CS283: Robotics Spring 2024: EKF & Particle Filter & Planning

Sören Schwertfeger / 师泽仁

ShanghaiTech University

EXTENDED KALMAN FILTER (EKF)

Following Material:

- Cyrill Stachniss, University of Bonn

Kalman Filter

Prediction (Time Update)

(1) Project the state ahead

$$\hat{x}^-_k = Ay_{k-1} + Bu_k$$

(2) Project the error covariance ahead

$$P^-_k = AP_{k-1}A^T + Q$$

Correction (Measurement Update)

(1) Compute the Kalman Gain

$$K = P^-_k H^T (HP^-_k H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}^-_k + K(z_k - H\hat{x}^-_k)$$

(3) Update Error Covariance

$$P_k = (I - KH)P^-_k$$

EKF: Non-linear Dynamic Systems

- Most realistic problems (in robotics) involve nonlinear functions

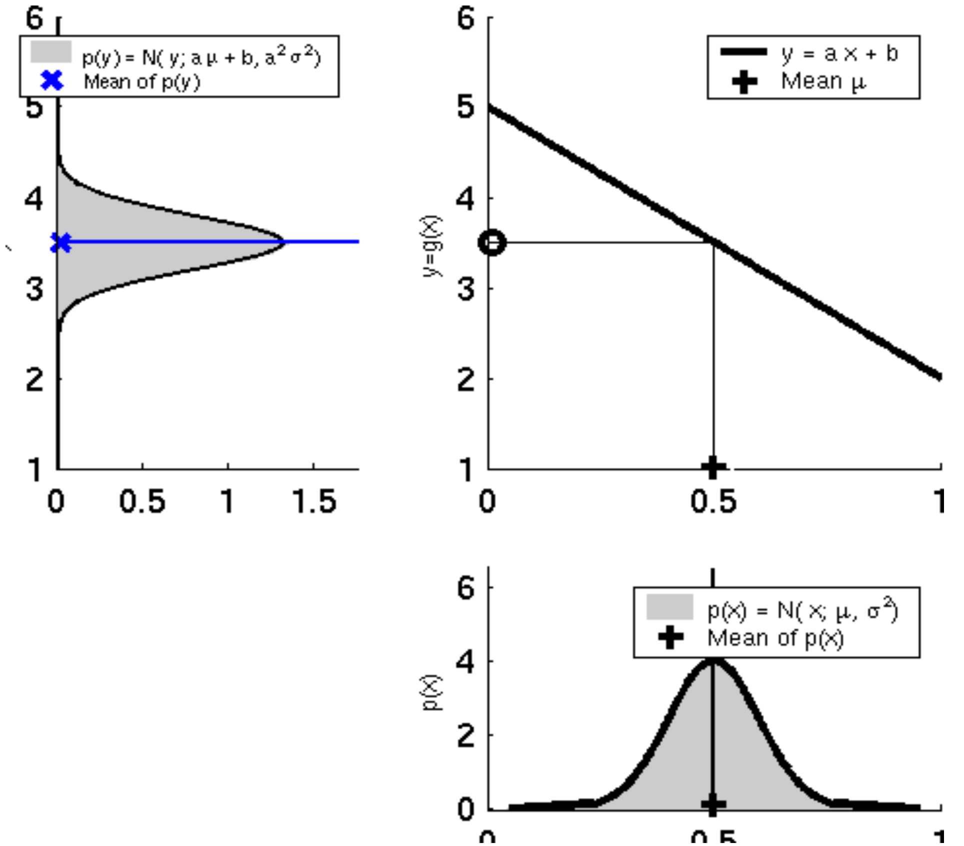
$$\cancel{x_t = A_t x_{t-1} + B_t u_t + \epsilon_t} \qquad \cancel{z_t = C_t x_t + \delta_t}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \qquad z_t = h(x_t) + \delta_t$$

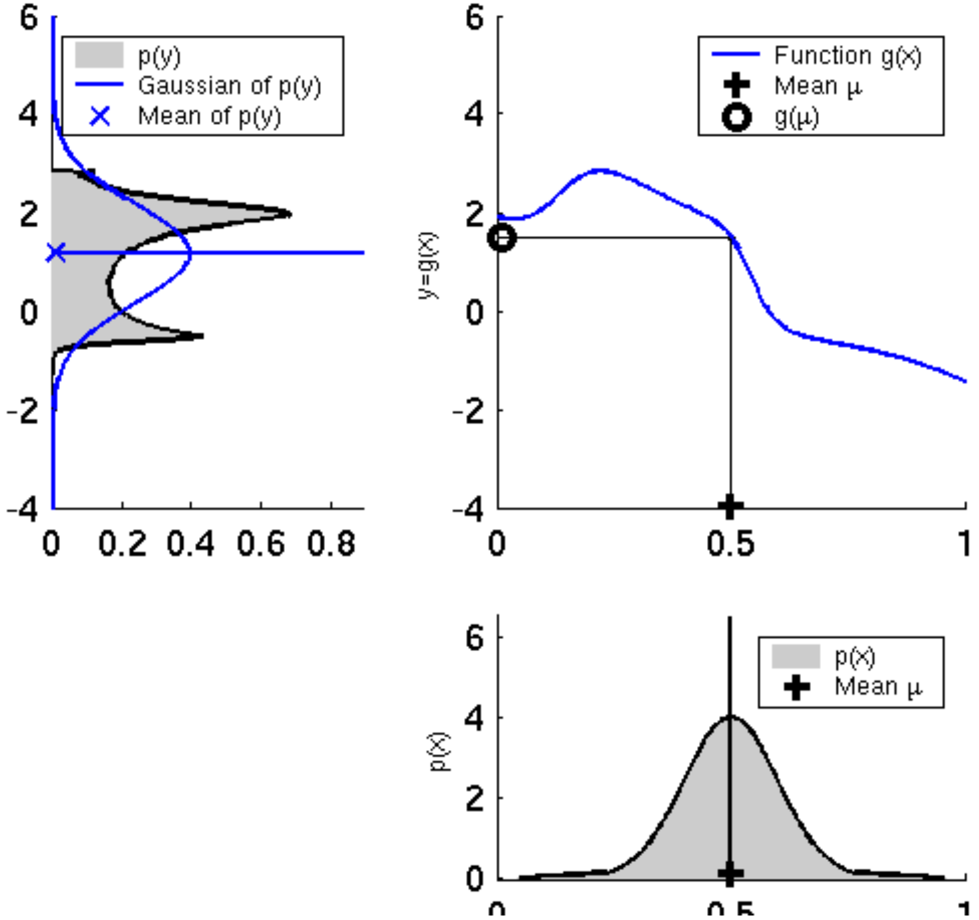
- Extended Kalman filter relaxes linearity assumption

Linearity Assumption Revisited



Courtesy: Thrun, Burgard, Fox

Non-linear system



Other Error Prop. Techniques

- **Second-Order Error Propagation**

Rarely used (complex expressions)

- **Monte-Carlo**

Non-parametric representation of uncertainties

1. Sampling from $p(X)$
2. Propagation of samples
3. Histogramming
4. Normalization

- **Extended Kalman Filter: EKF**

EKF Linearization: First Order Taylor Expansion

- Prediction:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=:\, G_t} (x_{t-1} - \mu_{t-1})$$

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=:\, H_t} (x_t - \bar{\mu}_t)$$

Jacobian matrices

Jacobian Matrix

- It' s a **non-square matrix** $n \times m$ in general
- Suppose you have a vector-valued function $f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix}$
- Let the **gradient operator** be the vector of (first-order) partial derivatives

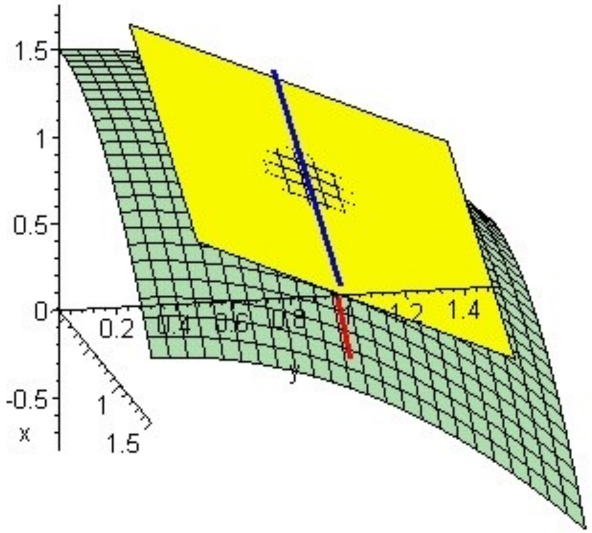
$$\nabla_{\mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix}^T$$

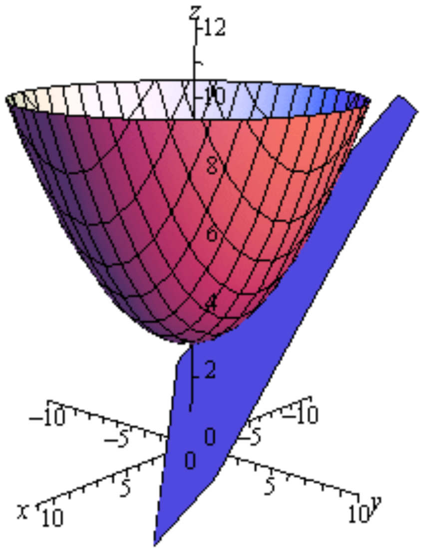
- Then, the **Jacobian matrix** is defined as

$$\mathbf{F}_{\mathbf{x}} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{bmatrix}$$

Jacobian Matrix

- It's the orientation of the **tangent plane** to the vector-valued function at a given point





- **Generalizes the gradient** of a scalar valued function
- Heavily used for **first-order error propagation…**

EKF Linearization: First Order Taylor Expansion

- Prediction:

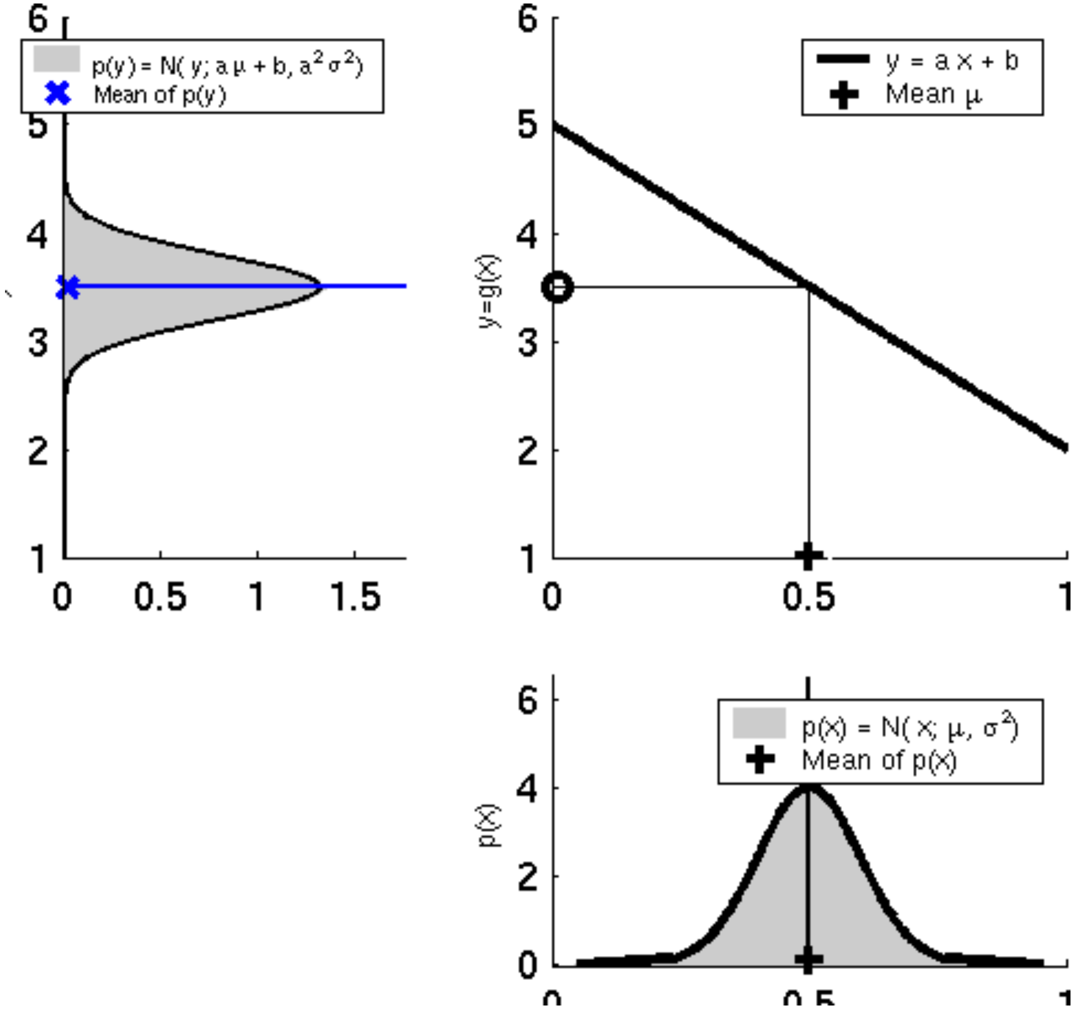
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=:\, G_t} (x_{t-1} - \mu_{t-1})$$

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=:\, H_t} (x_t - \bar{\mu}_t)$$

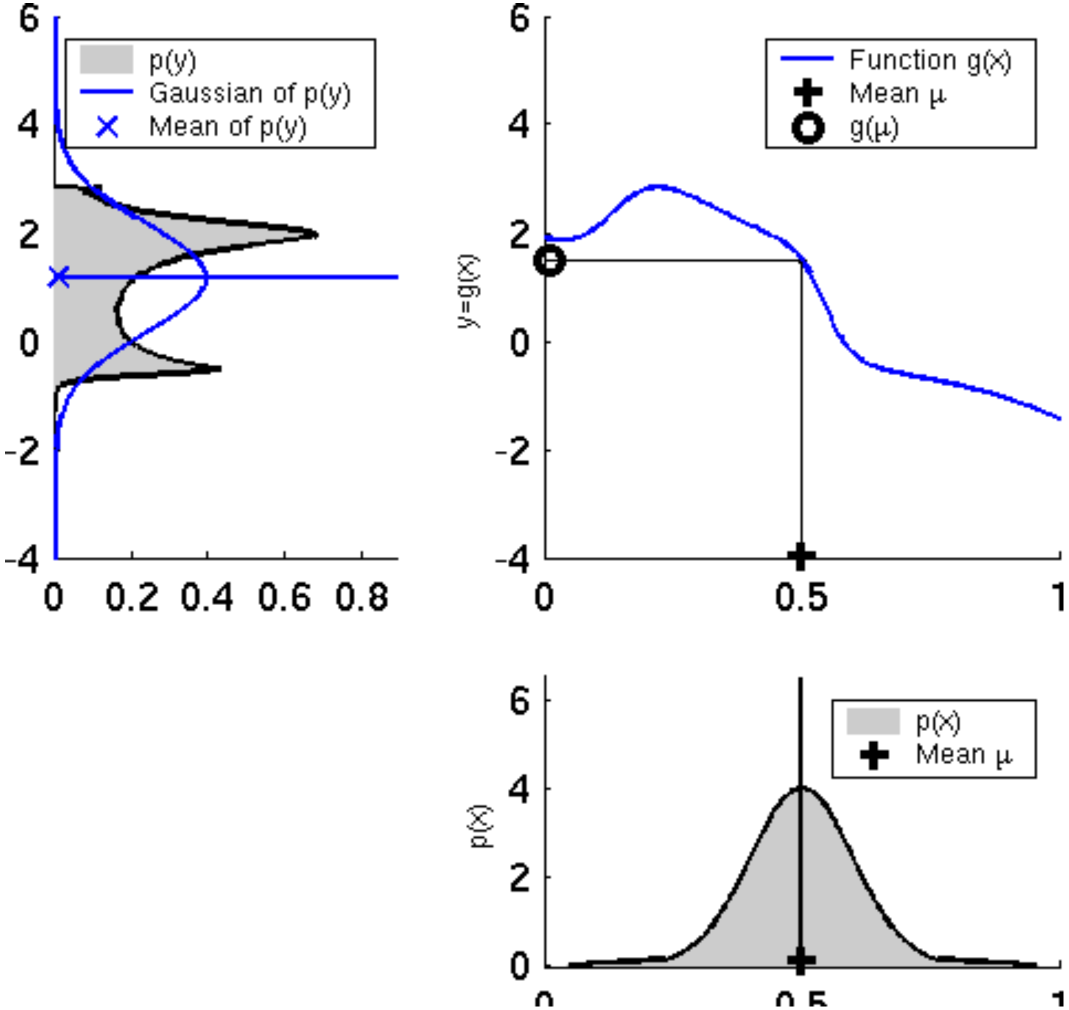
Jacobian matrices
=> linear functions!

