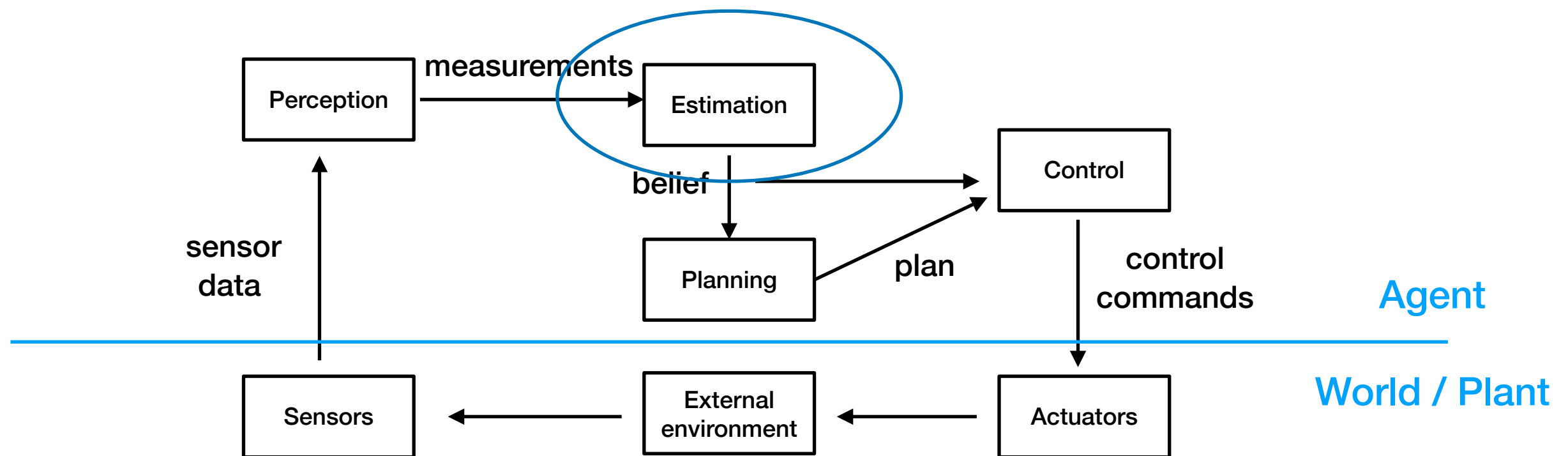


Simultaneous Localization and Mapping (SLAM)



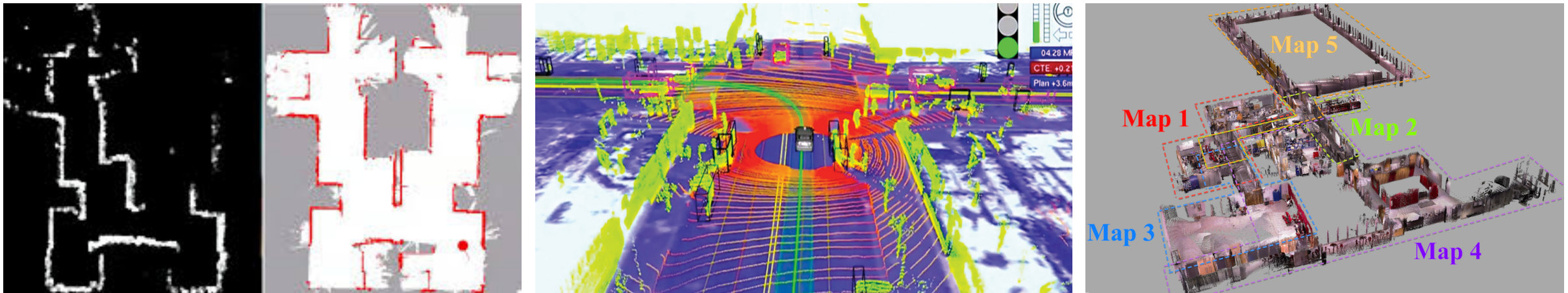
Big picture



We are going back to estimation, but now consider that we don't have a model of the environment a priori

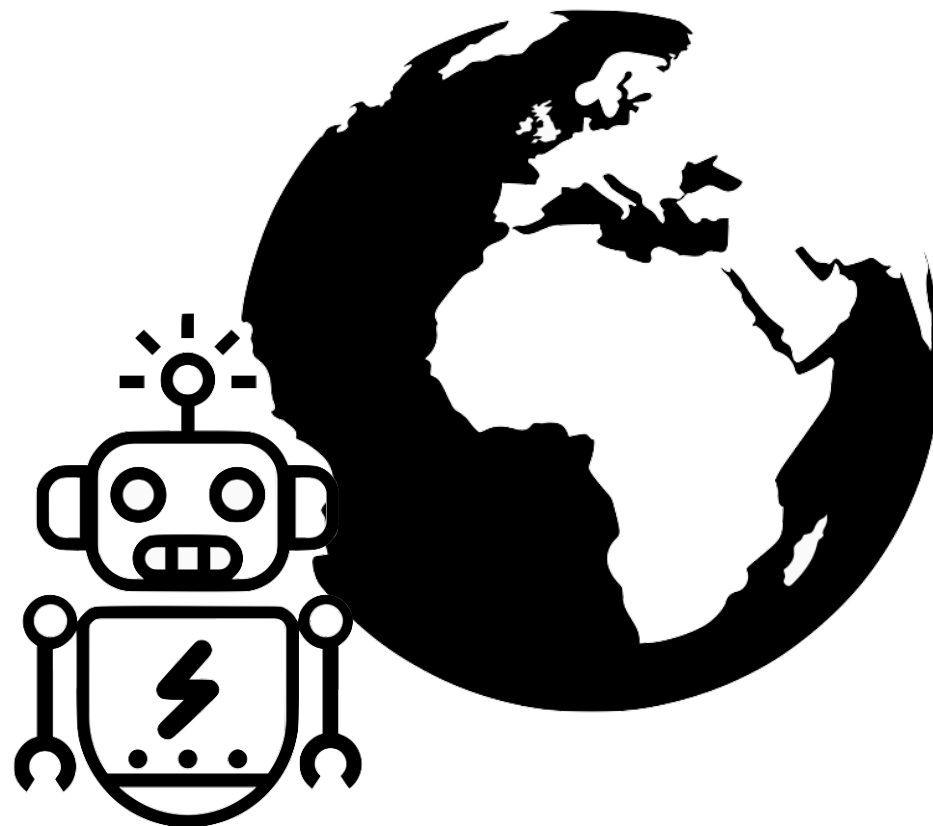
Mapping

- “What is around the robot?”
- Given a robot’s location, construct a representation of the world around it.
- How do we represent the map?
- What assumptions do we make about the map? (e.g., static?)



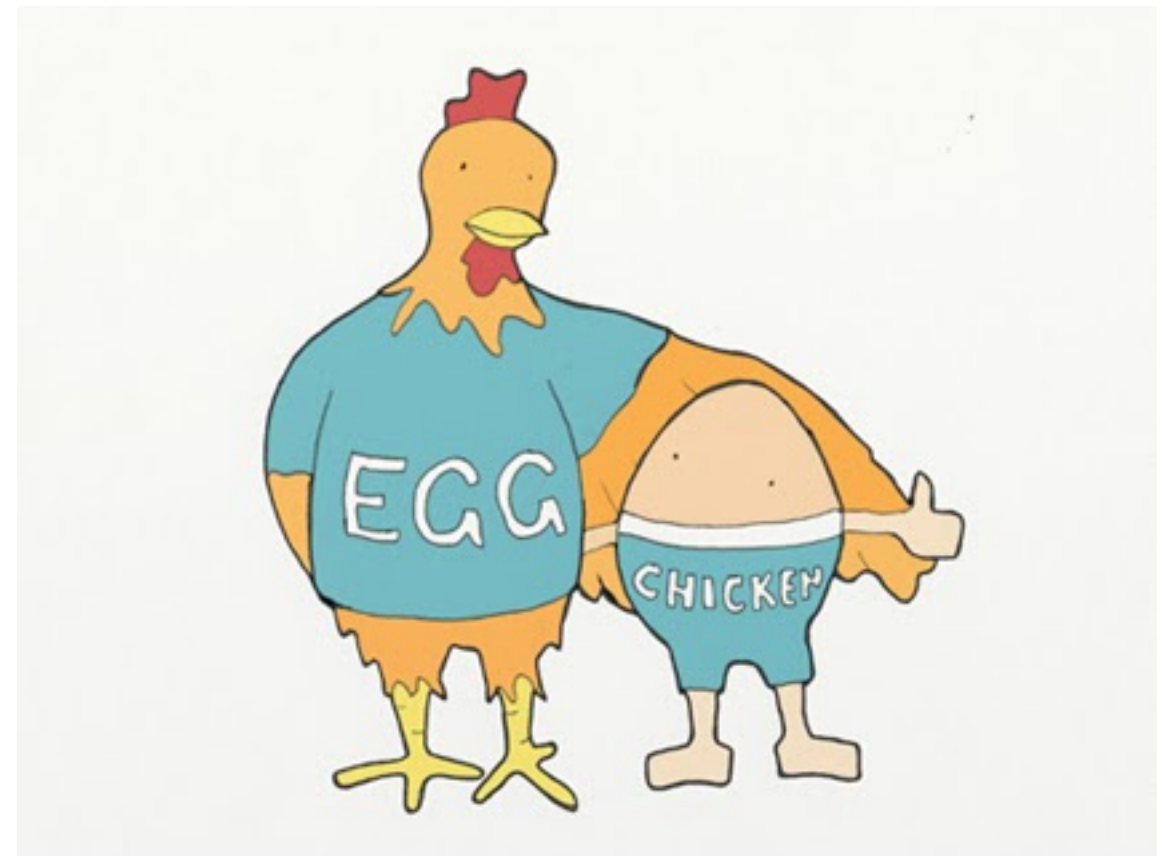
Localization

- “Where is the robot?”
- Given a map of the robot’s world, determine where the robot is.
- We have already seen various solutions to the localization problem (Gaussian filters, non-parametric filters, etc.)



