t)}{1 - p(m_i | z_t, x_t)} \]

reflects also the uncertainty of the sensor reading
Incremental Updating of Occupancy Grids
Computing the Most Likely Map

(a)  (b)  (c)

(d)  (e)  (f)

conflict
Computing the Most Likely Map

• compute values for \( m \) that maximize

\[
m^* = \arg \max_{m} P(m \mid z_1, \ldots, z_t, x_1, \ldots, x_t)
\]

• assuming a uniform prior probability for \( p(m) \), this is equivalent to maximizing (application of Bayes rule)

\[
m^* = \arg \max_{m} P(z_1, \ldots, z_t \mid m, x_1, \ldots, x_t)
\]

\[
= \arg \max_{m} \prod_{t=1}^{T} P(z_t \mid m, x_t)
\]

\[
= \arg \max_{m} \sum_{t=1}^{T} \ln P(z_t \mid m, x_t)
\]
Simultaneous Localization and Mapping

• Full SLAM: estimates entire path and map!

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

• Online SLAM:

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

integrations typically done one at a time

estimates most recent pose and map!
Graphical Model of Online SLAM

\[
p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \ldots dx_{t-1}
\]
Graphical Model of Full SLAM

\[ p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \]
Landmark based EKF-SLAM

- analog problem to localization with extended Kalman-Filter
- now the joint belief $p_t(x_t, \Theta_{1:N})$ is a $3+3N$ dimensional normal distributed random variable
- assumes the map $m=(\Theta_1, \ldots, \Theta_N)$ is a set of $N$ landmarks, $\Theta_j \in \mathbb{R}^{2 \times S}$, $S$..signature of landmark
- conceptually very simple
- scaling is problematic, covariance matrix $O(N^3)$
- only for “simple” problems
- data association problem
Data Association Problem

- arises if landmarks can not uniquely identified
- origins from radar tracking
Representation EKF-SLAM

- map with $N$ landmarks: $(3+3N)$-dimensional Gaussian

$$Bel(x_i, m_i) = \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_{3N} \end{pmatrix} \begin{pmatrix} \sigma^2_x & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma^2_y & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma^2_{\theta} \\ \sigma_{l_1x} & \sigma_{l_1y} & \sigma_{l_1\theta} \\ \sigma_{l_2x} & \sigma_{l_2y} & \sigma_{l_2\theta} \\ \vdots & \vdots & \vdots \\ \sigma_{l_{3N}x} & \sigma_{l_{3N}y} & \sigma_{l_{3N}\theta} \end{pmatrix} \begin{pmatrix} \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_{3N}} \\ \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_{3N}} \\ \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_{3N}} \\ \sigma^2_{l_1} & \sigma^2_{l_2} & \cdots & \sigma^2_{l_{3N}} \\ \sigma^2_{l_2l_1} & \sigma^2_{l_2l_2} & \cdots & \sigma^2_{l_2l_{3N}} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$
Victoria Park Data Set

[courtesy by E. Nebot]
Basic Idea

1. update pose $x, y, \theta$ and $\Sigma$ according the actual control and the previous state $\mu_{t-1}$

2. for each landmark observation $r_t^i, \phi_t^i, s_t^i$ repeat
   1. if the landmark observation is new then initialize the related parts of the state
   2. determine the measurement Jacobean matrix according to the landmark observation
   3. update the state using the EKF correction step