Linearity Assumption Revisited



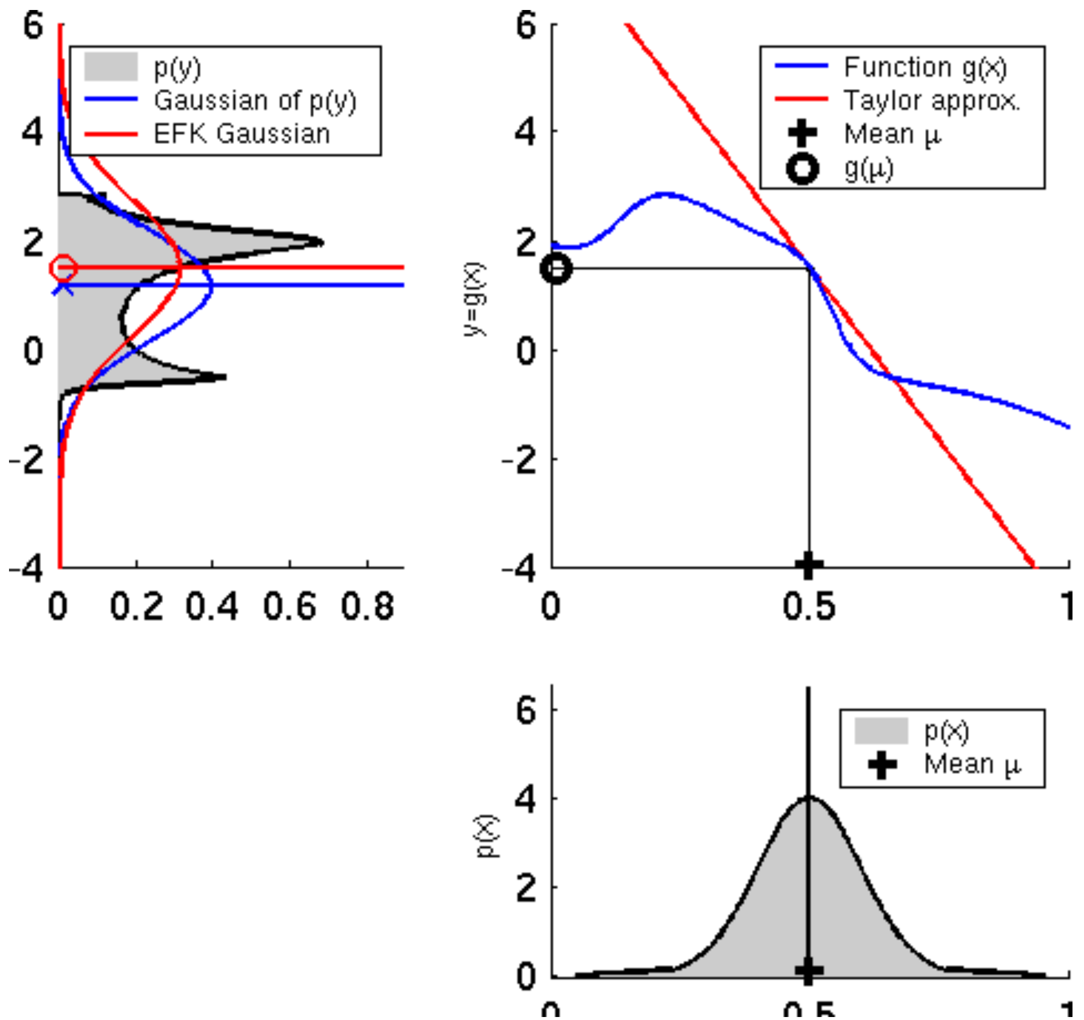
Courtesy: Thrun, Burgard, Fox

Non-Linear Function



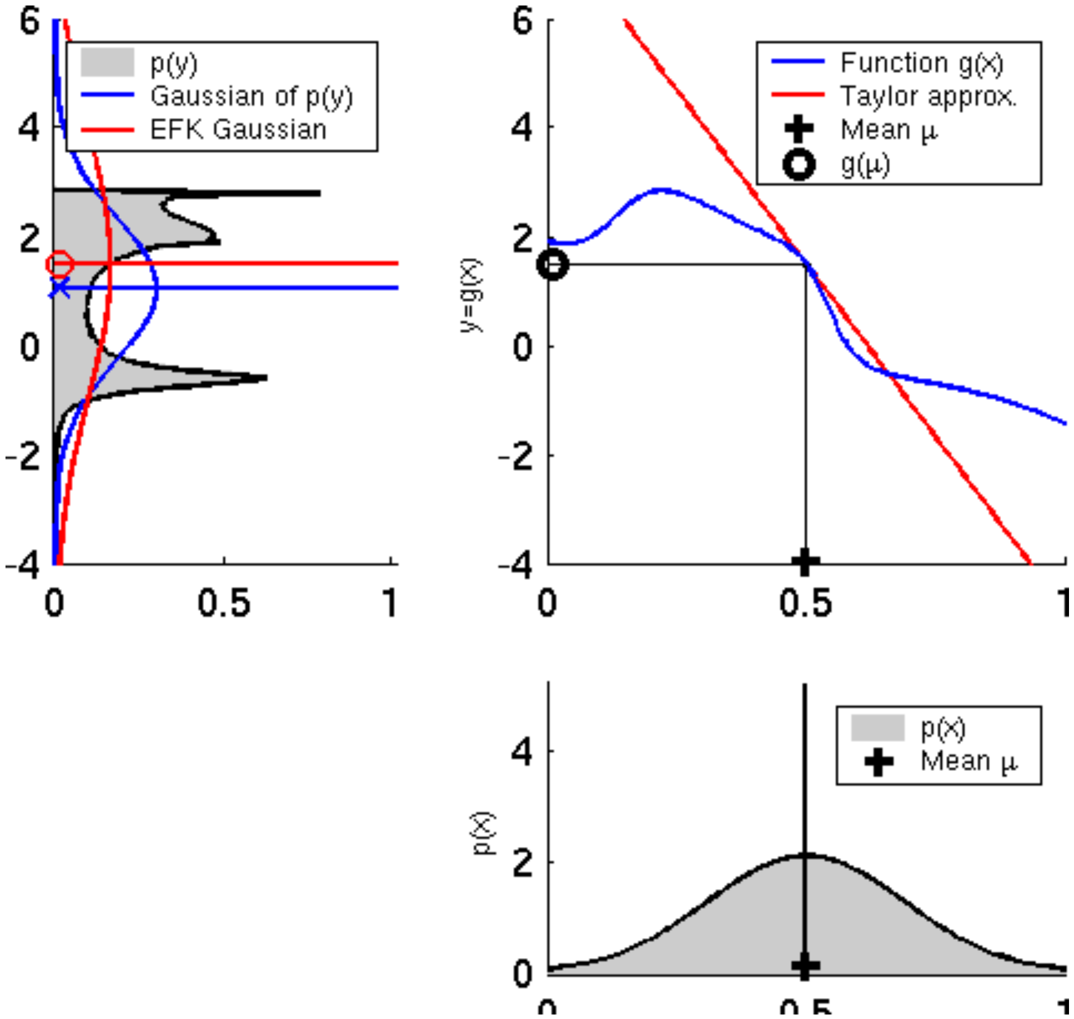
Courtesy: Thrun, Burgard, Fox

EKF Linearization (1)



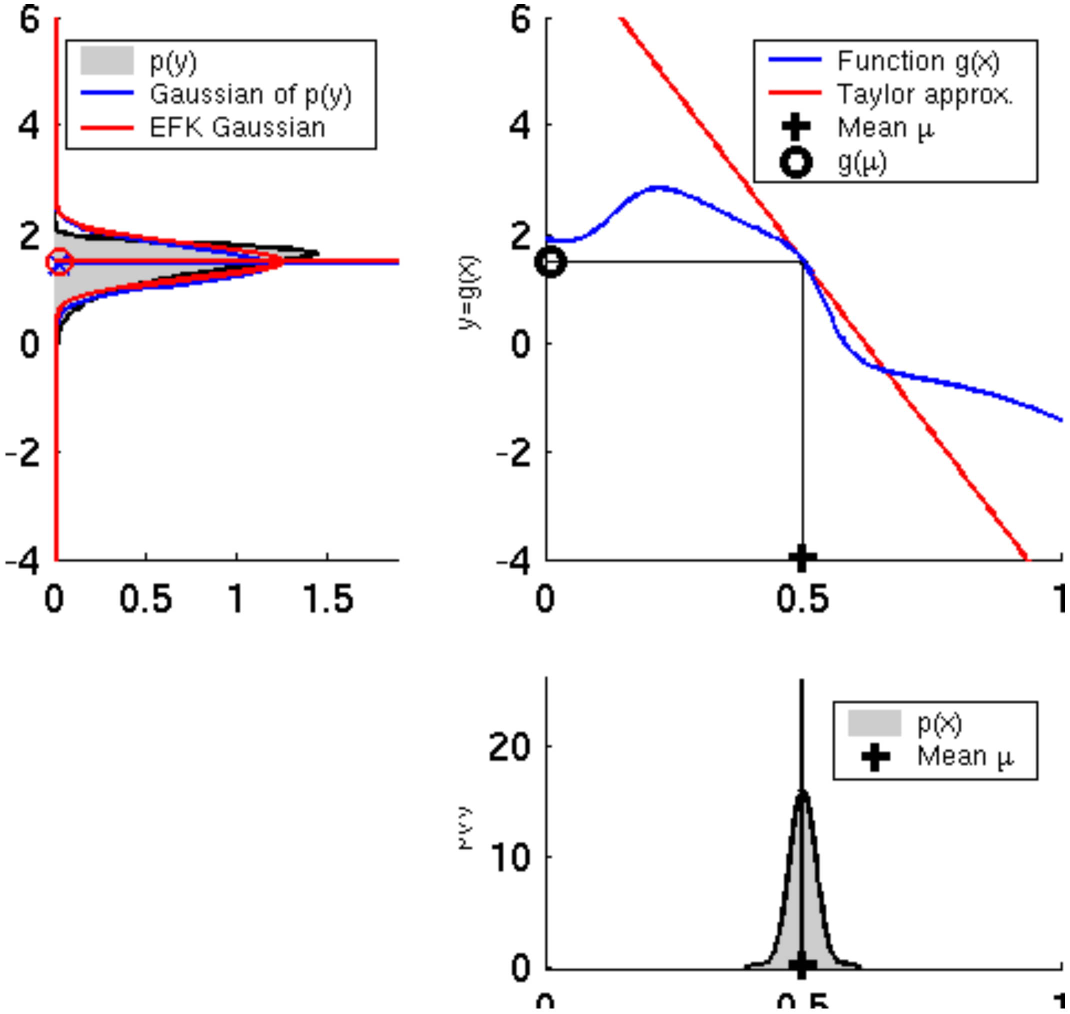
Courtesy: Thrun, Burgard, Fox

EKF Linearization (2)



Courtesy: Thrun, Burgard, Fox

EKF Linearization (3)



Courtesy: Thrun, Burgard, Fox

Extended Kalman Filter Algorithm

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

2: $\bar{\mu}_t = \underline{g(u_t, \mu_{t-1})}$

3: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$

4: $K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$

5: $\mu_t = \bar{\mu}_t + K_t(z_t - \underline{h(\bar{\mu}_t)})$

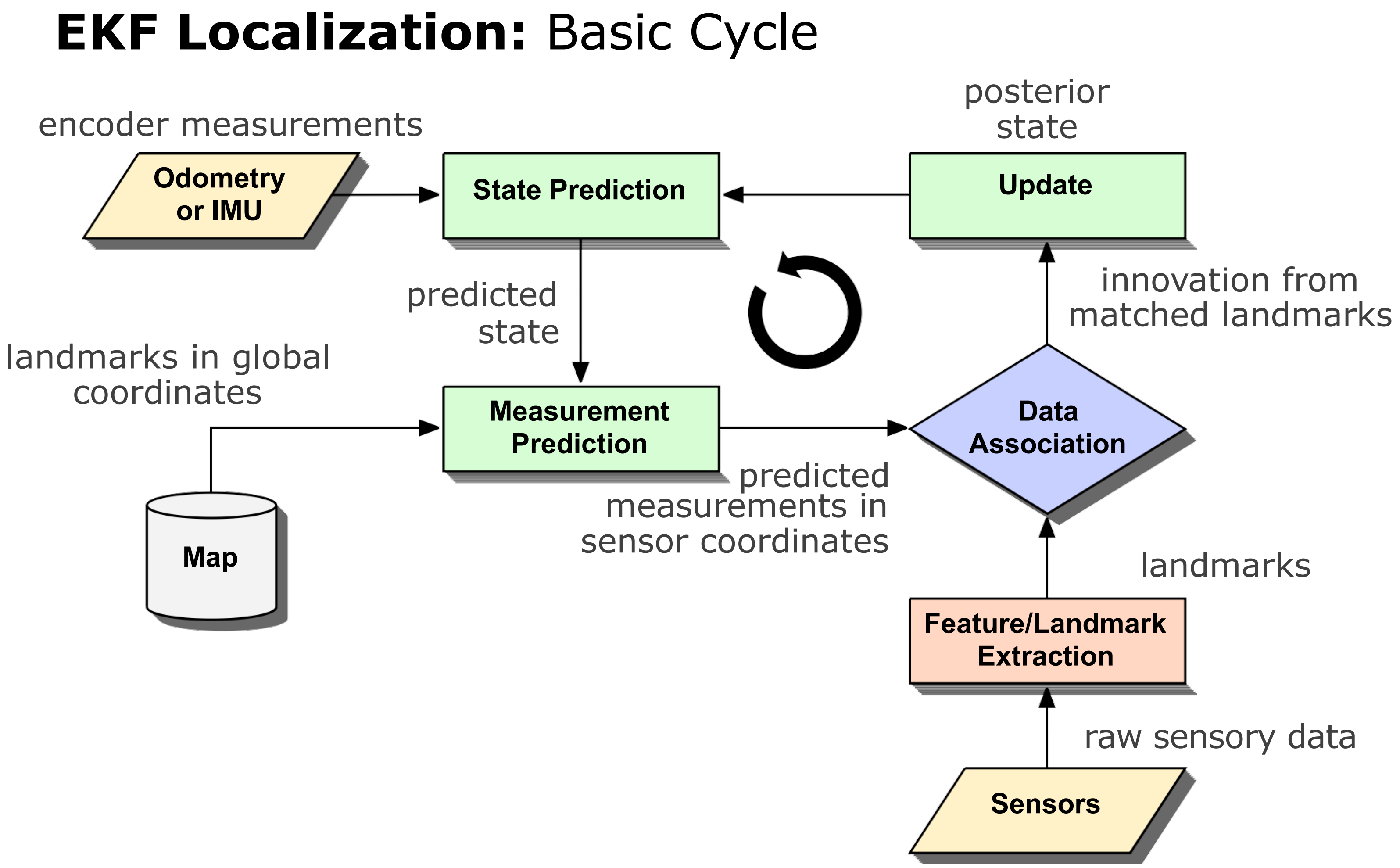
6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$