SLAM: Simultaneous Localization and Mapping

- But ...
- Localization assumes a map to be available
- Mapping assumes localization to be solved
- Chicken-and-egg problem



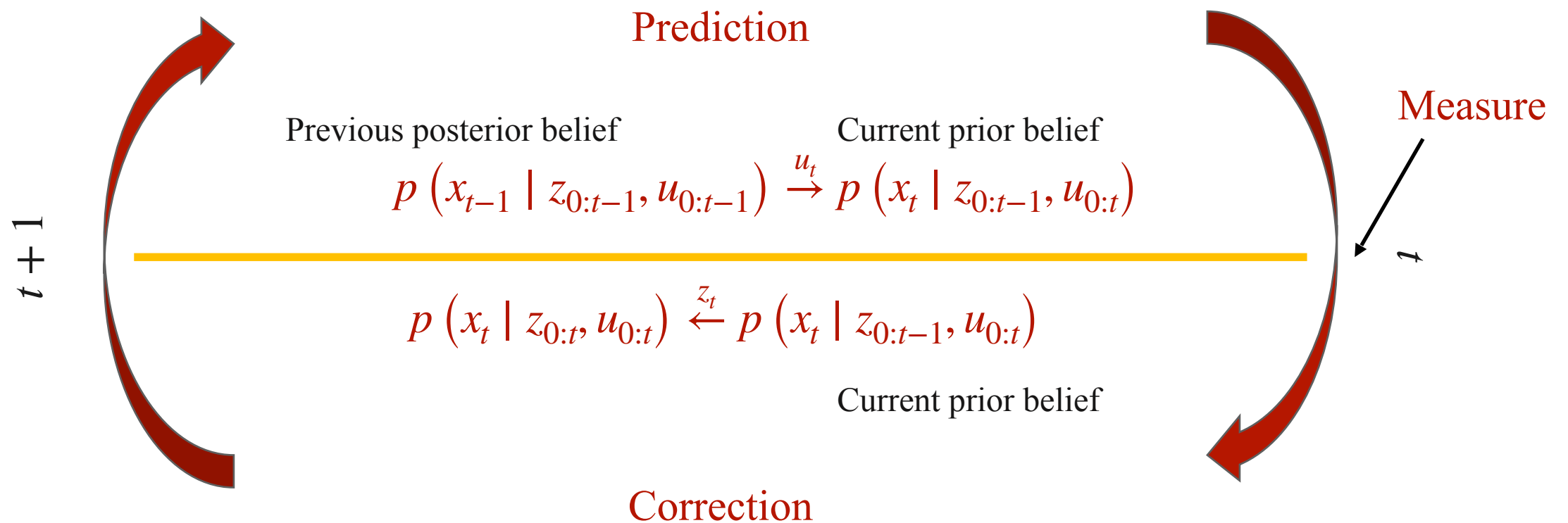
Recall the Bayes' Filter

But what is x_t ?

Find

- Recursive Estimate/Posterior belief:

$$bel(x_t) = p(x_t | z_{1:t}, u_{0:t}, x_0)$$

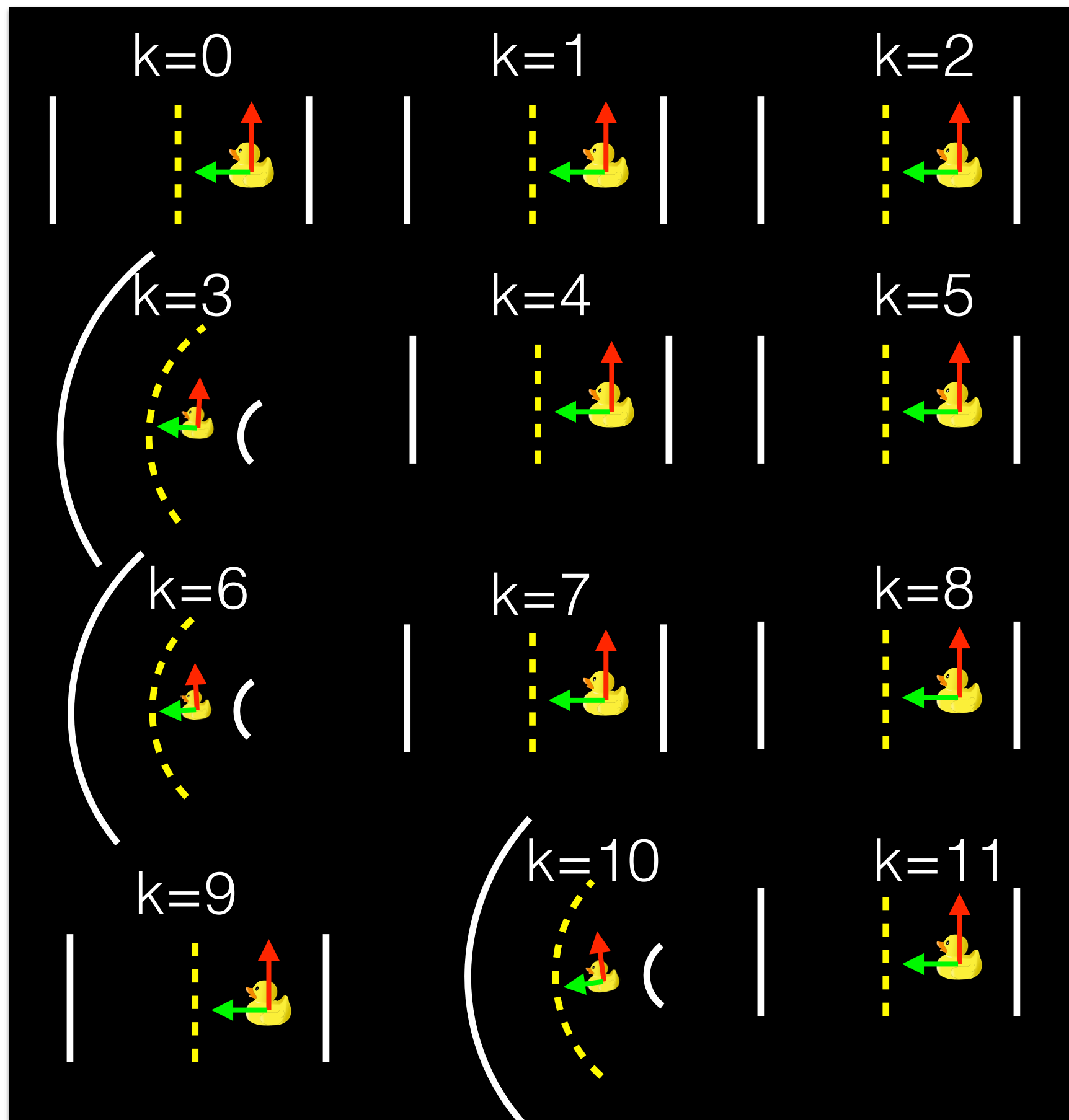
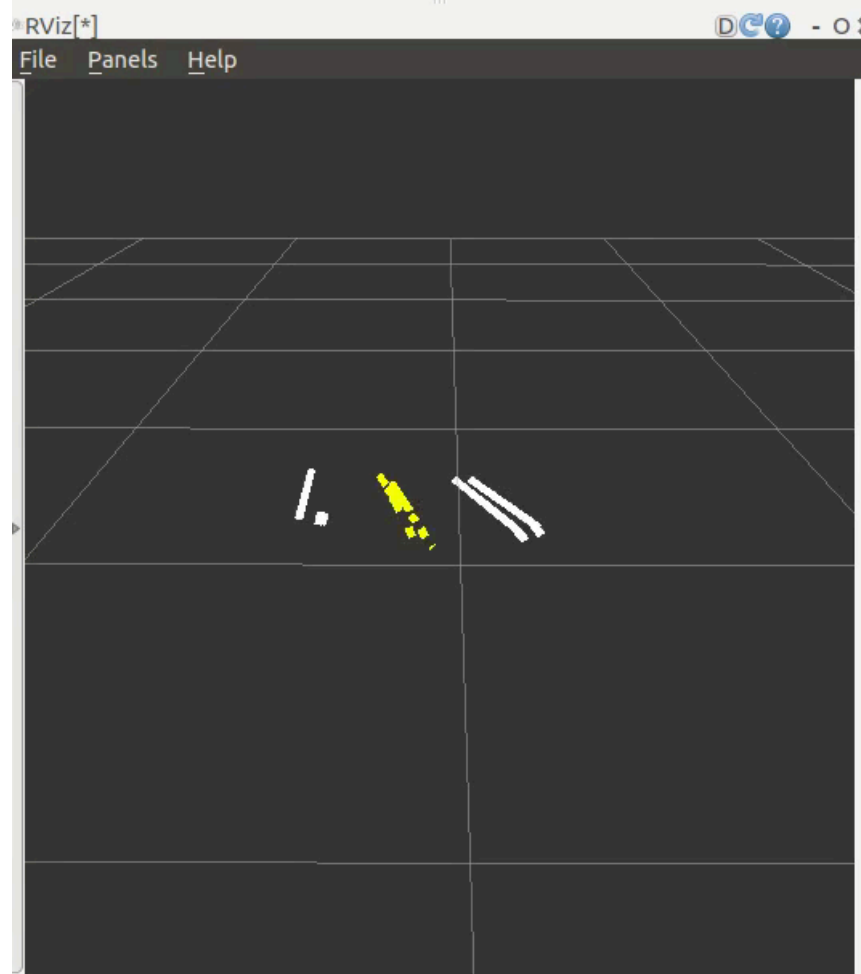
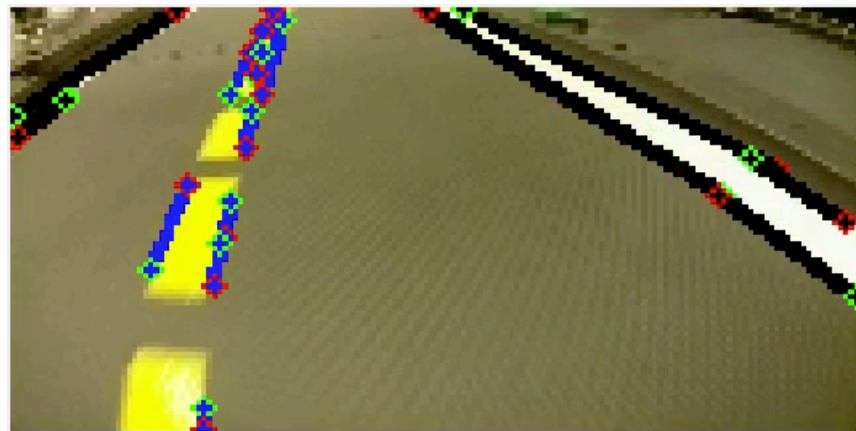