3. return the new state $\mu_t, \Sigma_t$
Assumptions

• landmarks distinguishable
• (maximum) number of landmarks are known in advance
• known correspondences
Algorithm with Known Correspondences

```
1: Algorithm EKF_SLAM_known_correspondences(μ_{t-1}, Σ_{t-1}, w_t, z_t, c_t):
2:     \( F_x = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \end{pmatrix} \)
3:     \( \bar{μ}_t = μ_{t-1} + F_x^T \begin{pmatrix} -\frac{3N}{ω_t} \sin(μ_{t-1,θ} + ω_t Δt) \\ \frac{2N}{ω_t} \cos(μ_{t-1,θ} + ω_t Δt) \end{pmatrix} \)
4:     \( G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{3N}{ω_t} \cos(μ_{t-1,θ} + ω_t Δt) & \frac{2N}{ω_t} \cos(μ_{t-1,θ} + ω_t Δt) \\ 0 & 0 & -\frac{3N}{ω_t} \sin(μ_{t-1,θ} + ω_t Δt) & \frac{2N}{ω_t} \sin(μ_{t-1,θ} + ω_t Δt) \end{pmatrix} F_x \)
5:     \( \bar{Σ}_t = G_t \otimes_{t-1} G_t^T + F_x^T R_t F_x \)
6:     \( Q_t = \begin{pmatrix} \sigma_r & 0 \\ 0 & \sigma_θ \end{pmatrix} \)
7:     for all observed features \( z_t = (r_t^i, φ_t^i, s_t^i)^T \) do
8:         \( j = c_t^i \)
9:         if landmark \( j \) never seen before
10:            \( \begin{pmatrix} \bar{μ}_{j,x} \\ \bar{μ}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{μ}_{t,x} \\ \bar{μ}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos(φ_t^i + \bar{μ}_{t,θ}) \\ r_t^i \sin(φ_t^i + \bar{μ}_{t,θ}) \end{pmatrix} \)
11:         endif
12:     \( δ = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{μ}_{j,x} - \bar{μ}_{t,x} \\ \bar{μ}_{j,y} - \bar{μ}_{t,y} \end{pmatrix} \)
13:     \( q = δ^T δ \)
14:     \( \vec{z}_t^i = \begin{pmatrix} \text{atan2}(δ_y, δ_x) - \bar{μ}_{t,θ} \\ \bar{μ}_{j,θ} \end{pmatrix} \)
15:     \( F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \end{pmatrix} \)
16:     \( H_t = \frac{1}{q} \begin{pmatrix} \sqrt{q}δ_x & 0 & \cdots & 0 \\ 0 & \sqrt{q}δ_y & 0 & \cdots & 0 \end{pmatrix} \)
17:     \( K_t^i = \bar{Σ}_t H_t^i T (H_t^i \otimes_{t} H_t^iT + Q_t)^{-1} \)
18:     \( \bar{μ}_t = \bar{μ}_t + K_t^i (z_t^i - \vec{z}_t^i) \)
19:     \( \bar{Σ}_t = (I - K_t^i H_t^iT) \otimes_{t} \)
20:     endfor
21:     \( \mu_t = \bar{μ}_t \)
22:     \( \Sigma_t = \bar{Σ}_t \)
23:     return \( \mu_t, \Sigma_t \)
```
EKF SLAM with Known Correspondences
General EFK-SLAM

- so far we assumed a known **number** of landmarks and known **correspondences**
- the algorithm is similar to the case with known correspondences
- now we have a **variable** number of landmarks $N$
- for every observed landmark **generate** a new landmark for the state vector and its covariance
- calculate the **Mahalanobis** distance to all previous collected landmarks
- if the distance is above a **threshold** add it to the state
- update the state according the **observation**
1: Algorithm EKF_SLAM($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, N_{t-1}$):

2: \quad N_t = N_{t-1}

3: \quad F_x = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}

4: \quad \bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_x}{\omega_t} \sin(\mu_{t-1,\theta}) + \frac{v_y}{\omega_t} \sin((\mu_{t-1,\theta} + \omega_t \Delta t)) \\ \frac{v_y}{\omega_t} \cos(\mu_{t-1,\theta} - \frac{v_x}{\omega_t} \cos((\mu_{t-1,\theta} + \omega_t \Delta t)) \\ 0 \end{pmatrix} \omega_t \Delta t

5: \quad G_t = I + F_x^T \begin{pmatrix} 0 & 0 & \frac{v_x}{\omega_t} \cos(\mu_{t-1,\theta}) - \frac{v_y}{\omega_t} \cos((\mu_{t-1,\theta} + \omega_t \Delta t)) \\ 0 & 0 & \frac{v_y}{\omega_t} \sin(\mu_{t-1,\theta}) - \frac{v_x}{\omega_t} \sin((\mu_{t-1,\theta} + \omega_t \Delta t)) \\ 0 \end{pmatrix} F_x

6: \quad \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_x^T R_t F_x

7: \quad Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_{\phi} & 0 \\ 0 & 0 & \sigma_g \end{pmatrix}

8: \quad \text{for all observed features } z_t^i = (r_t^i, \phi_t^i, s_t^i)^T \text{ do}

9: \quad \begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ \bar{\mu}_{t,s} \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ -\sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}

10: \quad \text{for } k = 1 \text{ to } N_{t+1} \text{ do}

11: \quad \delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}

12: \quad q_k = r_t^i \delta_k

13: \quad \hat{z}_t^k = \begin{pmatrix} \text{atan2}(\sqrt{q_k}, \bar{\mu}_{k,\theta}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,\theta} \end{pmatrix}