7: $return \ \mu_t, \Sigma_t$

$$\boxed{\begin{array}{c} A_t \leftrightarrow G_t \\ C_t \leftrightarrow H_t \end{array}}$$

KF vs. EKF

Landmark-based Localization



Landmark-based Localization

State Prediction (Odometry)

$$\hat{\mathbf{x}}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k)$$
$$\hat{C}_k = F_x\, C_k\, F_x^T + F_u\, U_k\, F_u^T$$

Control $\mathbf{u}_k$: wheel displacements s_l, s_r

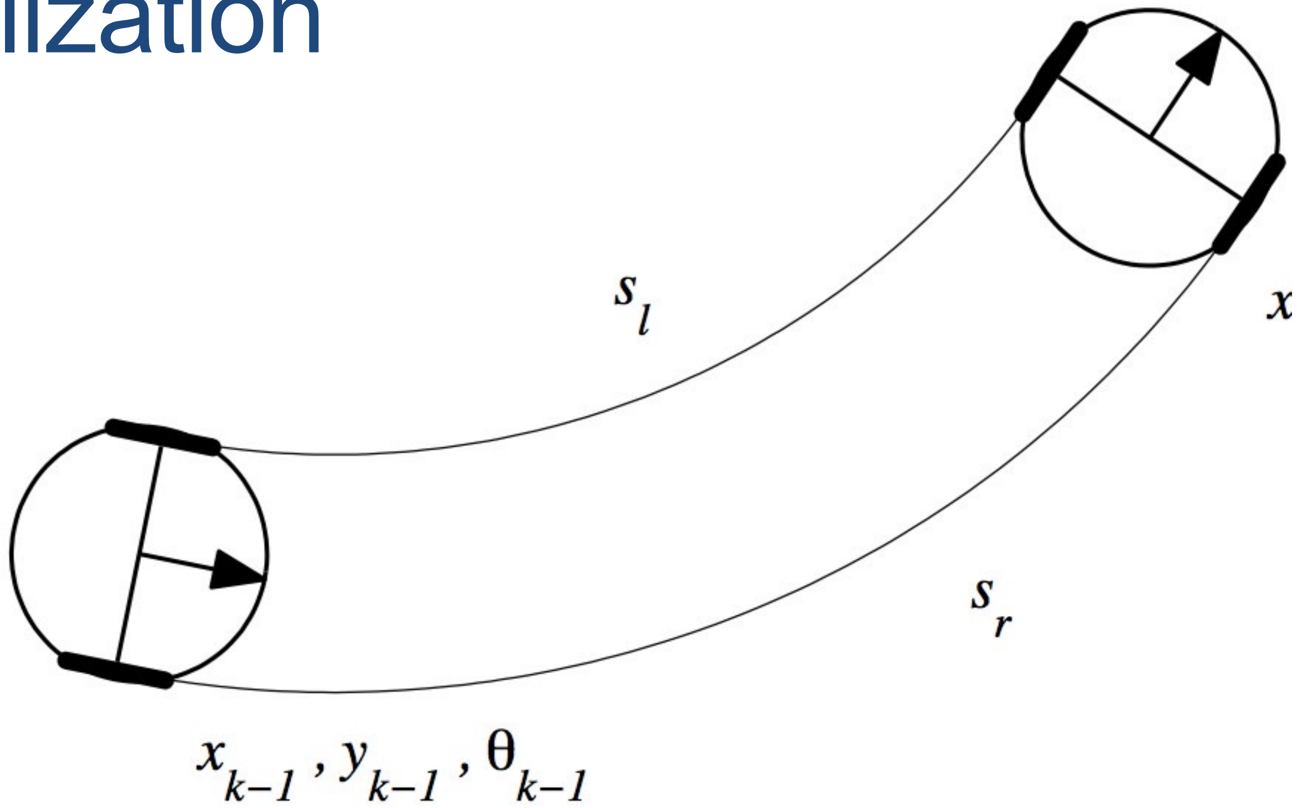
$$\mathbf{u}_k = (s_l \;\; s_r)^T \qquad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

Error model: linear growth

$$\begin{aligned} \sigma_l &= k_l\, |s_l| \\ \sigma_r &= k_r\, |s_r| \end{aligned}$$

Nonlinear process model f:

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(-\sin\,\theta_{k-1} + \sin(\theta_{k-1} + \frac{s_r-s_l}{b})\right) \\ \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(\cos\,\theta_{k-1} - \cos(\theta_{k-1} + \frac{s_r-s_l}{b})\right) \\ \frac{s_r-s_l}{b} \end{bmatrix}$$



Landmark-based Localization

State Prediction (Odometry)

$$\hat{\mathbf{x}}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k)$$
$$\hat{C}_k = F_x\, C_k\, F_x^T + F_u\, U_k\, F_u^T$$

Control $\mathbf{u}_k$: wheel displacements s_l, s_r

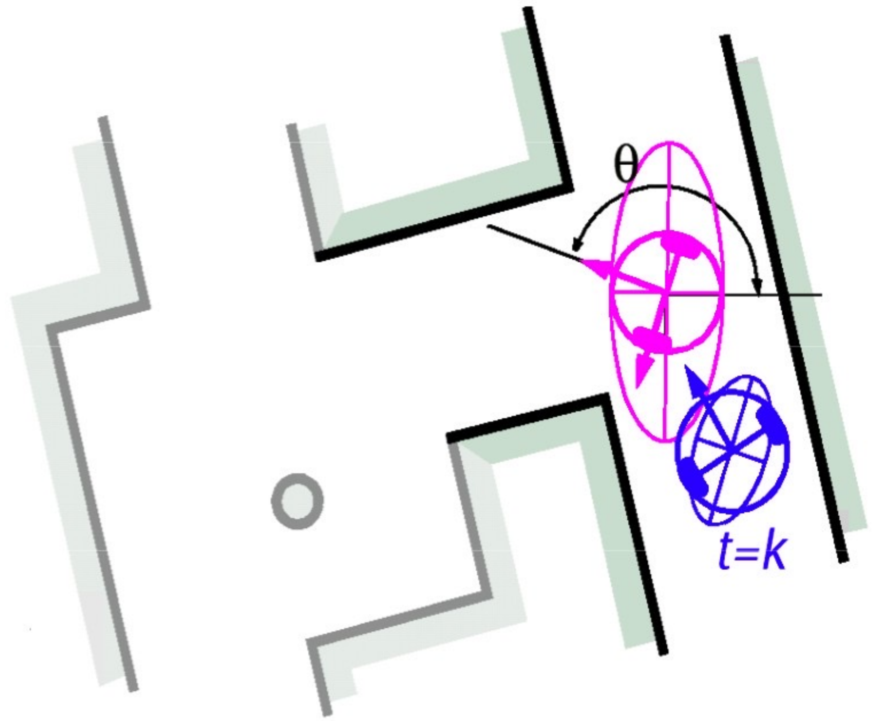
$$\mathbf{u}_k = (s_l \;\; s_r)^T \qquad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

Error model: linear growth

$$\begin{aligned} \sigma_l &= k_l\, |s_l| \\ \sigma_r &= k_r\, |s_r| \end{aligned}$$

Nonlinear process model f:

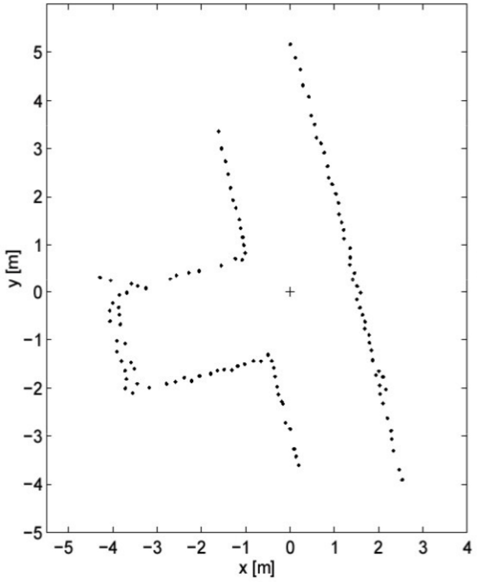
$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(-\sin\theta_{k-1} + \sin(\theta_{k-1} + \frac{s_r-s_l}{b})\right) \\ \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(\cos\theta_{k-1} - \cos(\theta_{k-1} + \frac{s_r-s_l}{b})\right) \\ \frac{s_r-s_l}{b} \end{bmatrix}$$



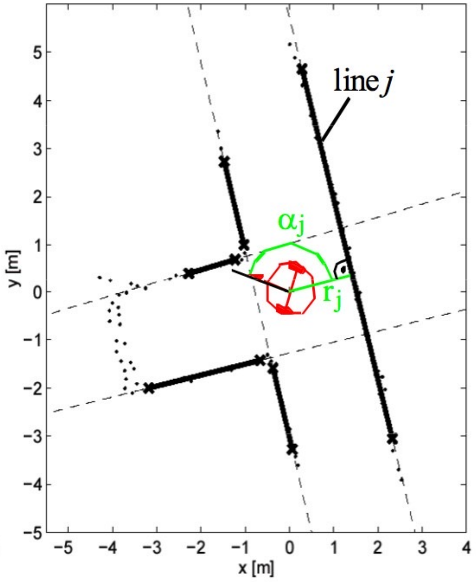
Landmark-based Localization

Landmark Extraction (Observation)

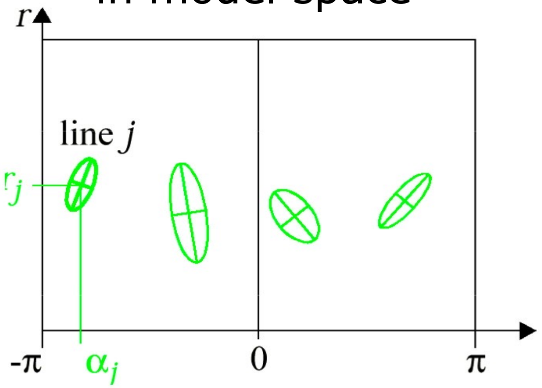
Raw laser range data



Extracted lines



Extracted lines in model space



$$\mathbf{z}_k = \begin{bmatrix} \alpha \\ r \end{bmatrix}$$

line model

$$x\ cos(\alpha) + y\ sin(\alpha) - r = 0$$

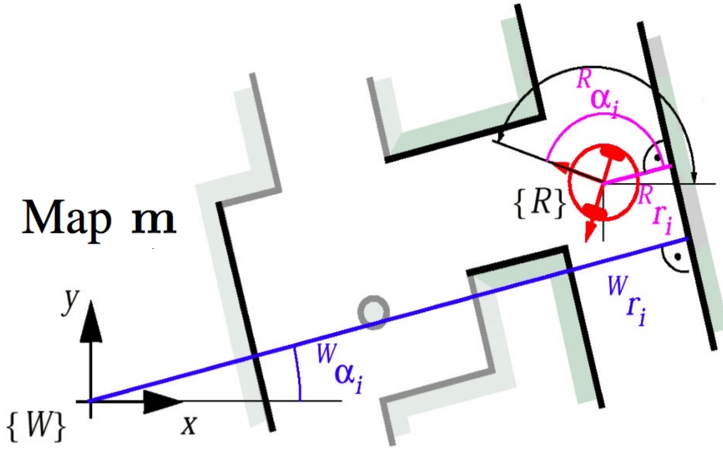
$$R_k = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_r^2 \end{bmatrix}$$

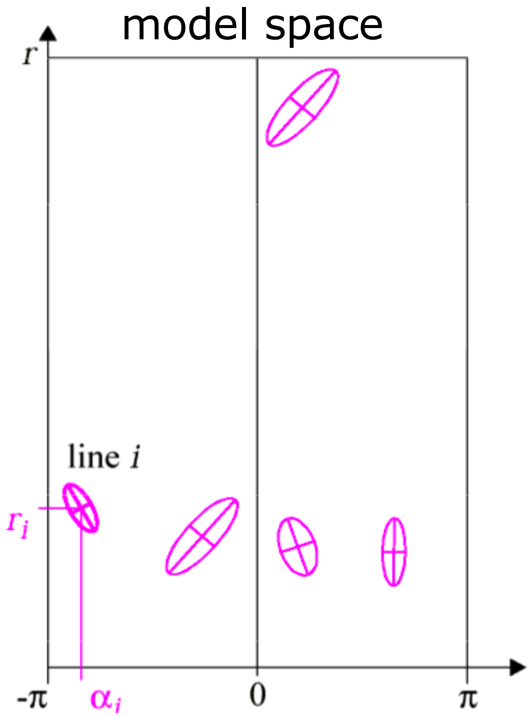
Landmark-based Localization

Measurement Prediction

- ...is a coordinate frame transform world-to-sensor
- Given the predicted state (robot pose), predicts the location $\hat{\mathbf{z}}_k$ and location uncertainty $H\,\hat{C}_k\,H^T$ of expected observations in sensor coordinates

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





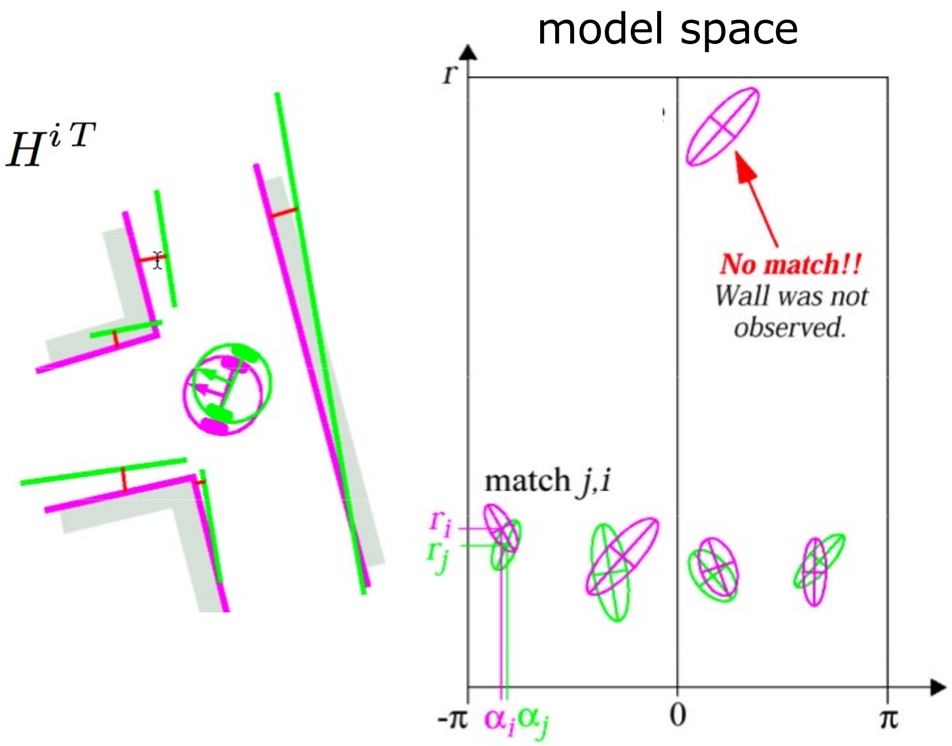
Landmark-based Localization

Data Association (Matching)

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

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

- Innovation and innovation covariance ν_k^{ij}



Green: observation

Magenta: measurement prediction

Landmark-based Localization

Update

- Kalman gain

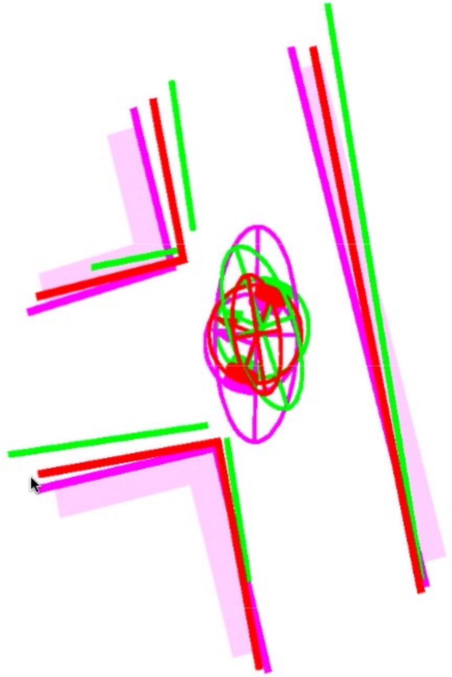
$$K_k = \hat{C}_k \, H^T \, S_k^{-1}$$

- State update (robot pose)

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

- State covariance update

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



Red: posterior estimate

ADMIN

Project

- 2^{nd} meeting of project – done for most groups
    - Sophia group & Robot dog group & mission planning LLM: meet after the lecture?