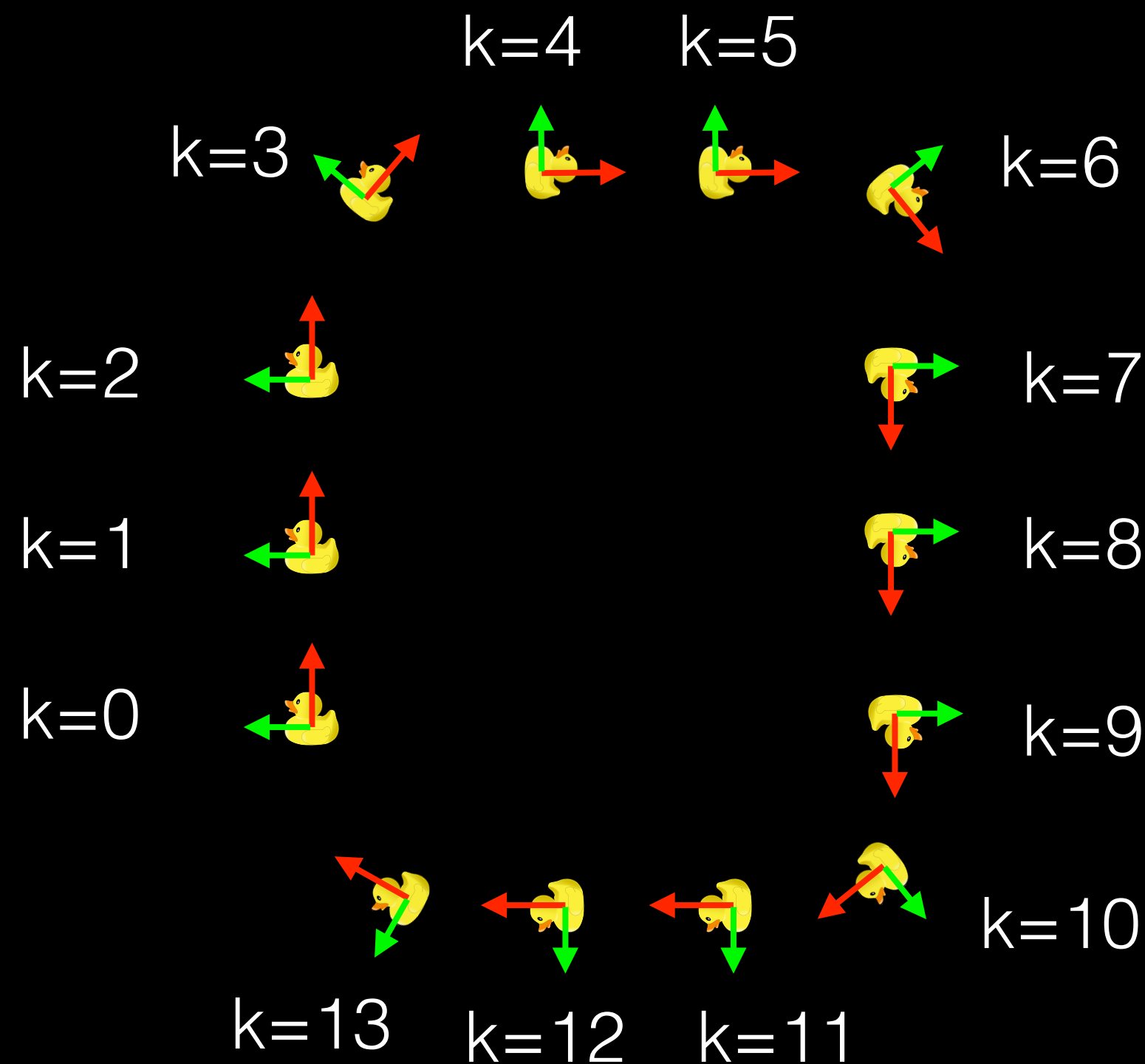
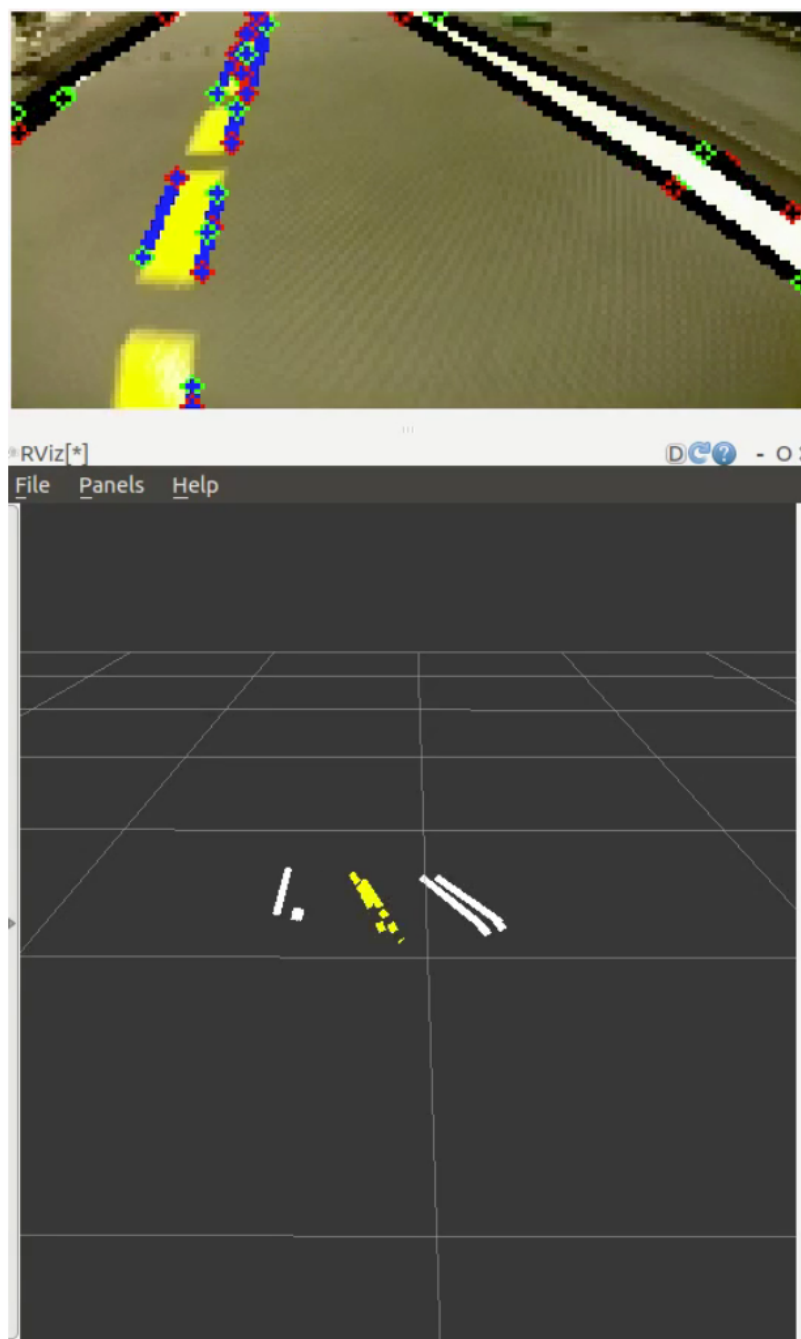
Key Considerations

- Map representation
- Batch vs. Online
- Data association (feature correspondences)