14: \quad F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \end{pmatrix}

15: \quad H_t^k = \frac{1}{q_k} \begin{pmatrix} -\sqrt{q_k} \delta_{k,x} & -\sqrt{q_k} \delta_{k,y} & 0 & \sqrt{q_k} \delta_{k,x} & \sqrt{q_k} \delta_{k,y} & 0 \end{pmatrix} F_{x,k}

16: \quad \Psi_k = H_t^k \Sigma_t [H_t^k]^T + Q_t

17: \quad \pi_k = (z_t^i - \hat{z}_t^k)^T \Psi_k^{-1} (z_t^i - \hat{z}_t^k)

18: \quad \text{endfor}

19: \quad \pi_{N_t+1} = \alpha

20: \quad j(i) = \text{argmin } \pi_k

21: \quad N_t = \max \{ N_t, j(i) \}

22: \quad K_t^i = \Sigma_t [H_t^i(j(i))]^T \Psi_i^{-1}

23: \quad \hat{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i(j(i)))

24: \quad \Sigma_t = (I - K_t^i H_t^i(j(i))) \Sigma_t

25: \quad \text{endfor}

26: \quad \mu_t = \bar{\mu}_t

27: \quad \Sigma_t = \bar{\Sigma}_t

28: \quad \text{return } \mu_t, \Sigma_t
SLAM with Particle Filter

- to solve the full SLAM problem
- the idea is to estimate the joint probability $p(x_{1:t},m|z_{1:t},u_{1:t-1})$ about the robot path and the map
- use a special type of particle filter Rao-Blackwellized particle filter using particles for some variables and Gaussian for other variables
- each sample represents its own path and map (landmark) estimation
- the map depends on the poses of the robot
- we know how to build a map given the position of the sensor is known
Representation

<table>
<thead>
<tr>
<th>Particle ( k )</th>
<th>( x_{1:t}^{[k]} = {(x, y, \theta)^T}_{1:t}^{[k]} )</th>
<th>( \mu_1^{[k]}, \Sigma_1^{[k]} )</th>
<th>( \mu_2^{[k]}, \Sigma_2^{[k]} )</th>
<th>( \ldots )</th>
<th>( \mu_N^{[k]}, \Sigma_N^{[k]} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k = 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( k = 2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( k = M )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each sample represents a potential path and for each landmark a EKF.
Basic Algorithm

• do the following $M$ times:
  • retrieval: retrieve a pose $x^{[k]}_{t-1}$ from the particle set $Y_{t-1}$.
  • prediction: sample a new pose $x^{[k]}_{t} \sim p(x_{t}|x^{[k]}_{t-1},u_{t})$
  • measurement update: for each observed feature $z_{t}^{i}$ identify the correspondence $j$ for the measurement $z_{t}^{i}$, and incorporate the measurement $z_{t}^{i}$ into the corresponding EKF, by updating the mean $\mu_{j,t}^{[k]}$ and covariance $\Sigma_{j,t}^{[k]}$
  • importance weight: calculate the importance weight $w^{[k]}$ for the new particle

• resampling. sample, with replacement, $M$ particles, where each particle is sampled with a probability proportional to $w^{[k]}$
Factored Posterior (Landmarks)

\[
p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t})
\]
Knowledge of the robot’s true path renders landmark positions conditionally independent.
Factored Posterior

\[ p(x_{1:t}, l_{1:m} | z_{1:t}, u_{0:t-1}) = p(x_{1:t} | z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} | x_{1:t}, z_{1:t}) = p(x_{1:t} | z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i | x_{1:t}, z_{1:t}) \]

robot path posterior (localization problem)

conditionally independent landmark positions
update the EKFs

- do this for every particle separately
- repeat for each landmark observation separately
- if observation does not match landmark $j$ do not change mean $\mu_{j,t-1}^{[k]}$ and covariance $\Sigma_{j,t-1}^{[k]}$
- else use the previous EKF update step for known pose and correspondences
update the sample weight

- the weight of the samples is calculated according to importance sampling

\[
\begin{align*}
\omega_t^{[k]} &= \frac{\text{target distribution}}{\text{proposal distribution}} \\
&= \frac{p(x_t^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t})}{p(x_t^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\
&= \eta p(z_t \mid x_t^{[k]}, z_{1:t-1}, c_{1:t-1})
\end{align*}
\]

\[
\omega_t^{[k]} \approx \eta \left\| 2\pi Q_t^{[k]} \right\|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (z_t - \hat{z}_t^{[k]}) Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]}) \right\}
\]

\[
Q_t^{[k]} = G_t^{[k]T} \Sigma_{n,t-1}^{[k]} G_t^{[k]} + R_t
\]
FastSLAM – Action Update