Midterm; April 23; during class hours

- Midterm: “test-run” for the final…
- Content:
  - Everything till (including) April 9 lecture.
    - Take a look at facts, algorithms, concepts
  - Take a look at the homeworks again
  - Sample exam: https://robotics.shanghaitech.edu.cn/sites/default/files//files/final_Example.pdf

- You are allowed to bring **3** A4 sheets (so 6 pages) of info to the exams (including Final – so for midterm maybe use 1.5 or 2 A4 sheets). You can write/ print anything on those sheets. On top of **every page** (so 6 times) there needs to be your **name (pinyin), student ID and ShanghaiTech email** address. We will check every cheat sheet before the exam and **confiscate** every sheet without name or with a name that is not yours.

- No electronics/ calculator/ smartwatch allowed

Presentation

- Upload your pdf/ ppt to your HW git latest tomorrow 22:00!
- If you want to use your own pdf/ ppt later you'll loose 30% of that score.
- Be present at all 3 presentation slots – listen carefully – everybody needs to ask in total at least 3 questions (for 3 different presentations)

r24hw_xuteng
r24hw_wangyzh2023
r24hw_zhoutt2023
r24hw_taoheng2023
r24hw_renwq2023
r24hw_luoxr2023
r24hw_liuxzh2023
r24hw_majx2023
r24hw_zhangjj2023
r24hw_guojing2023
r24hw_wushx2023
r24hw_zhangyq2023
r24hw_duanxin2023
r24hw_maxu2023
r24hw_zhangjt12023
r24hw_zhangsa2023
r24hw_linyx2023
r24hw_yushb2023
r24hw_xiefj
r24hw_lipc
r24hw_guszh2023
r24hw_hanzht2022
r24hw_shiyd2023

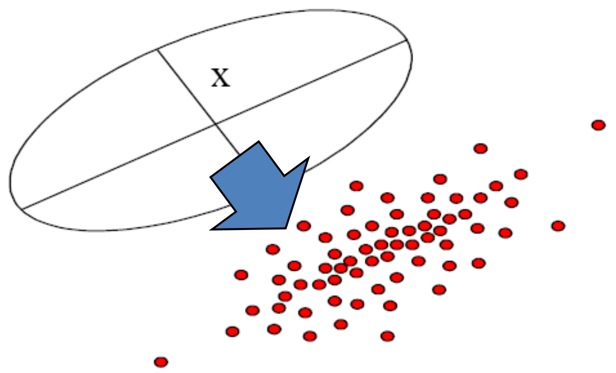
PARTICLE FILTER

Following Material:

- Wolfram Burgard, University of Freiburg

Particle Filter SLAM: FastSLAM

- **FastSLAM approach**
  - Using particle filters.
  - Particle filters: mathematical models that represent probability distribution as a set of discrete particles that occupy the state space.



probability distribution (ellipse) as particle set (red dots)

- Particle filter update
  - Generate new particle distribution using motion model and controls

a) For each particle:

1. Compare particle’s prediction of measurements with actual measurements
2. Particles whose predictions match the measurements are given a high weight

b) Filter resample:

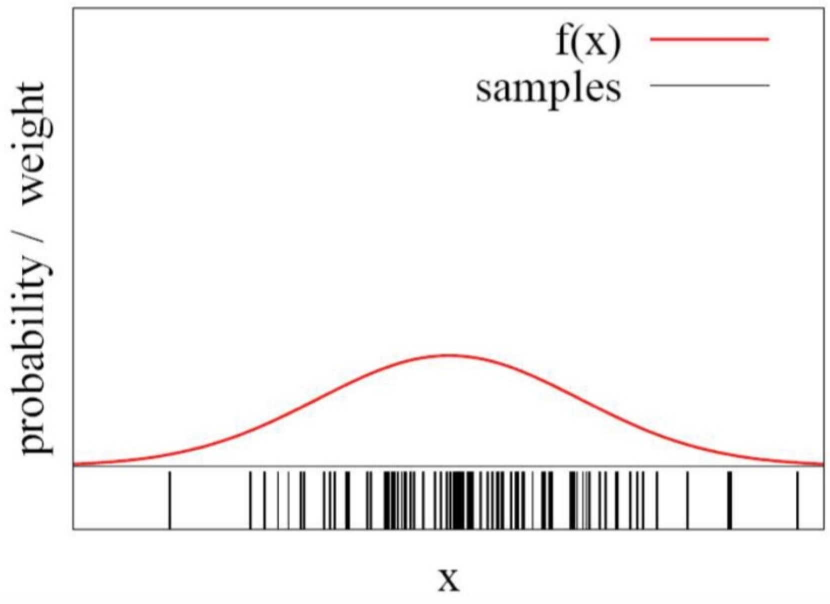
- Resample particles based on weight
- Filter resample
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.

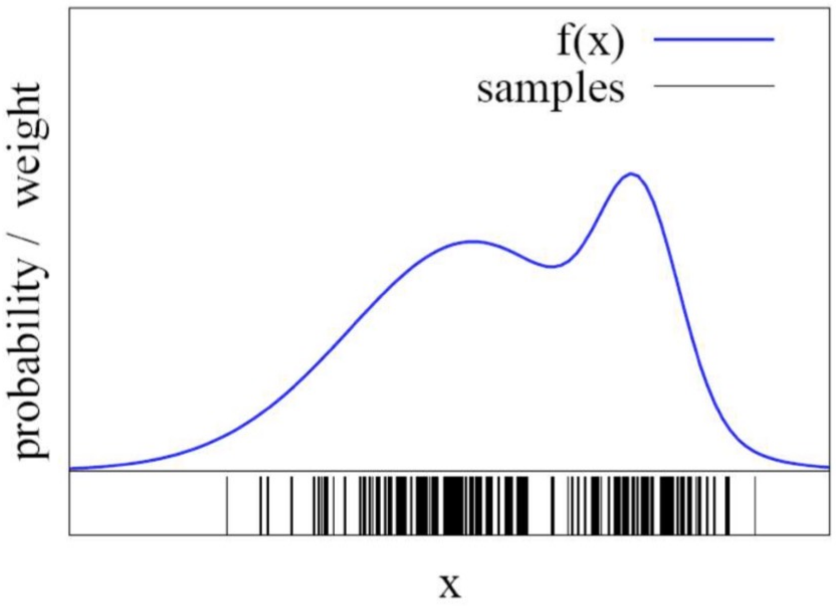
Motivation

- Particle filters are a way to efficiently represent non-Gaussian distribution
- Basic principle
  - Set of state hypotheses (“particles”)
  - Survival-of-the-fittest

Function Approximation

- Particle sets can be used to approximate functions

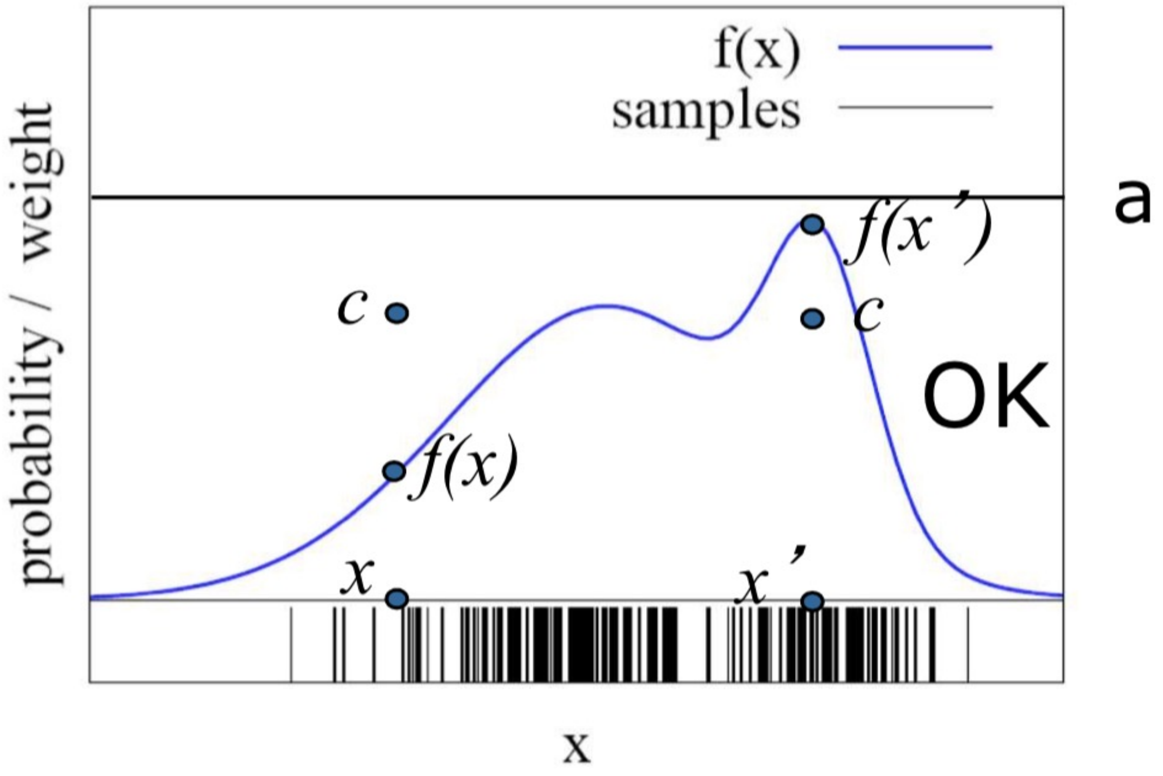




- The more particles fall into an interval, the higher the probability of that interval
- How to draw samples from a function/distribution?

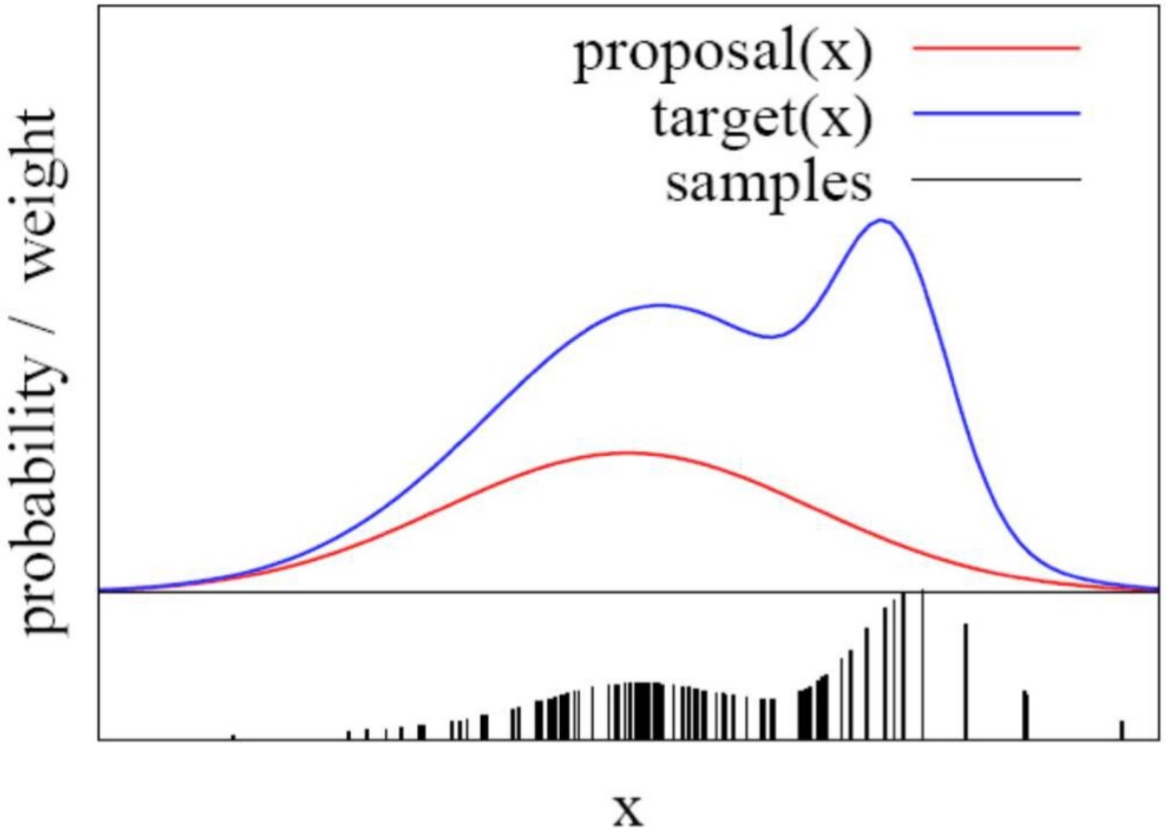
Rejection Sampling

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

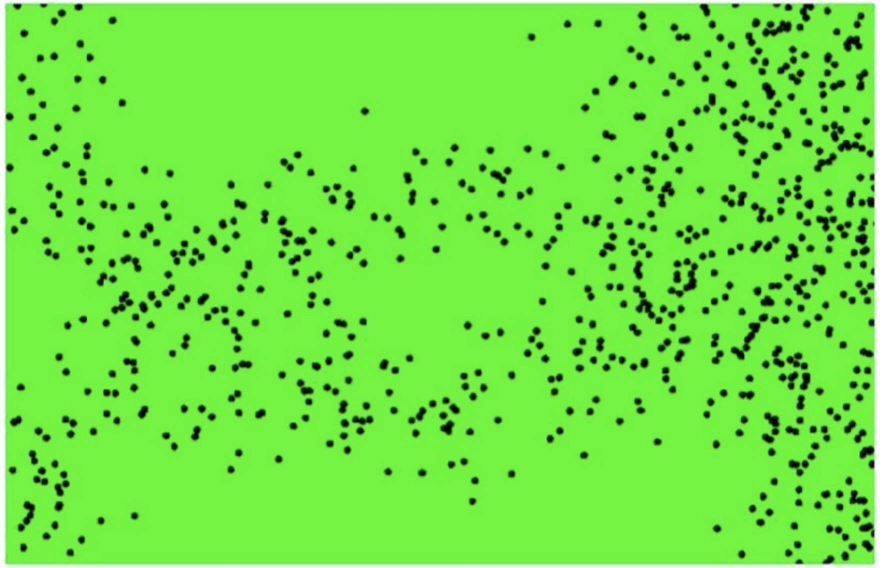


Importance Sampling Principle

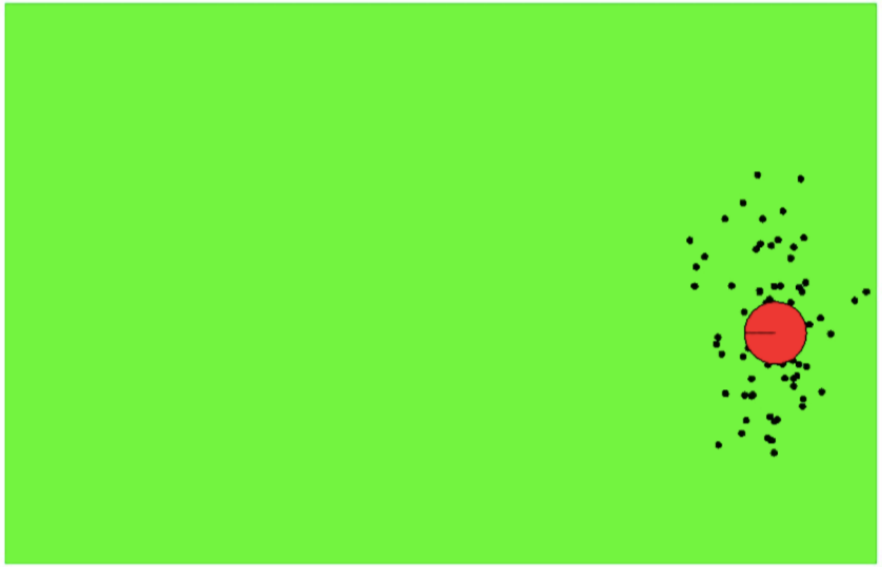
- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the "differences between g and f "
- $w = f/g$
- f is called target
- g is called proposal
- Pre-condition:
  - $f(x) > 0 \rightarrow g(x) > 0$