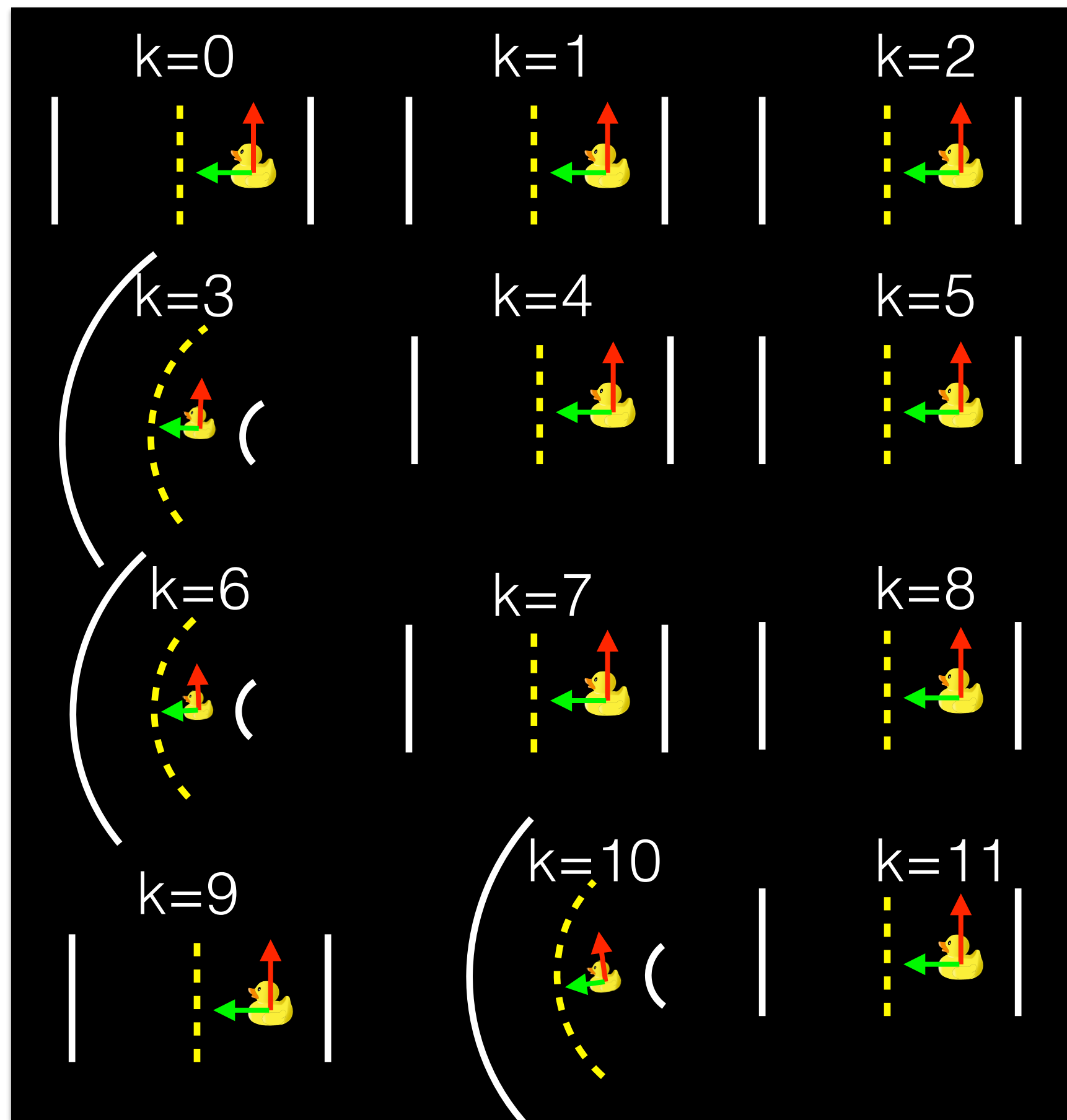
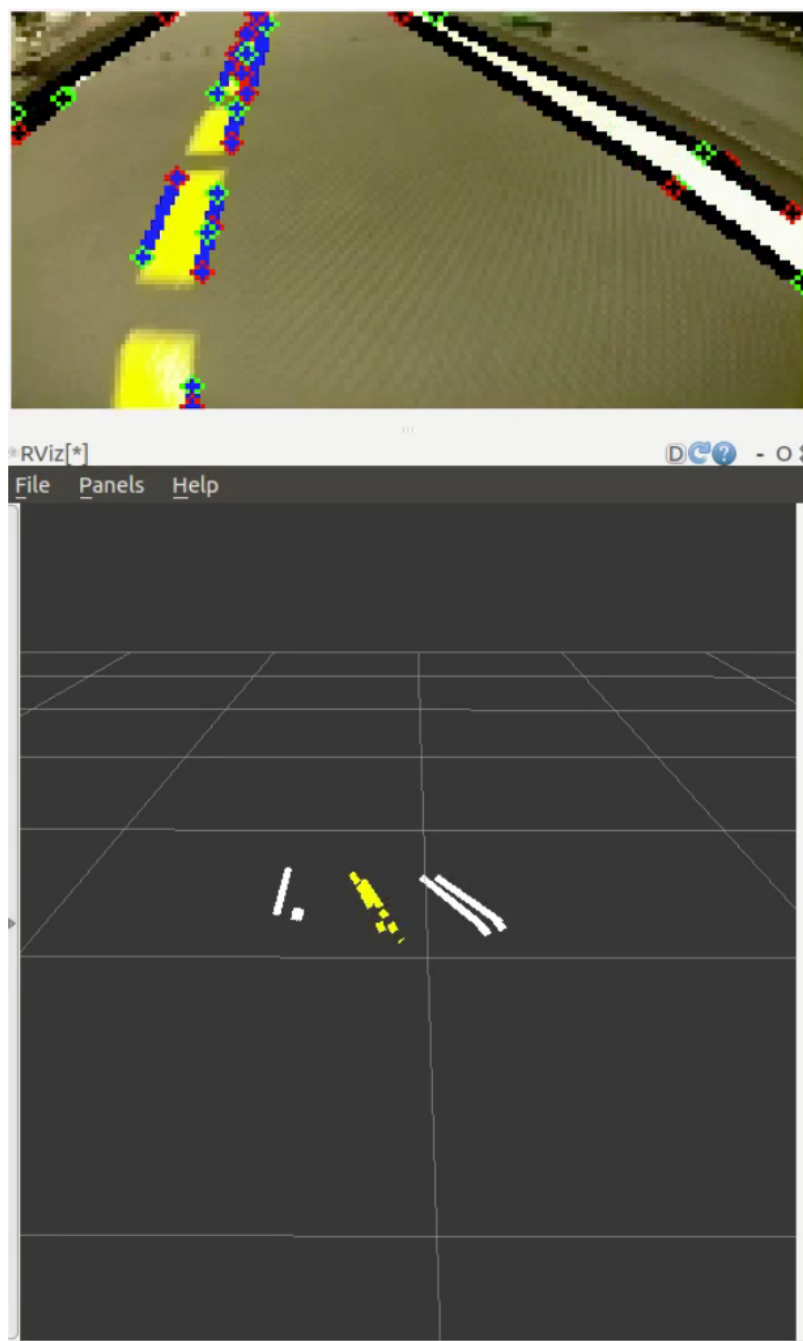
Mapping from Known Poses

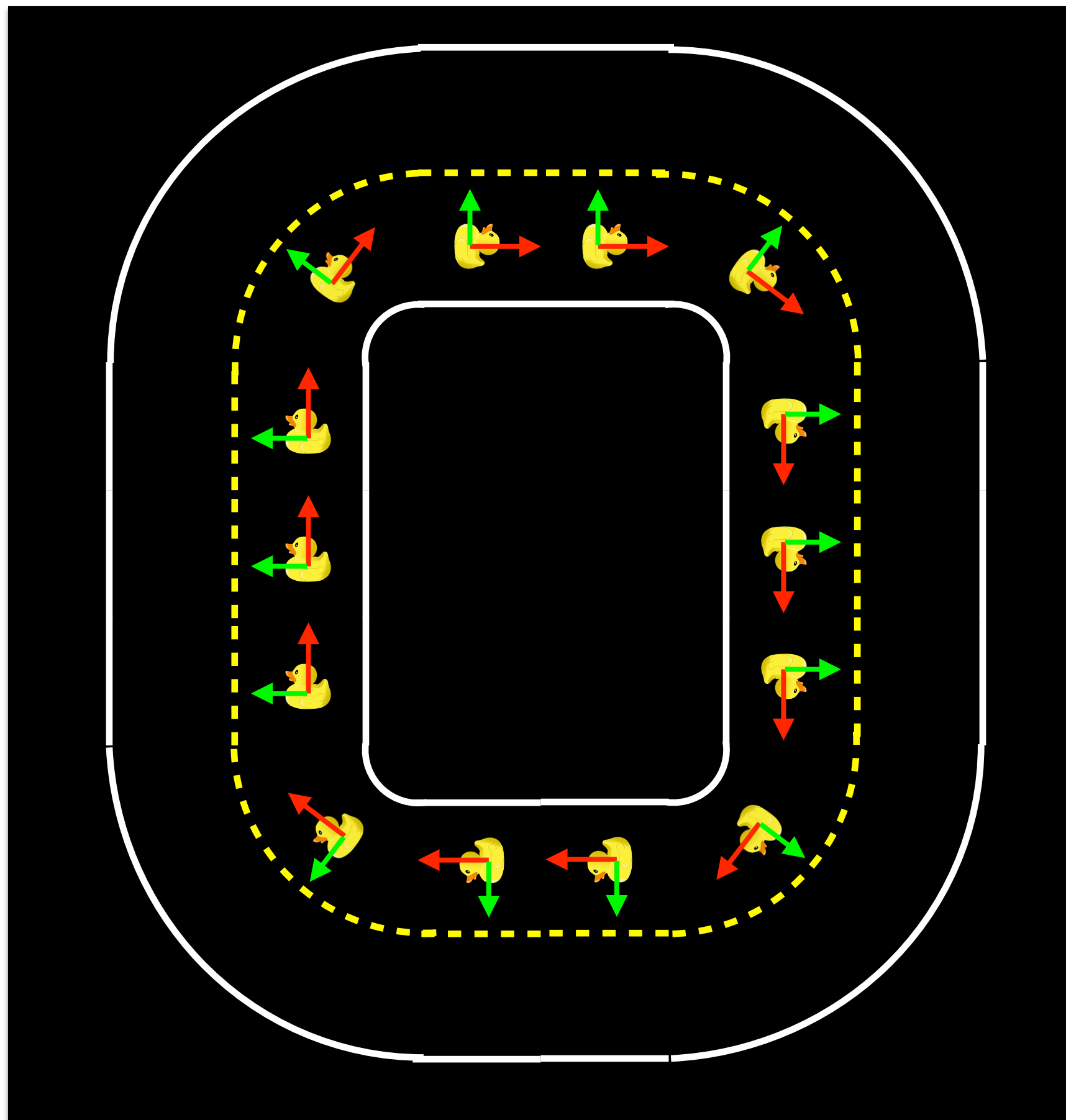
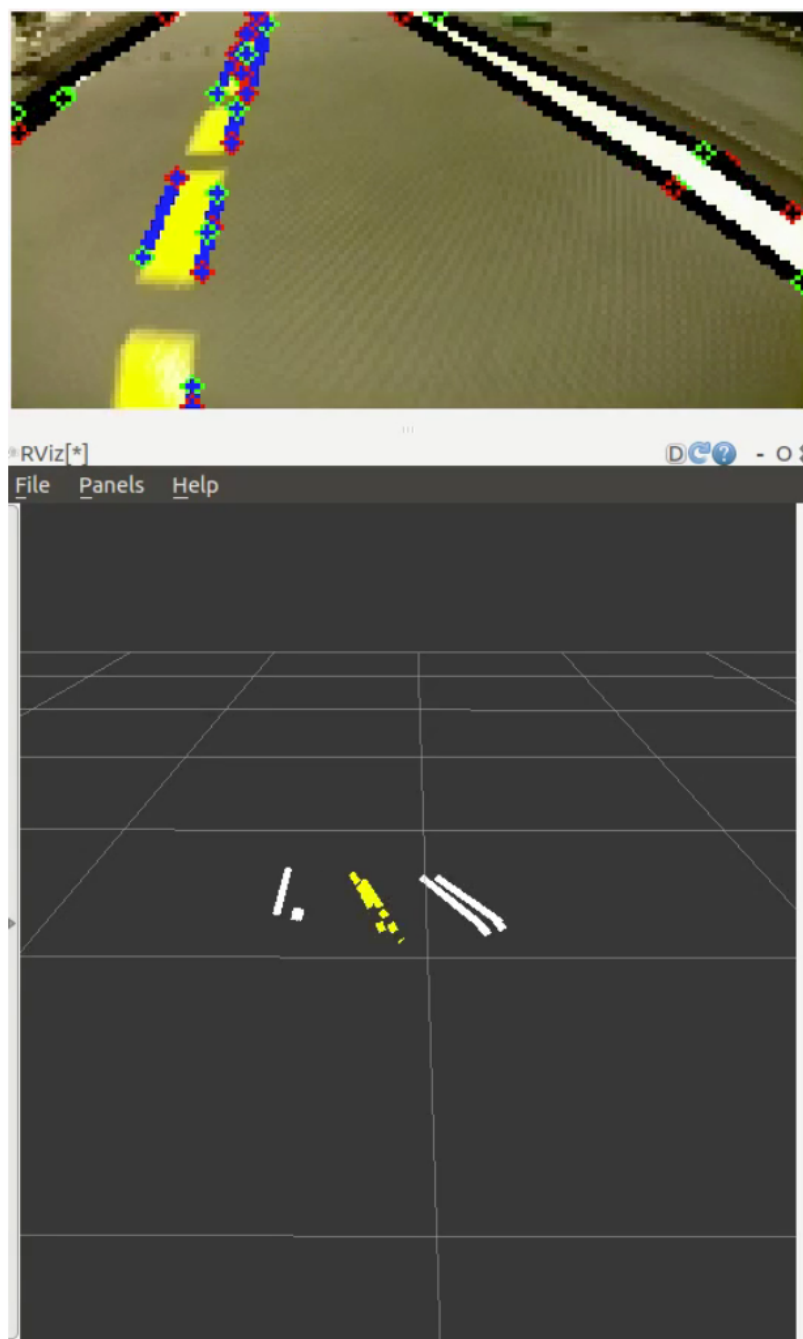
- Problem statement:
 - We know the trajectory of the robot and we have all of the measurements
 - We are only estimating the map (x_t is just the map)
 - How to represent it?
 - What assumptions should we make about it?
 - e.g. it's static?
 - $f(x_t, u_t, w_t) = x_{t-1} \rightarrow \bar{bel}(x_t) = bel(x_{t-1})$