Particle #1

Particle #2

Particle #3
FastSLAM – Sensor Update

Particle #1

Particle #2

Particle #3
FastSLAM – Sensor Update

Particle #1

Weight = 0.8

Particle #2

Weight = 0.4

Particle #3

Weight = 0.1
Results – Victoria Park

4 km traverse
< 5 m RMS position error
100 particles

Blue = GPS
Yellow = FastSLAM
Grid-Based RBPF

• same basis ideas
• assume known pose for mapping
• instead of landmarks use a grid-based representation
• commonly used technique for indoor environments
Rao-Blackwellization

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

\[ p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(m \mid x_{1:t}, z_{1:t}) \]
Rao-Blackwellization

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

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

this is localization, use MCL

use the pose estimate from the MCL part and apply mapping with known poses
Rao-Blackwellized Mapping

• each particle represents a possible trajectory of the robot

• each particle maintains its own map and updates it upon “mapping with known poses”

• each particle survives with a probability proportional to the likelihood of the observations relative to its own map
Particle Filter Example

map of particle 1

3 particles

map of particle 2

map of particle 3
Basic Algorithm

1: Algorithm FastSLAM\_occupancy\_grids(\(X_{t-1}, u_t, z_t\)):

2: \[\bar{X}_t = X_t = \emptyset\]

3: for \(k = 1\) to \(M\) do

4: \(x_t^{[k]} = \text{sample\_motion\_model}(u_t, x_t^{[k]}_{t-1})\)

5: \(w_t^{[k]} = \text{measurement\_model\_map}(z_t, x_t^{[k]}, m_t^{[k]}_{t-1})\)

6: \(m_t^{[k]} = \text{updated\_occupancy\_grid}(z_t, x_t^{[k]}, m_t^{[k]}_{t-1})\)

7: \(\bar{X}_t = \bar{X}_t + (x_t^{[k]}, m_t^{[k]}, w_t^{[k]})\)

8: endfor

9: for \(k = 1\) to \(M\) do

10: draw \(i\) with probability \(\propto w_t^{[i]}\)

11: add \((x_t^{[i]}, m_t^{[i]})\) to \(X_t\)

12: endfor

13: return \(X_t\)
Problem

• each map is quite **big** in case of grid maps
• since each particle maintains its **own** map
• therefore, one needs to keep the number of particles **small**

• solution: compute **better** proposal distributions!

• idea: **improve** the pose estimate before applying the particle filter
Pose Correction Using Scan Matching

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

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

- current measurement
- robot motion
- map constructed so far
Further Improvements

• improved proposals will lead to more accurate maps
• they can be achieved by adapting the proposal distribution according to the most recent observations
• flexible re-sampling steps can further improve the accuracy
Improved Proposal

the proposal adapts to the structure of the environment
Outdoor Campus Map

- 30 particles
- 250x250m²
- 1.088 miles (odometry)
- 20cm resolution during scan matching
- 30cm resolution in final map
Exploration

- using a mapping or SLAM algorithm a robot is able to **automatically** generate a map of an environment
- the robot has to **decide** where to go next or which part
- the robot needs a exploration **strategy**
Frontier-Based Exploration

- very simple and **popular** approach by Yamauchi
- basis is **occupancy** grid with three classes of cells
  - open: \( p(m_{xy}) < \) prior probability
  - unknown: \( p(m_{xy}) = \) prior probability
  - occupied: \( p(m_{xy}) > \) prior probability
- find frontier cells that are at the **border** of open and unknown areas, an open cell is adjacent to an unknown cell
- **group** adjacent to frontier regions (blob detection – region growing)
- **select** large enough regions as frontier
- **navigate** to a frontier
Example
Mapping in ROS

• ROS supports **laser-based** indoor SLAM
• uses **grid-based** Rao-Blackwellized particle filter
• based on the **open-source** implementation gmapping (openslam.org)
  • improved proposal distribution
  • adaptive resampling
  • loop closure
• **off-the-shelf** generation of 2d maps
• **needs** odometry and laser-scans

• for usage and parameter look to the **tutorial**
Examples
Thank you!