Importance Sampling with Resampling



Weighted Samples



After Resampling

Particle Filter Algorithm

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

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

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

Particle Filter Algorithm

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

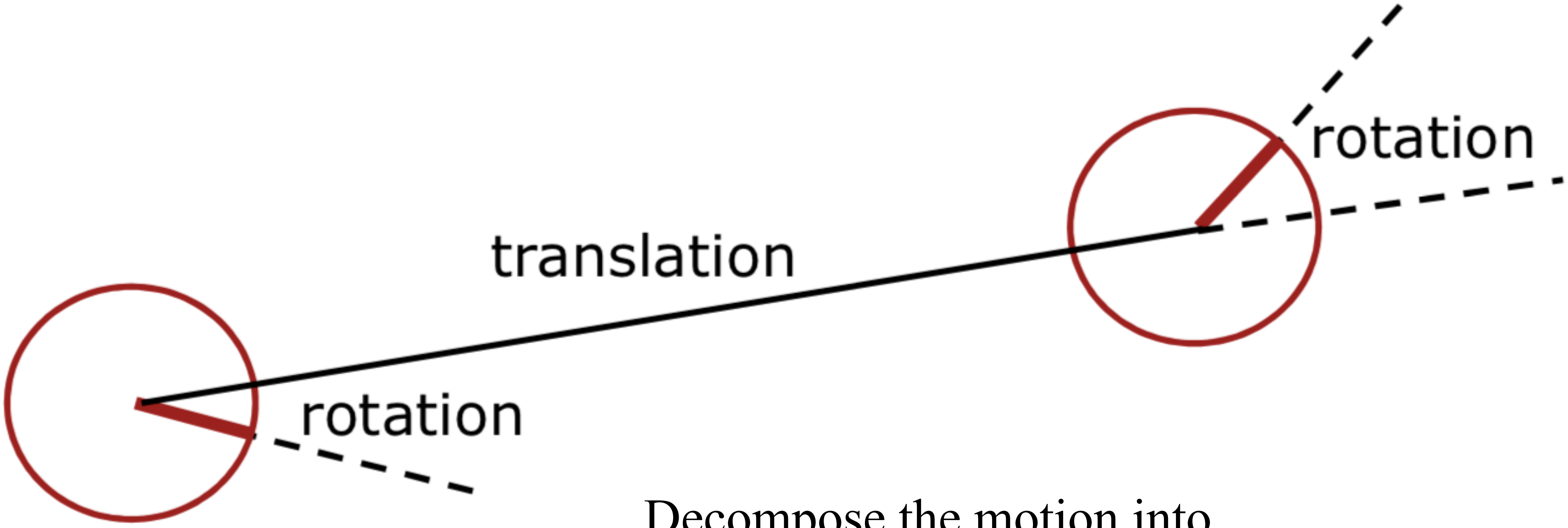
Mobile Robot Localization

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

Motion Model Reminder

According to the estimated motion

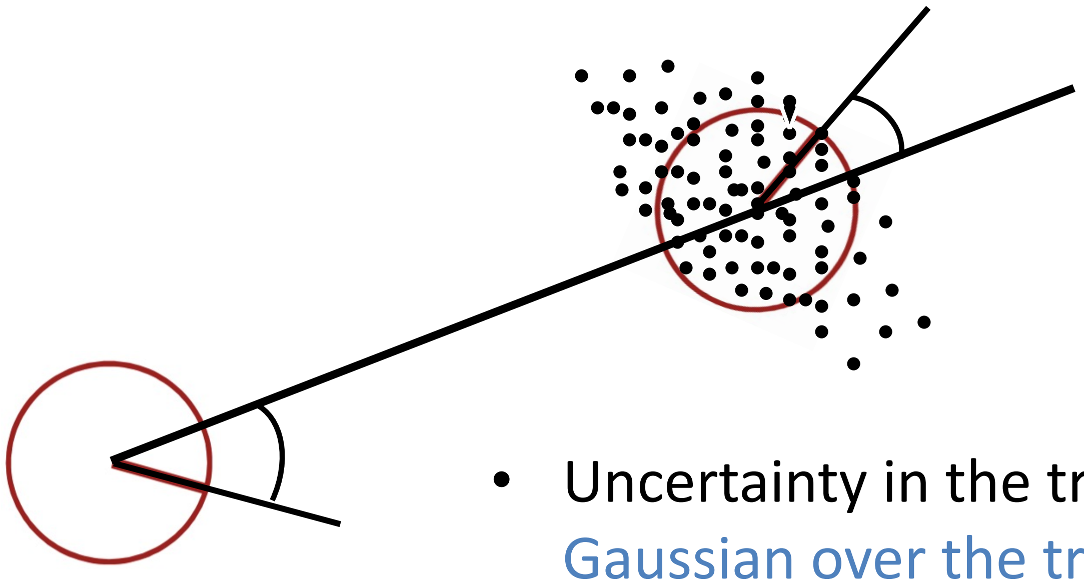
Motion Model Reminder



Decompose the motion into

- Traveled distance
- Start rotation
- End rotation

Motion Model Reminder



- Uncertainty in the translation of the robot:
  Gaussian over the traveled distance
- Uncertainty in the rotation of the robot:
  Gaussians over start and end rotation
- For each particle, draw a new pose by sampling from these three individual normal distributions

Mobile Robot Localization Using Particle Filters (1)

- Each particle is a potential pose of the robot
- The set of weighted particles approximates the posterior belief about the robot’s pose (target distribution)

Mobile Robot Localization Using Particle Filters (2)

- Particles are drawn from the motion model (proposal distribution)
- Particles are weighted according to the observation model (sensor model)
- Particles are resampled according to the particle weights

Mobile Robot Localization Using Particle Filters (3)

- Why is resampling needed?
  - We only have a finite number of particles
  - Without resampling: The filter is likely to loose track of the “good” hypotheses
  - Resampling ensures that particles stay in the meaningful area of the state space

SLAM Using Particle Filters – Grid-based SLAM

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

Rao-Blackwellization

- $$\underbrace{p(\overbrace{x_{1:t}}^{\text{Poses}}, \overbrace{m}^{\text{Map}} | \overbrace{z_{1:t}}^{\text{Observations}}, \overbrace{u_{0:t-1}}^{\text{Movements}})}_{\text{SLAM posterior}} = \underbrace{p(x_{1:t}, |z_{1:t}, u_{0:t-1})}_{\text{Robot path posterior}} \cdot \underbrace{p(m|x_{1:t}, z_{1:t})}_{\text{Mapping with known poses}}$$

Rao-Blackwellization

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

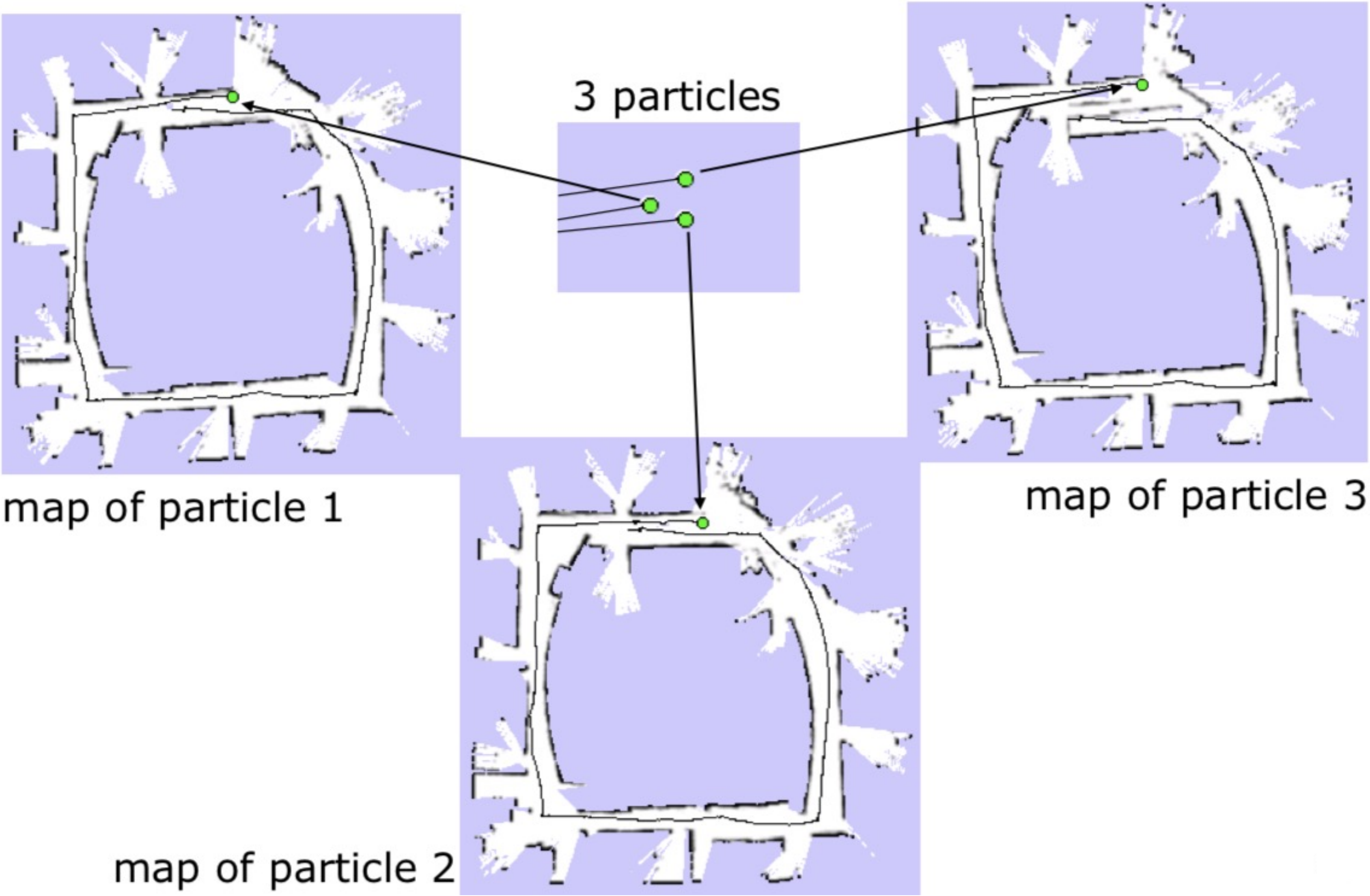
This is Localization, use MCL

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

Mapping with Rao- Blackwellized Particle Filters

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

Particle Filter Example



Problem

- Each map is quite big in case of grid maps
- 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

FastSLAM with Improved Odometry

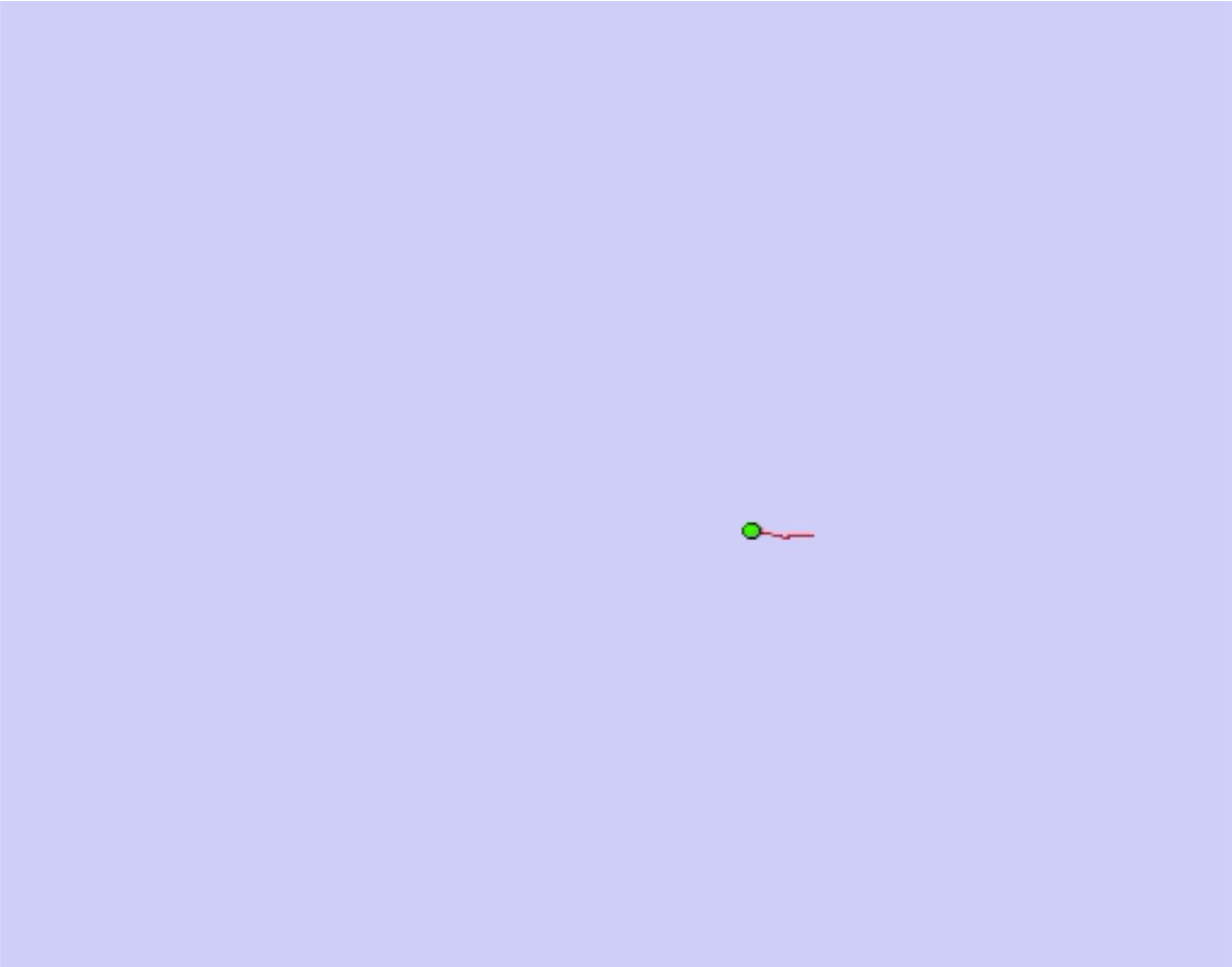
- Scan-matching provides a locally consistent pose correction
- Pre-correct short odometry sequences using scan-matching and use them as input to FastSLAM
- Fewer particles are needed, since the error in the input is smaller