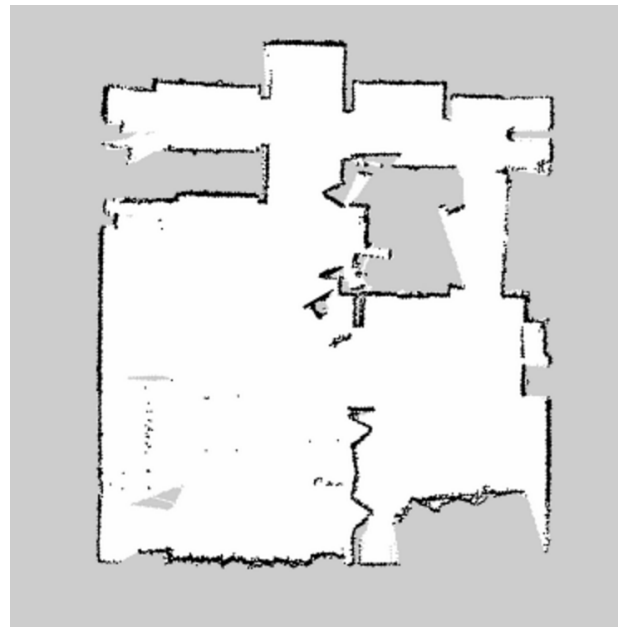


If trajectory were given:





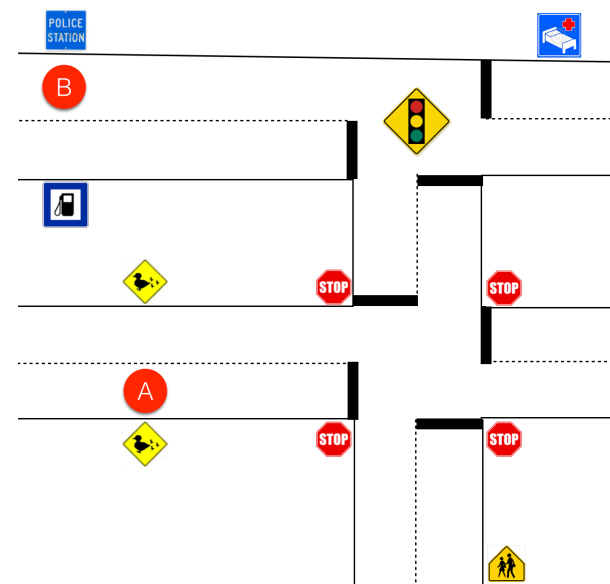
Map Representations



Occupancy grid map

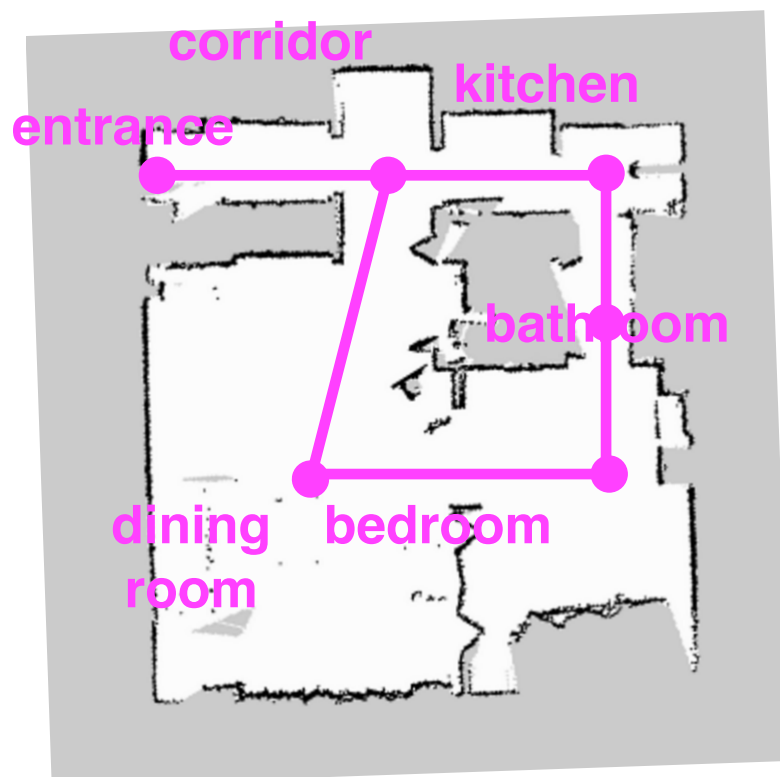
- metric information
- traversability, obstacles

□ = free ■ = obstacle ■ = unknown



Feature - based map

- metric information
- distinguishable landmarks
- no info about obstacles



Topological/semantic map

- no metric information
- semantic labels
- no info about obstacles

Point clouds, other 3D maps



Occupancy Grid Maps

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- **Key assumptions**
 - Occupancy of individual cells ($m[xy]$) is independent

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

Robot positions are known!

Updating Occupancy Grid Maps

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

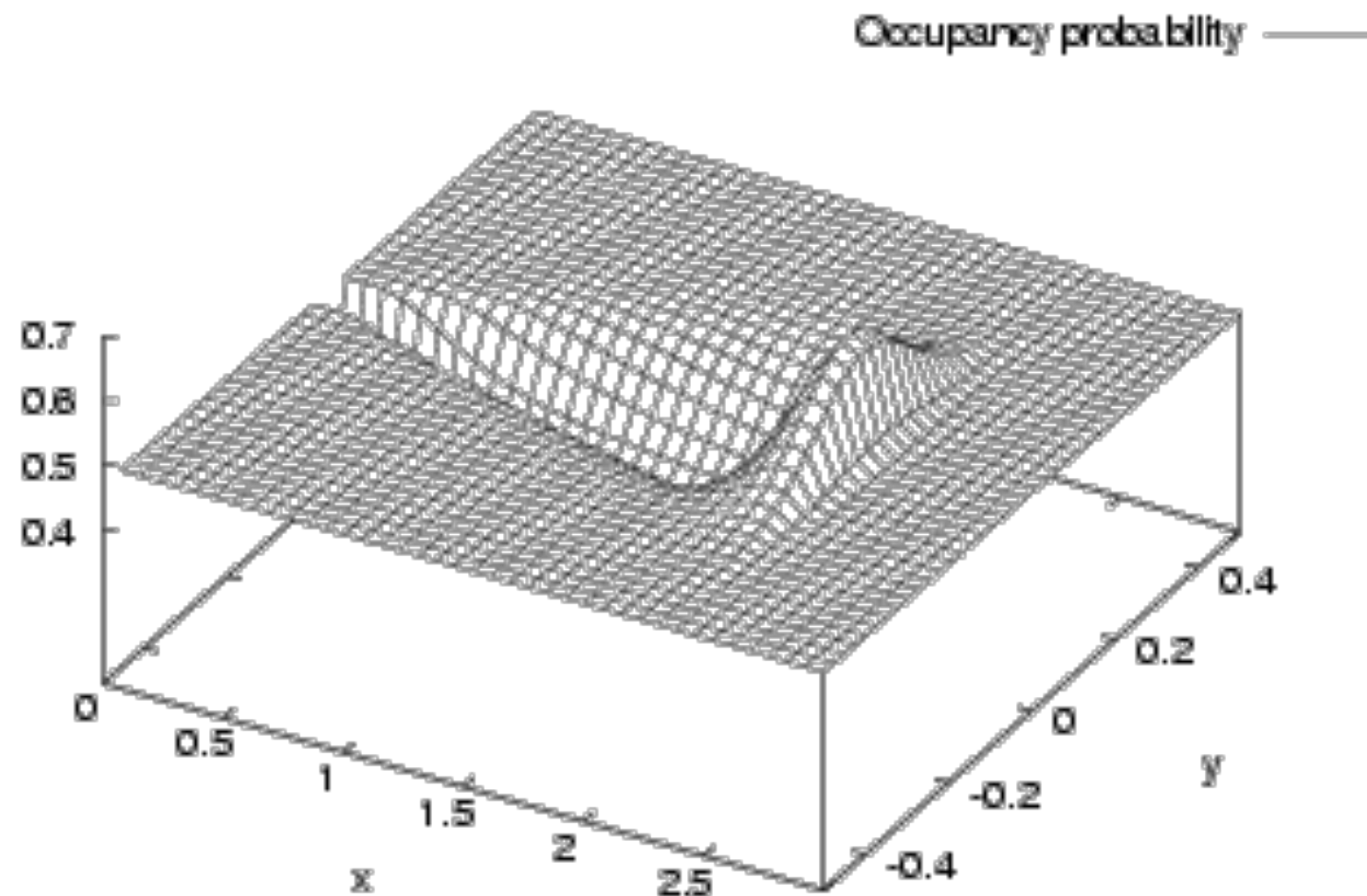
$$Bel(m_t^{[xy]}) = \eta p(z_t | m_t^{[xy]}) \int p(m_t^{[xy]} | m_{t-1}^{[xy]}, u_{t-1}) Bel(m_{t-1}^{[xy]}) dm_{t-1}^{[xy]}$$

- **Additional assumption**: Map is static

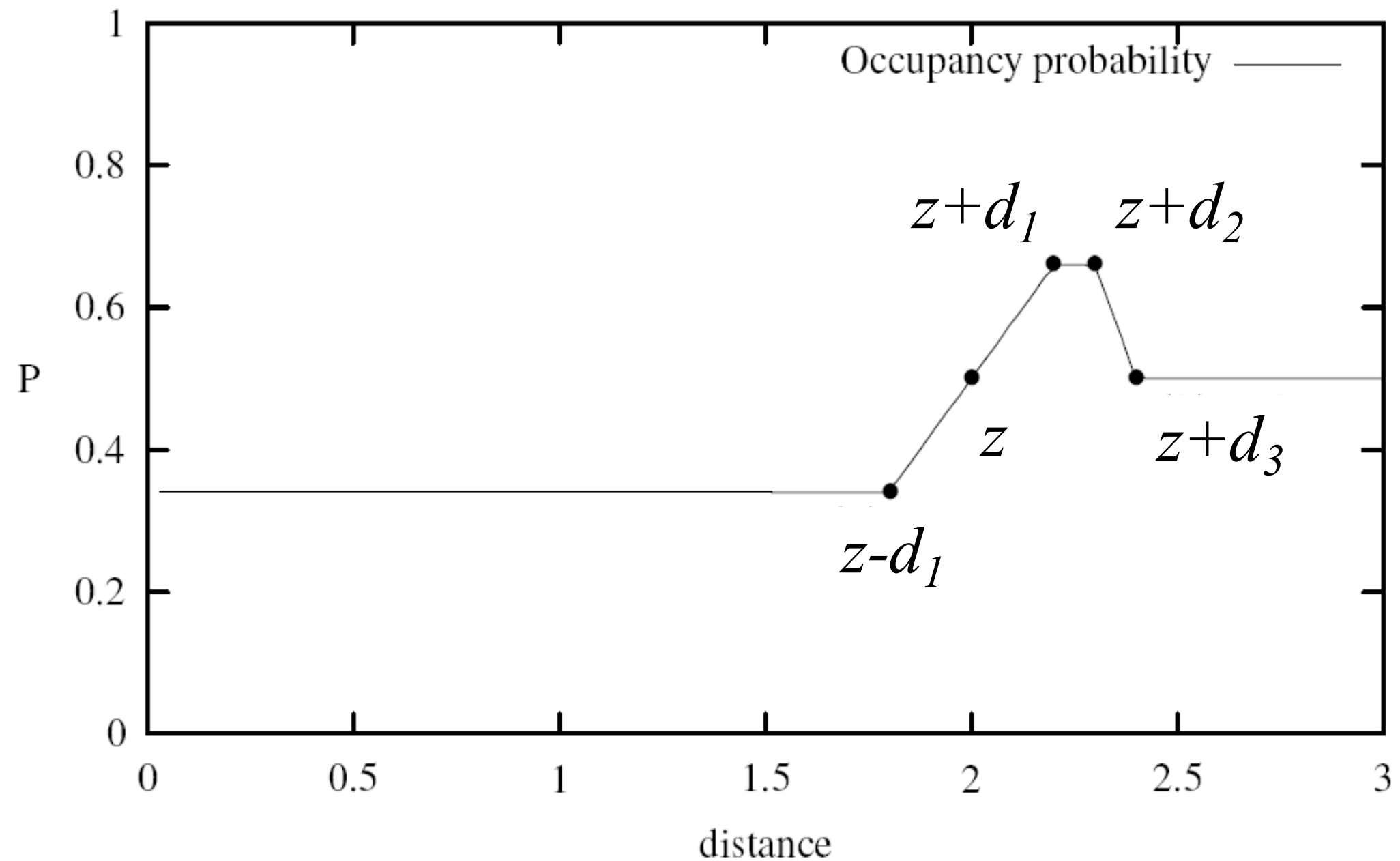
$$Bel(m_t^{[xy]}) = \eta p(z_t | m_t^{[xy]}) Bel(m_{t-1}^{[xy]})$$

Ultrasonic Sensor Model

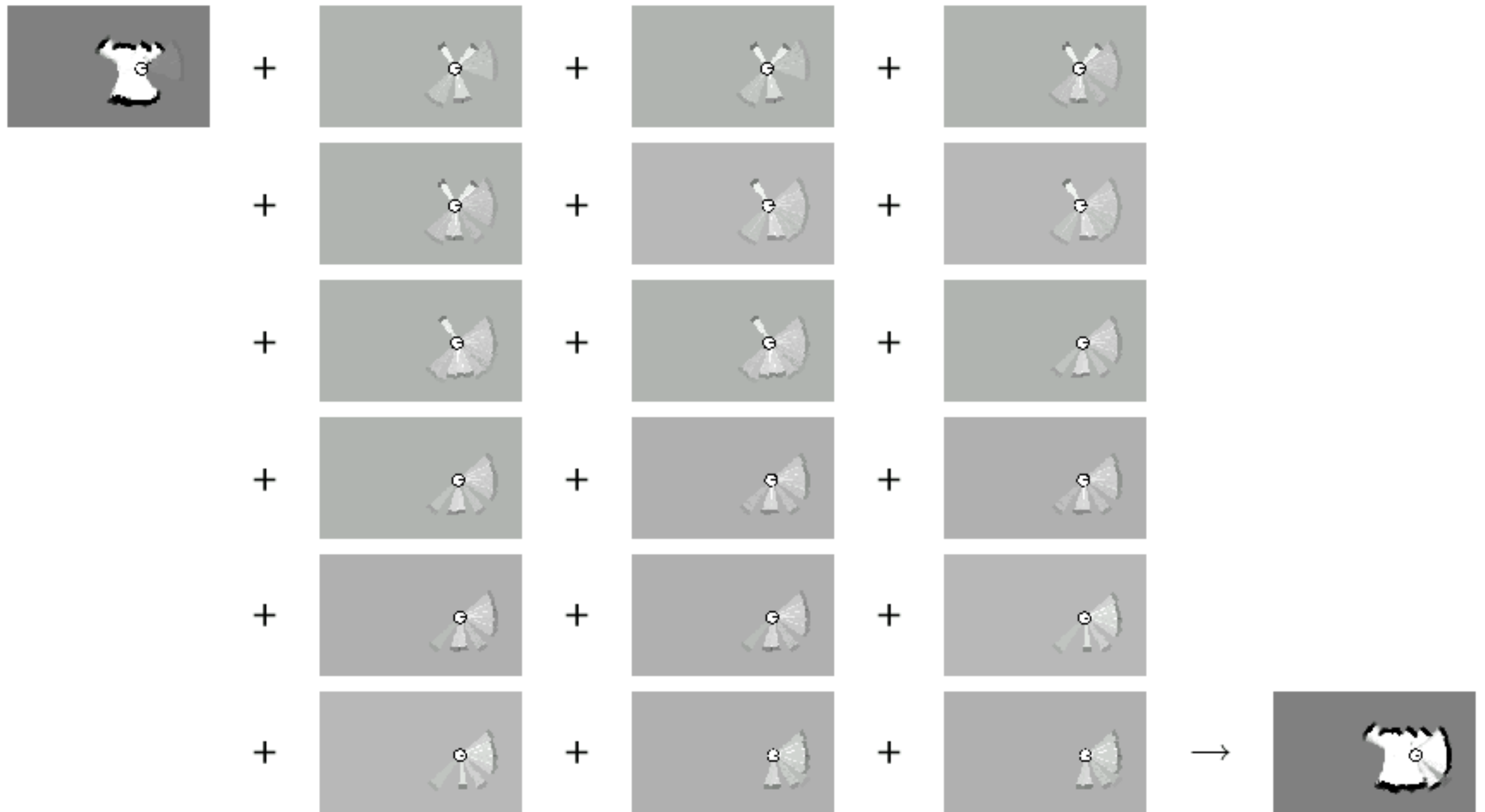
Combination of a linear function and a Gaussian:



Laser Rangefinder Sensor Model



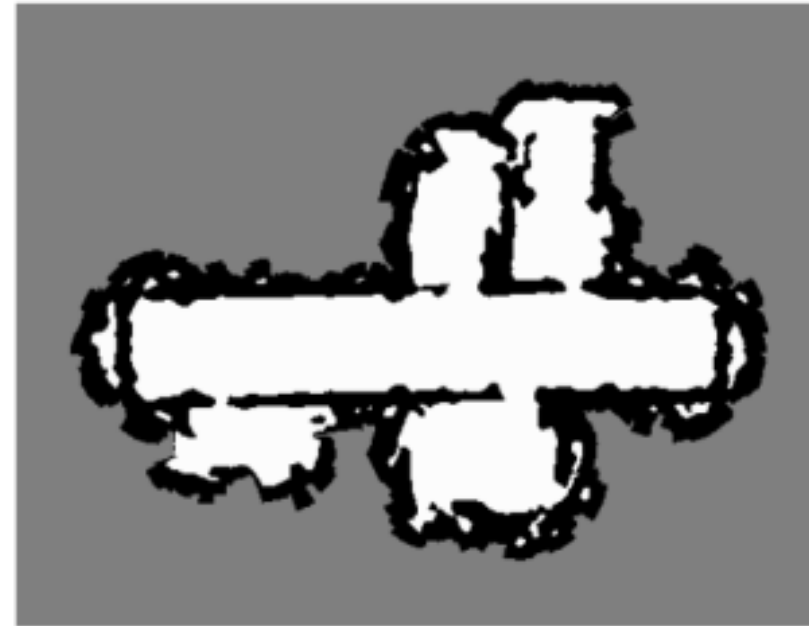
Incrementally Update the Map



Resulting Map

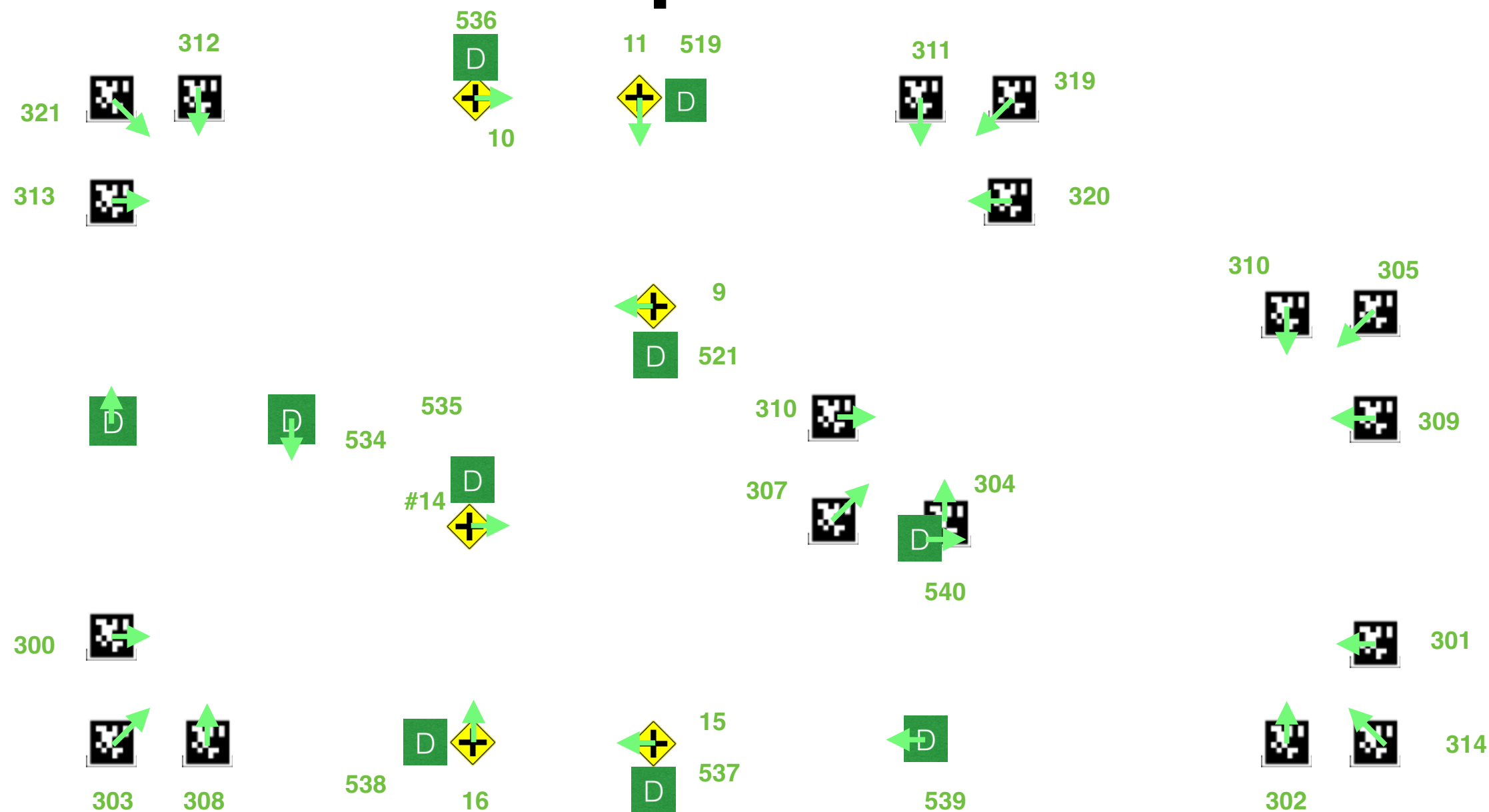


Maximum Likelihood Estimate

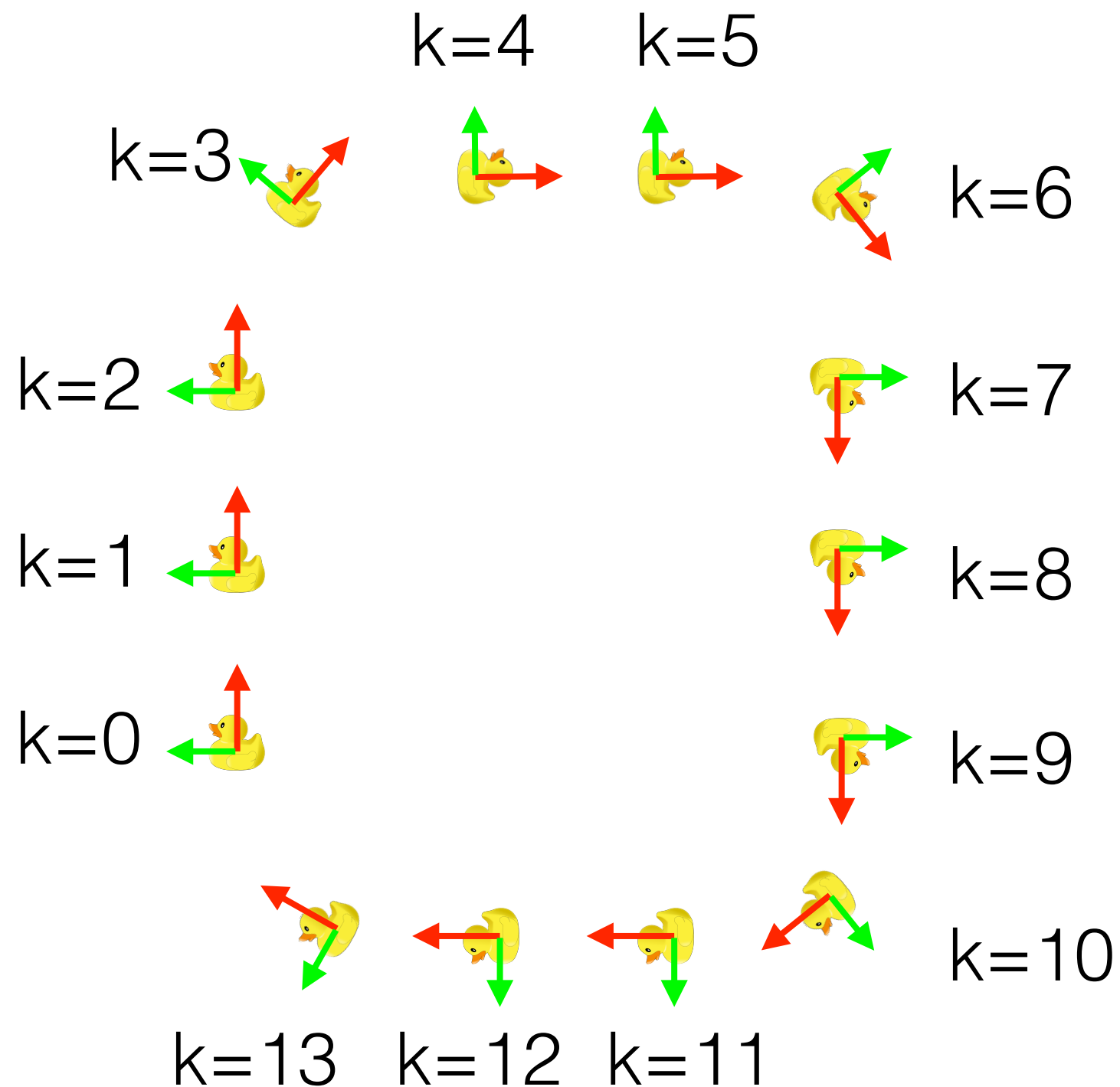


Clip the occupancy grid map at a threshold of 0.5