Raw Odometry

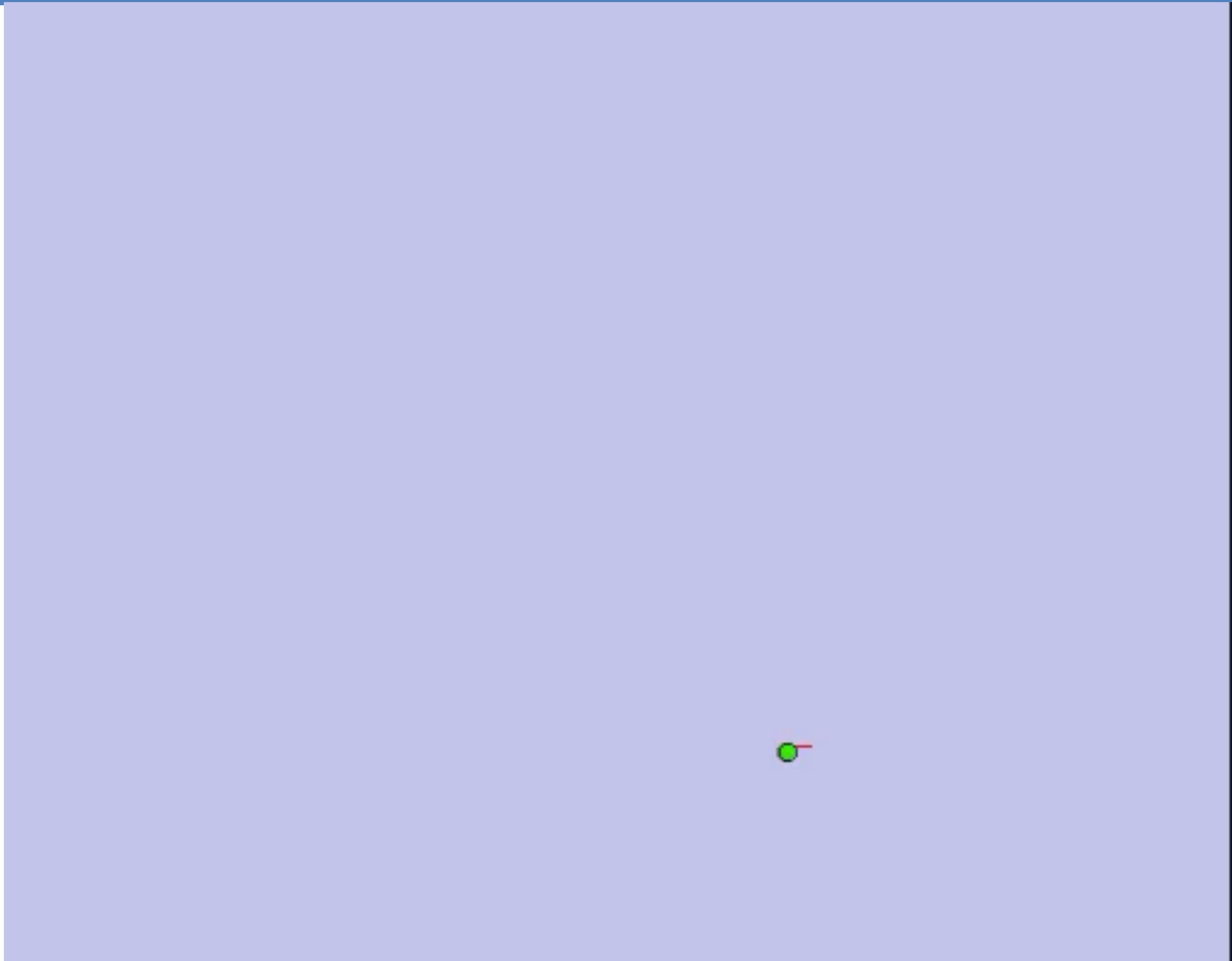
- Famous Intel Research Lab dataset (Seattle) by Dirk Hähnel

Courtesy of S. Thrun

http://robots.stanford.edu/videos.html

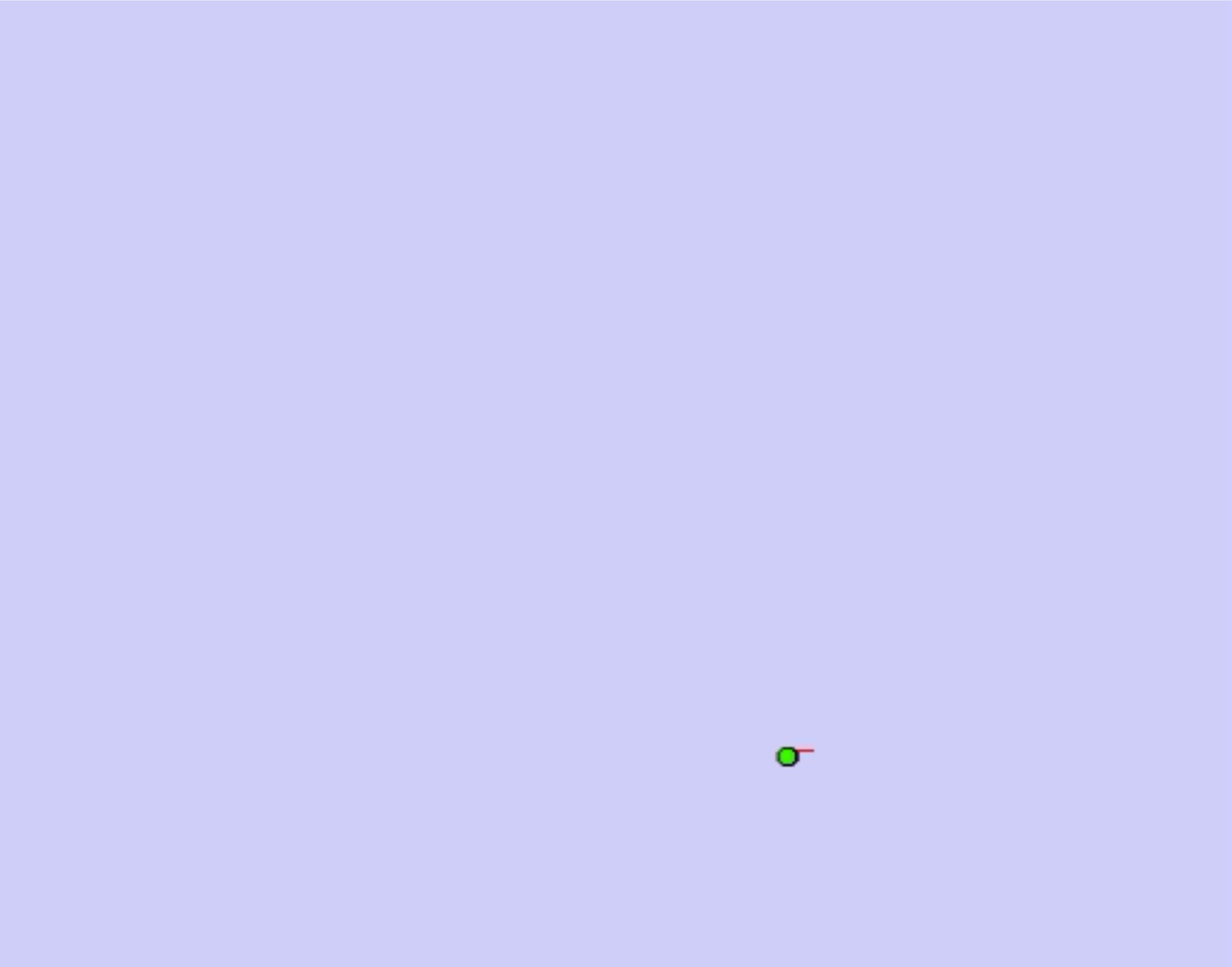


Scan Matching: compare to sensor data from previous scan



Courtesy of S. Thrun

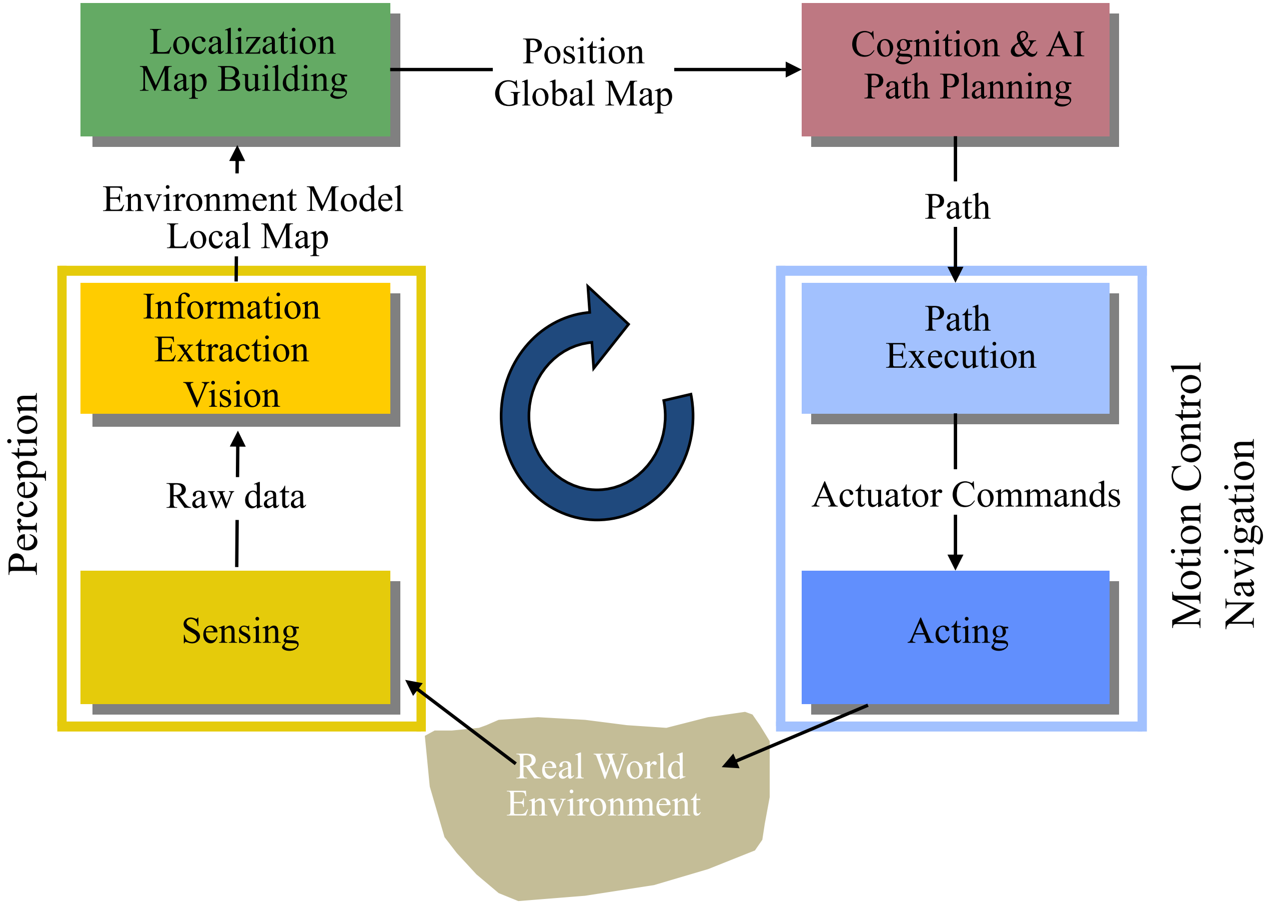
FastSLAM: Particle-Filter SLAM



Courtesy of S. Thrun

PLANNING

General Control Scheme for Mobile Robot Systems

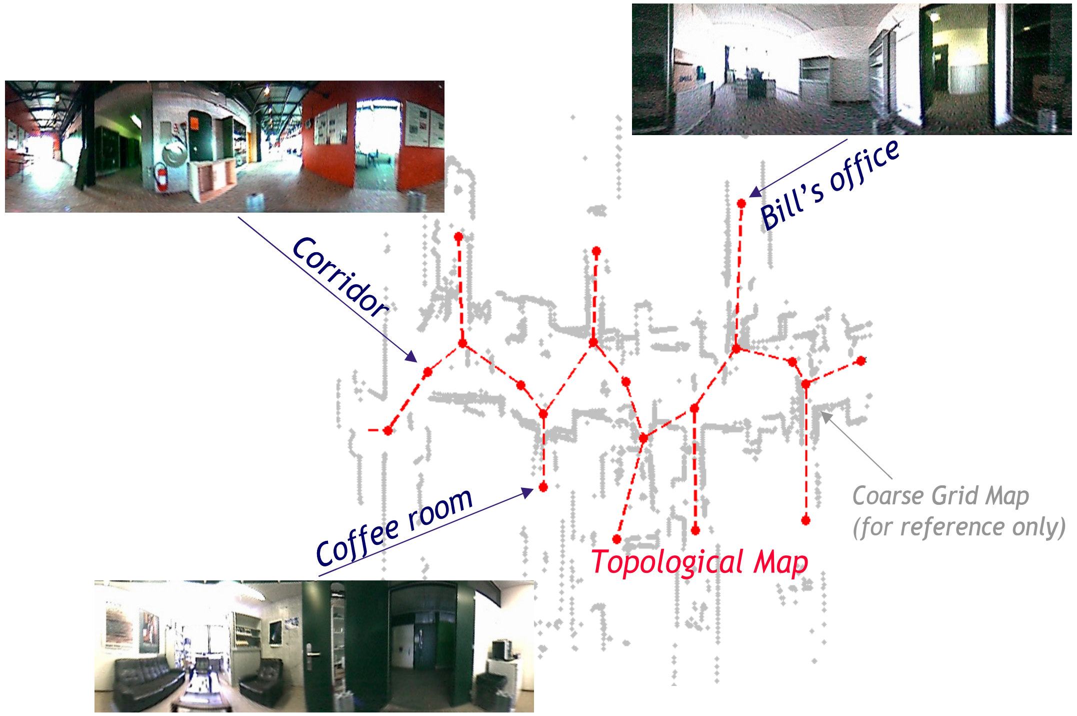


With material from Roland Siegwart and Davide Scaramuzza, ETH Zurich

The Planning Problem

- The problem: find a path in the work space (physical space) from the initial position to the goal position avoiding all collisions with the obstacles

- Assumption: there exists a good enough map of the environment for navigation.

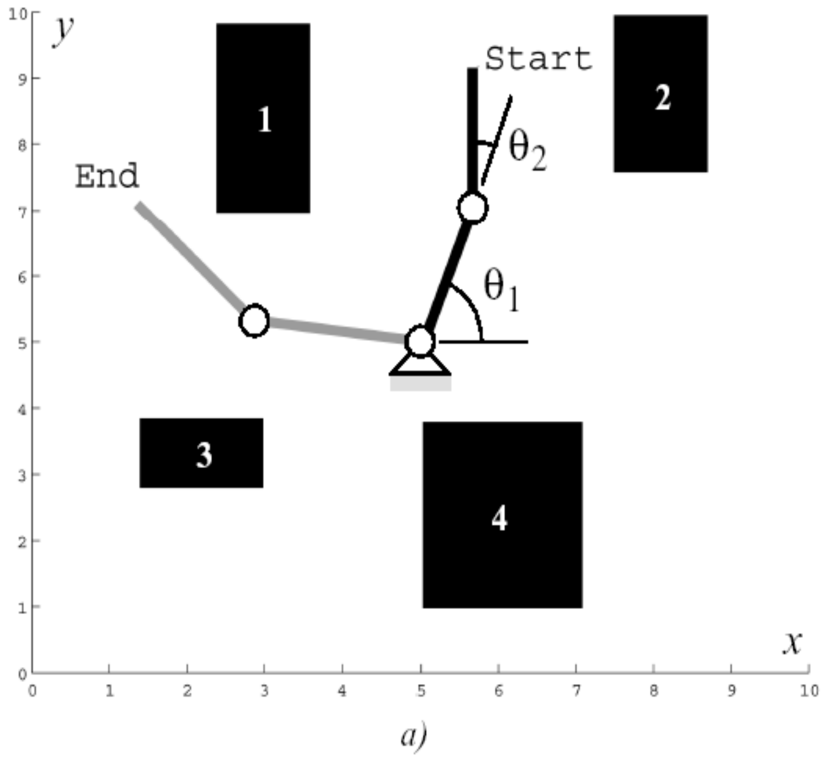


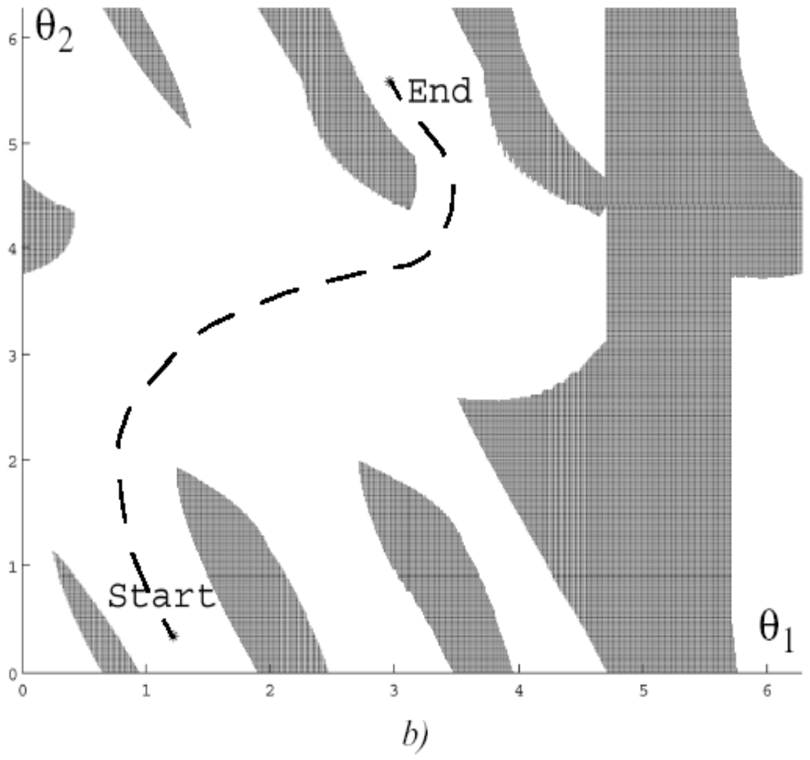
The Planning Problem

- We can generally distinguish between
  - (global) path planning and
  - (local) obstacle avoidance.
- First step:
  - Transformation of the map into a representation useful for planning
  - This step is planner-dependent
- Second step:
  - Plan a path on the transformed map
- Third step:
  - Send motion commands to controller
  - This step is planner-dependent (e.g. Model based feed forward, path following)

Work Space (Map) → Configuration Space

- State or configuration q can be described with k values q_i





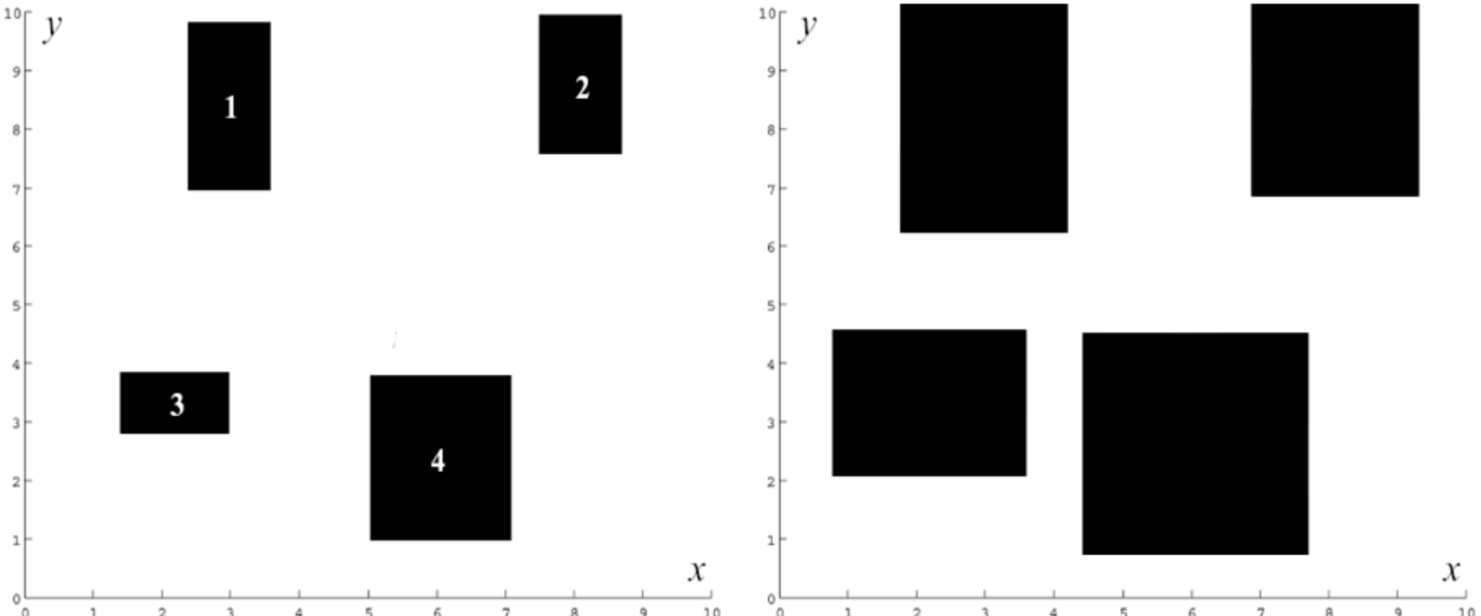
Work Space

Configuration Space:
the dimension of this
space is equal to the Degrees of Freedom (DoF)
of the robot

- What is the configuration space of a mobile robot?

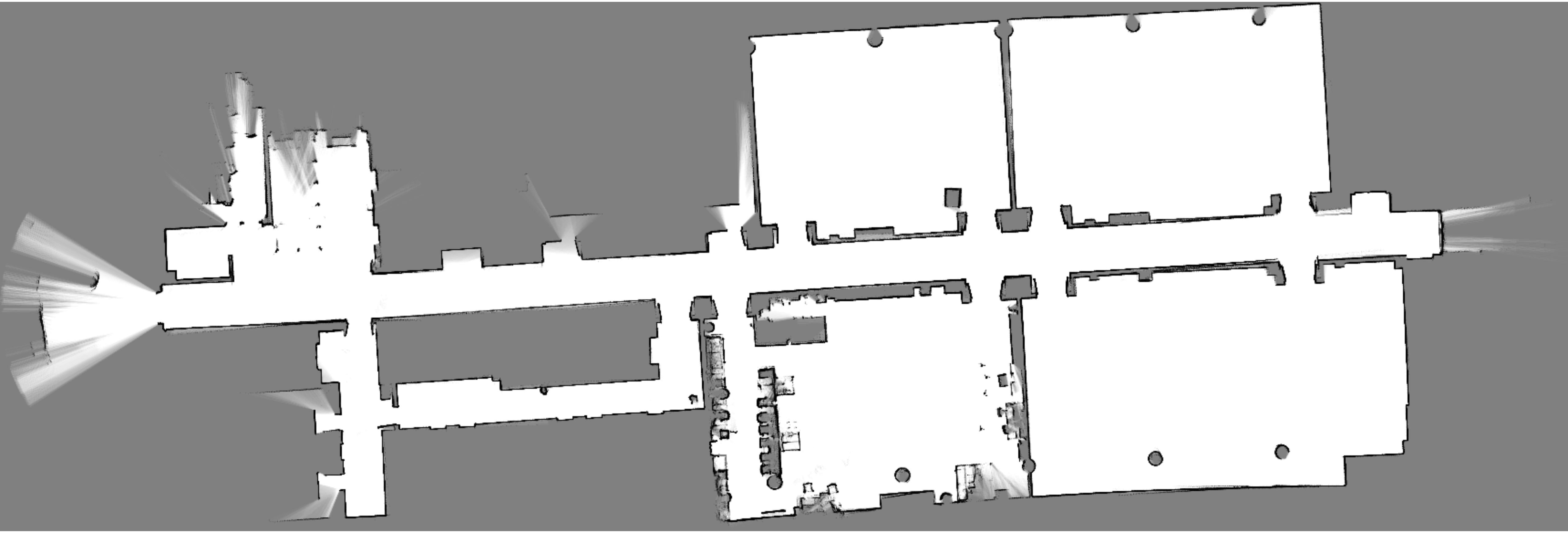
Configuration Space for a Mobile Robot

- Mobile robots operating on a flat ground (2D) have 3 DoF: (x, y, θ)
- Differential Drive: only two motors => only 2 degrees of freedom directly controlled (forward/ backward + turn) => non-holonomic
- Simplification: assume robot is holonomic and it is a point => configuration space is reduced to 2D (x,y)
- => inflate obstacle by size of the robot radius to avoid crashes => obstacle growing



Typical Configuration Space: Occupancy grid

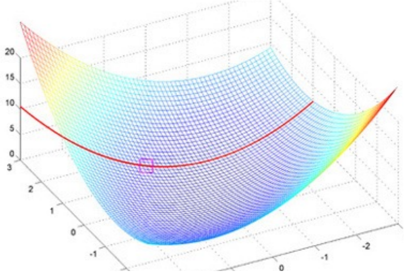
- Fixed cell decomposition: occupancy grid example: STAR Center



Path Planning: Overview of Algorithms

1. Optimal Control

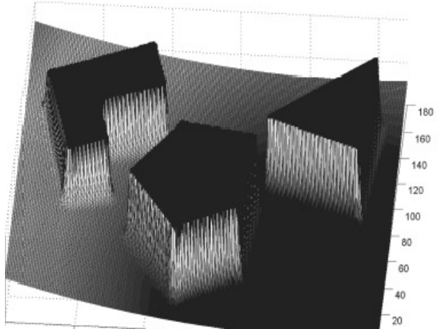
- Solves truly optimal solution
- Becomes intractable for even moderately complex as well as nonconvex problems



Source: http://mitocw.udsm.ac.tz

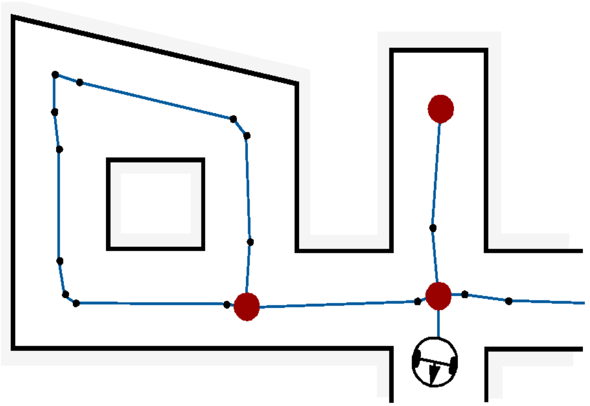
2. Potential Field

- Imposes a mathematical function over the state/configuration space
- Many physical metaphors exist
- Often employed due to its simplicity and similarity to optimal control solutions

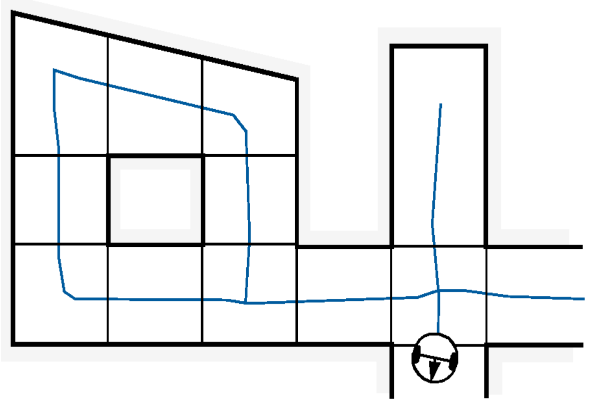


3. Graph Search

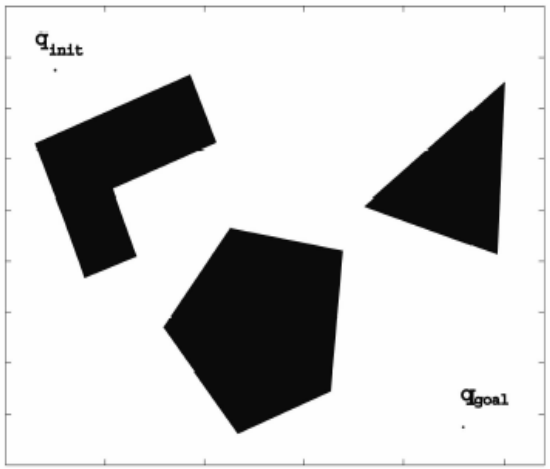
- Identify a set edges between nodes within the free space



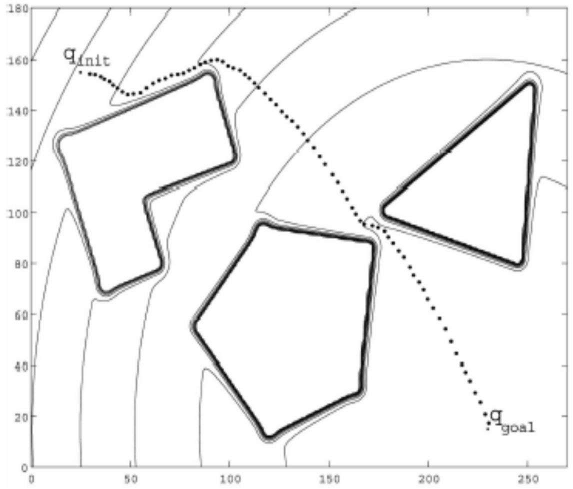
- Where to put the nodes?

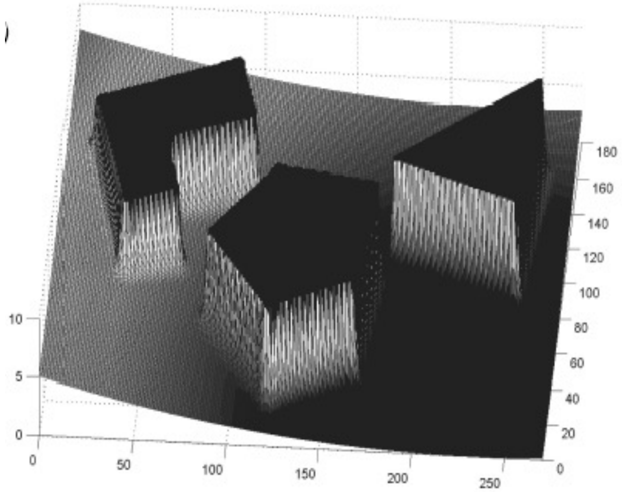


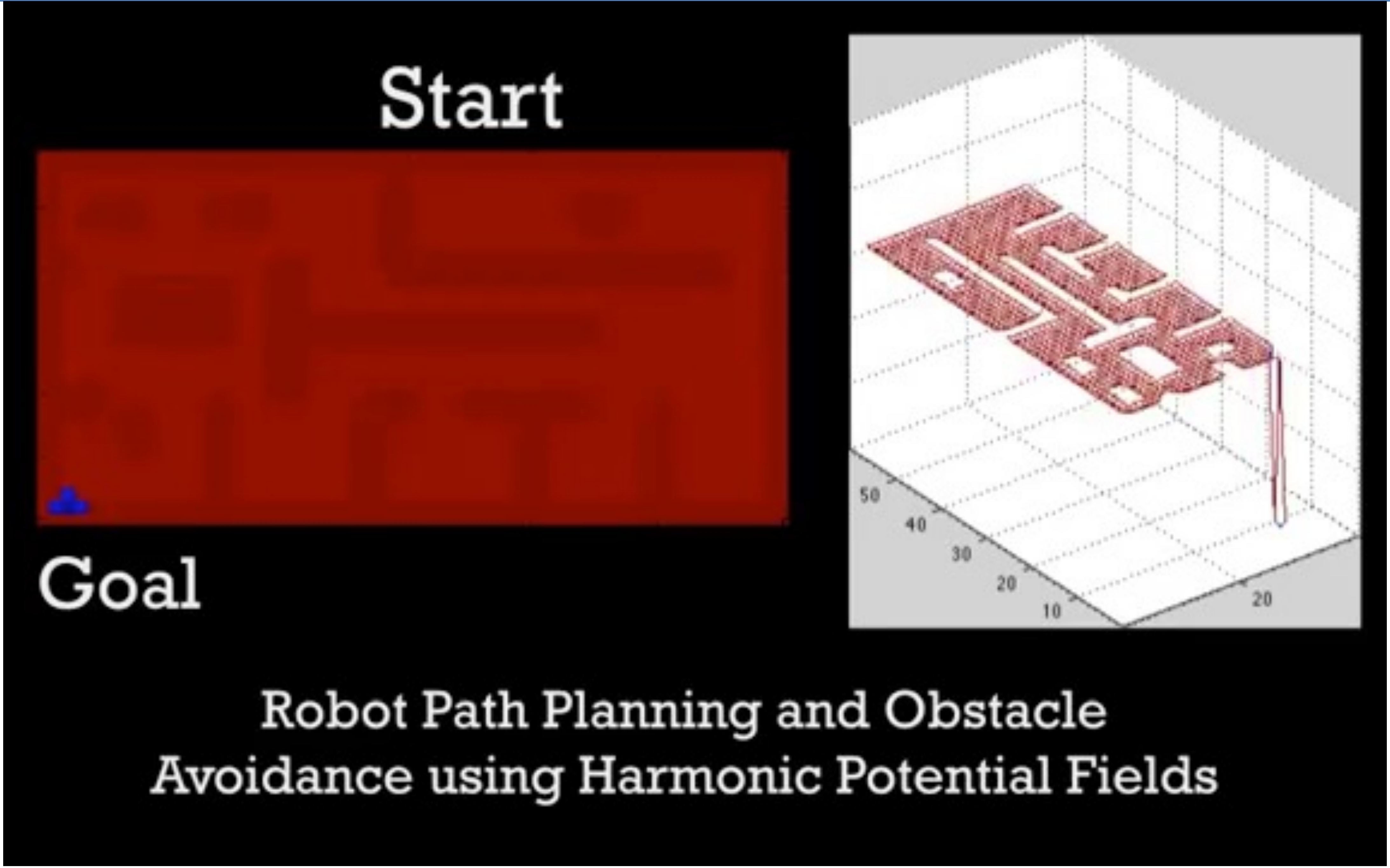
Potential Field Path Planning Strategies



- Robot is treated as a *point under the influence* of an artificial potential field.
- Operates in the continuum
  - Generated robot movement is similar to a ball rolling down the hill
  - Goal generates attractive force
  - Obstacle are repulsive forces






Start
Goal
50
40
30
20
10
20
Robot Path Planning and Obstacle
Avoidance using Harmonic Potential Fields

Potential Field Path Planning: Potential Field Generation

- Generation of potential field function *U(q)*
  - attracting (goal) and repulsing (obstacle) fields
  - summing up the fields
  - functions must be differentiable
- Generate artificial force field *F(q)*

$$F(q) = -\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

- Set robot speed (v_x, v_y) proportional to the force $F(q)$ generated by the field
  - the force field drives the robot to the goal
  - if robot is assumed to be a point mass
  - Method produces both a plan *and* the corresponding control

Potential Field Path Planning: Attractive Potential Field

- Parabolic function representing the Euclidean distance $\rho_{goal} = \left\| q - q_{goal} \right\|$ to the goal

$$\begin{aligned} U_{att}(q) &= \frac{1}{2} k_{att} \cdot \rho_{goal}^2(q) \\ &= \frac{1}{2} k_{att} \cdot (q - q_{goal})^2 \end{aligned}$$

- Attracting force converges linearly towards 0 (goal)

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q) \\ &= k_{att} \cdot (q - q_{goal}) \end{aligned}$$

Potential Field Path Planning: Repulsing Potential Field

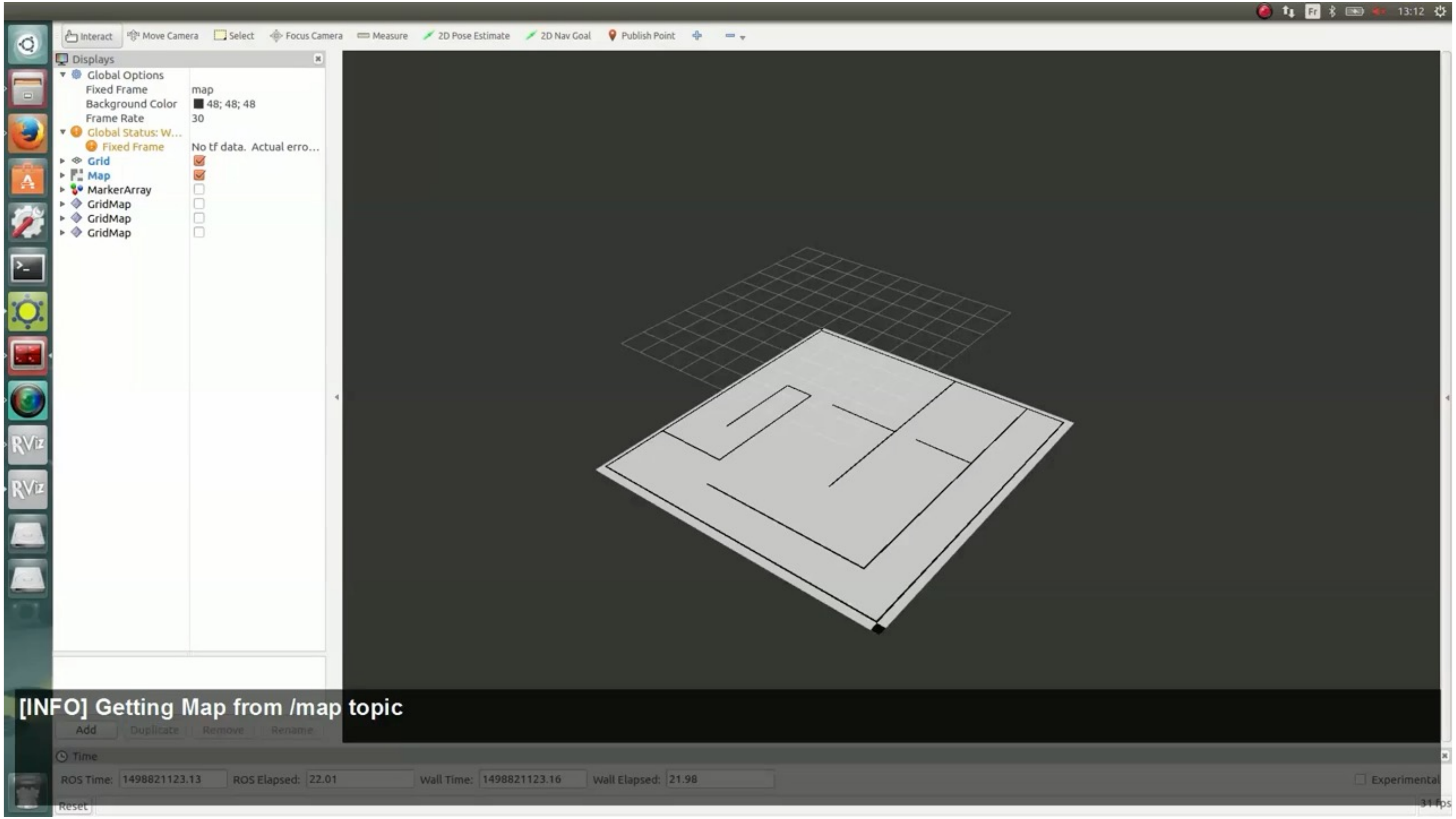
- Should generate a barrier around all the obstacle
  - strong if close to the obstacle
  - not influence if far from the obstacle

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

  - $\rho(q)$: minimum distance to the object
  - Field is positive or zero and *tends to infinity* as q gets closer to the object

ROS Grid Map Package

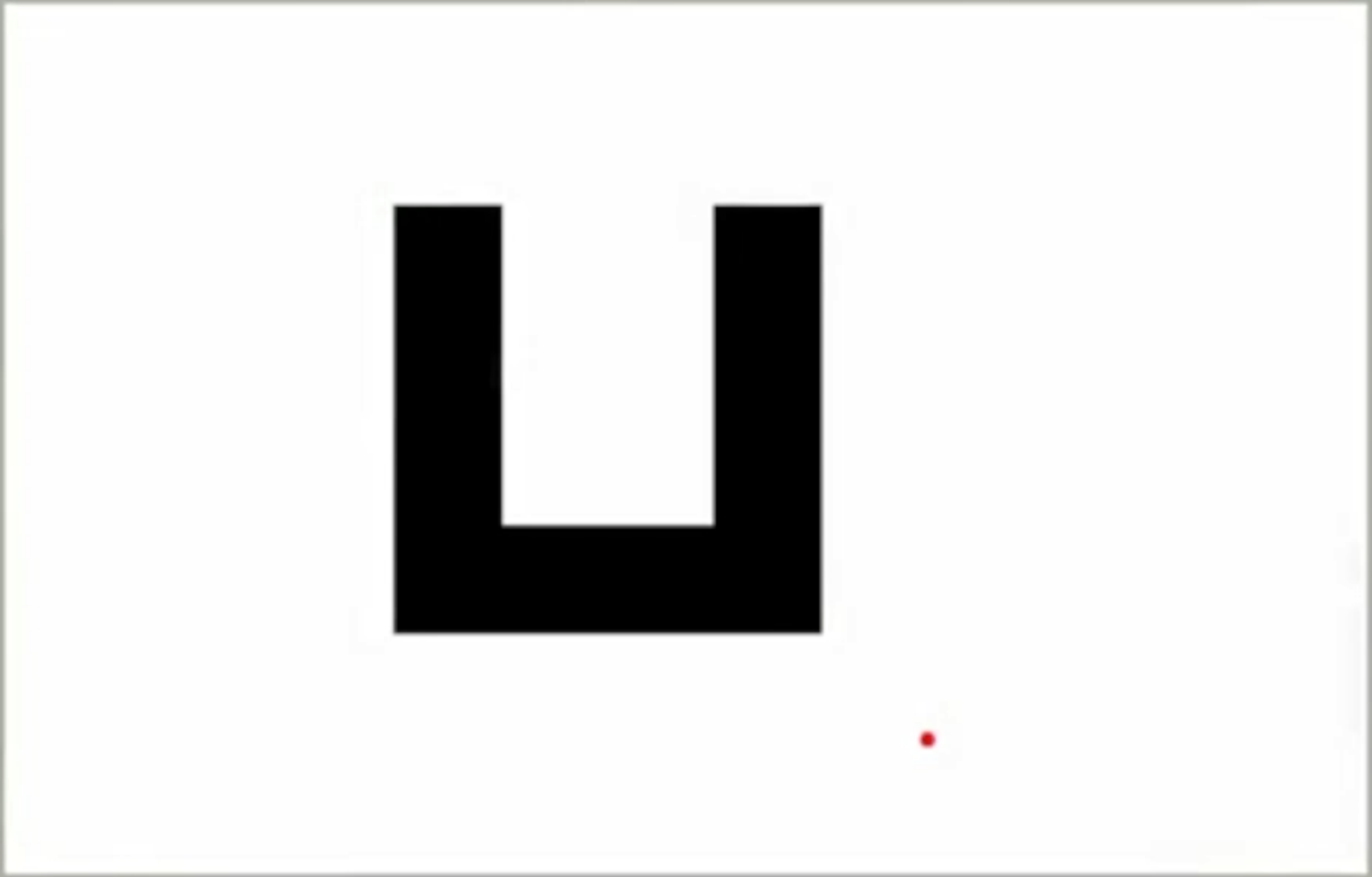
http://wiki.ros.org/grid_map



Potential Field Path Planning:

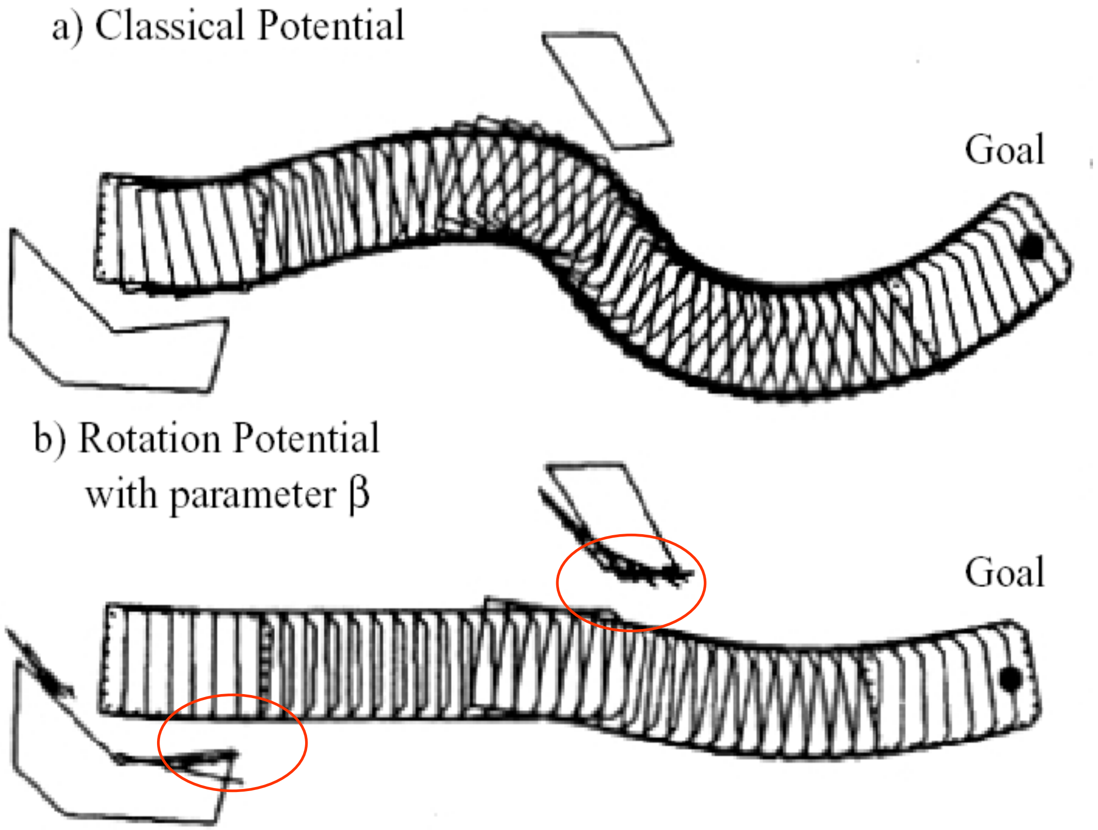
- Notes:
  - Local minima problem exists
  - Problem is getting more complex if the robot is not considered as a point mass
  - If objects are non-convex there exists situations where several minimal distances exist → can result in oscillations

Example Configuration Space



Potential Field Path Planning: Extended Potential Field Method

- Additionally a *rotation potential field* and a *task potential field* is introduced
- Rotation potential field
  - force is also a function of robots orientation relative to the obstacles. This is done using a gain factor that reduces the repulsive force when obstacles are parallel to robot's direction of travel
- Task potential field
  - Filters out the obstacles that should not influence the robots movements, i.e. only the obstacles in the sector in front of the robot are considered



Khatib and Chatila