Feature-Based Maps



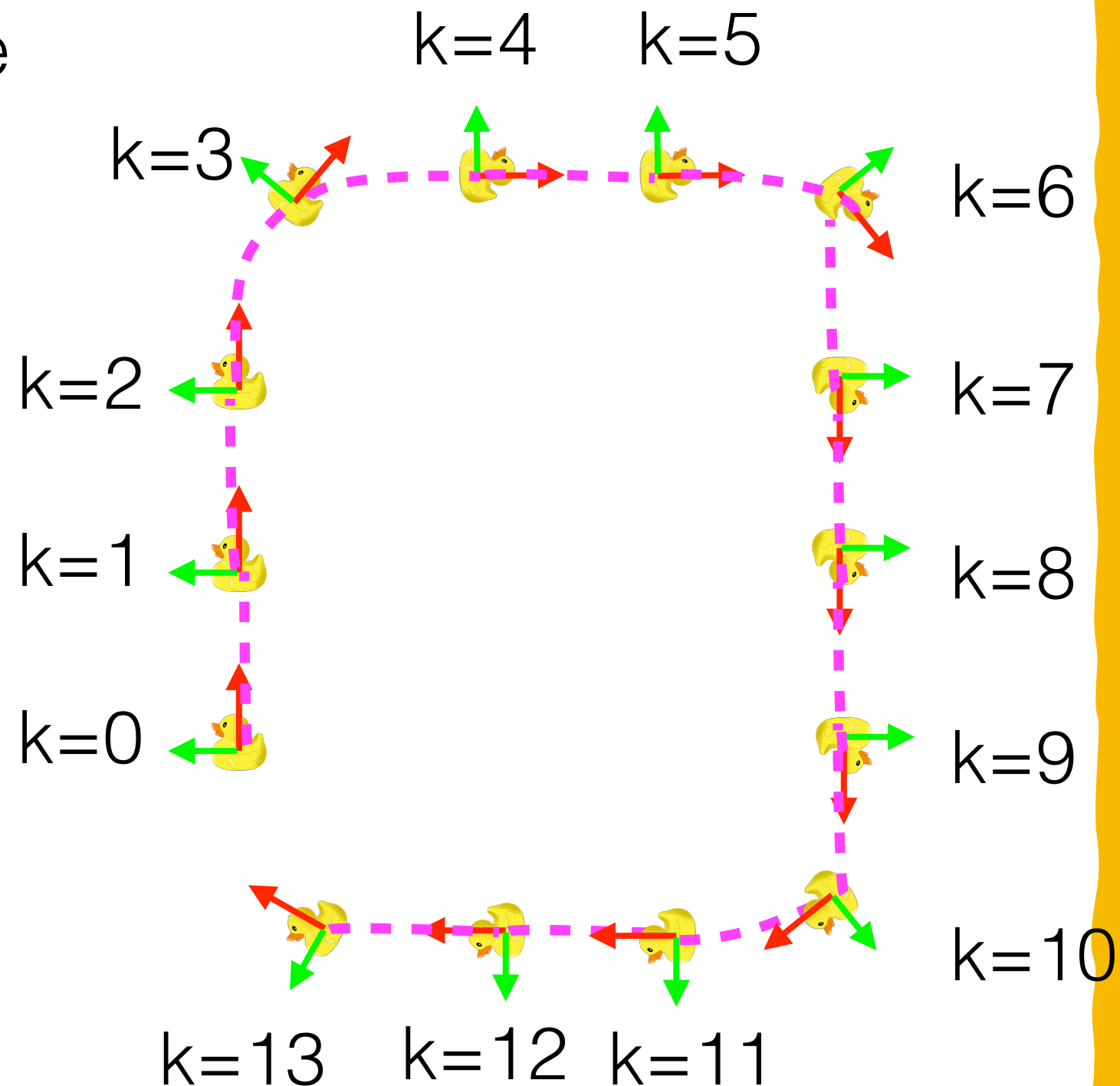
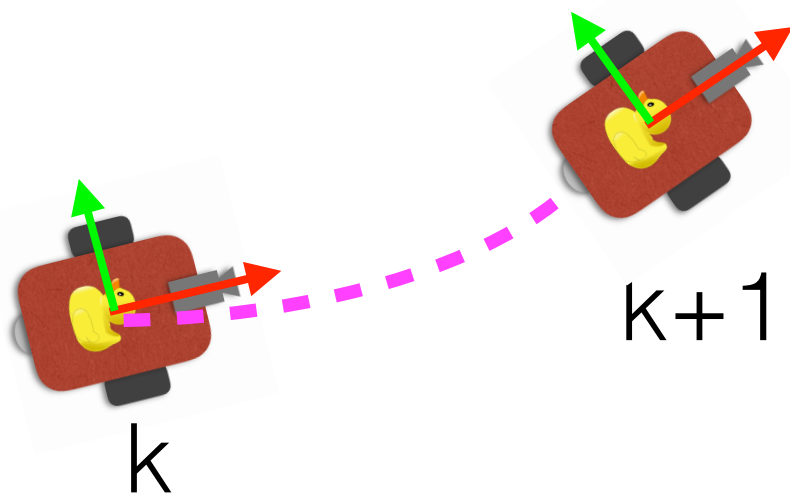
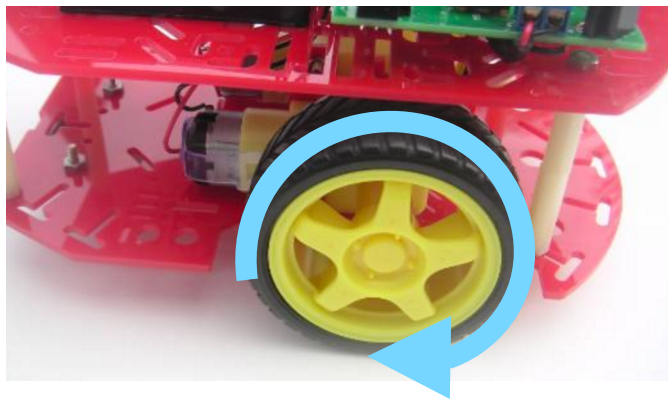
Pose Estimation (with no prior map)



Motion integration (odometry)

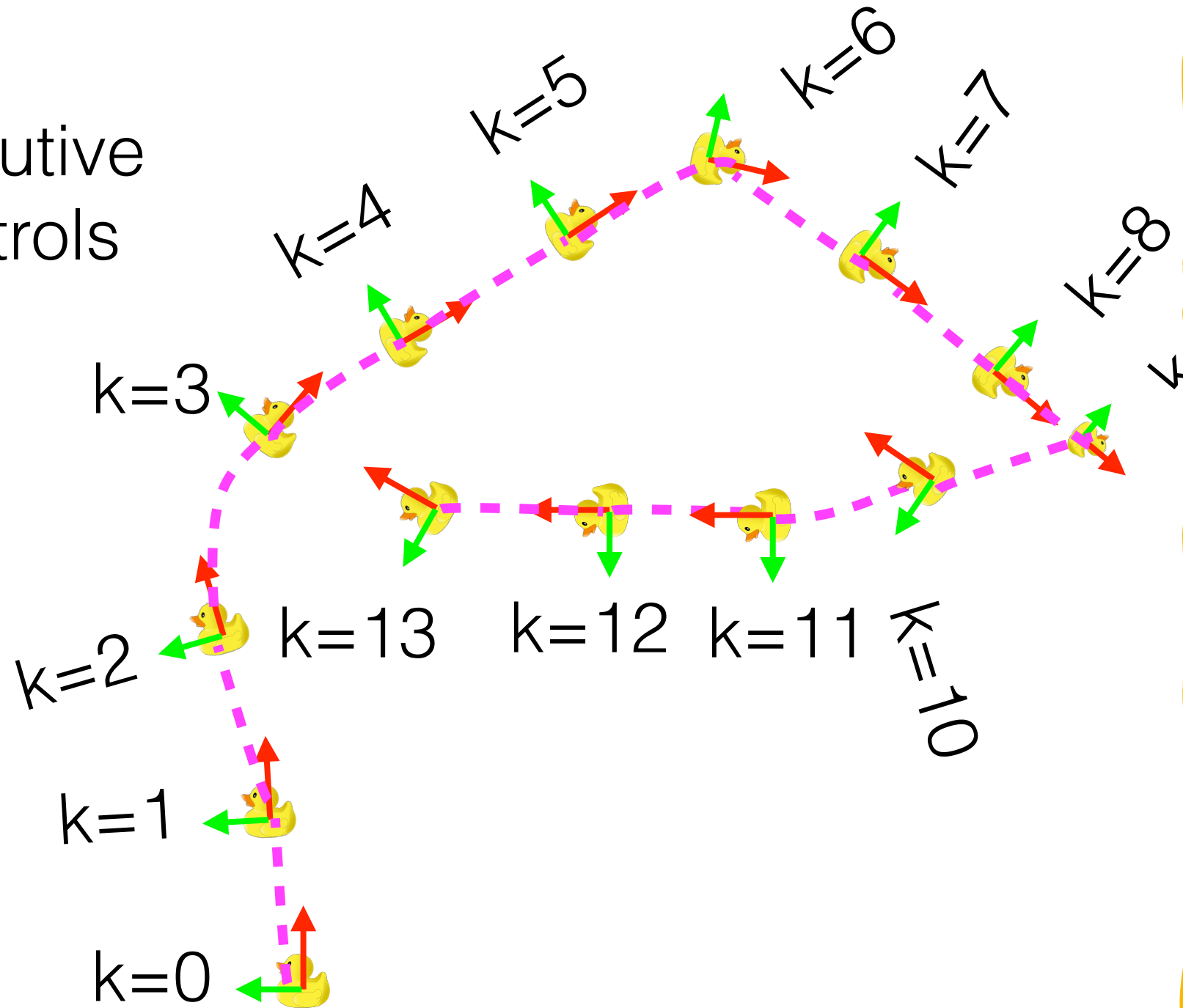
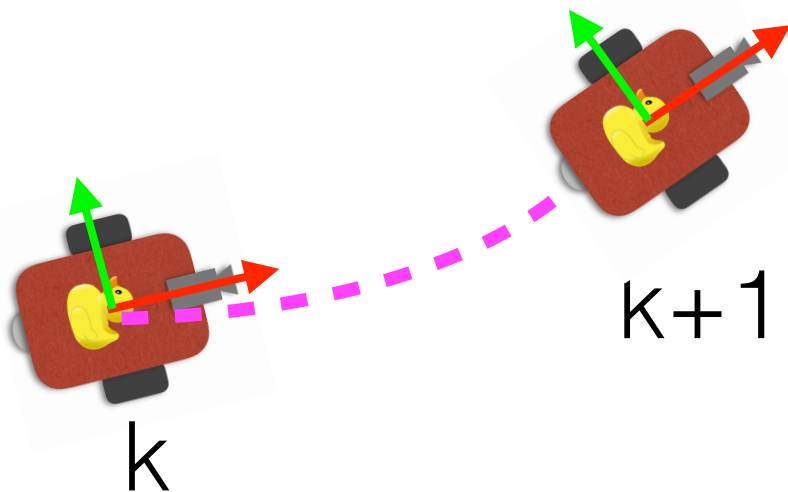
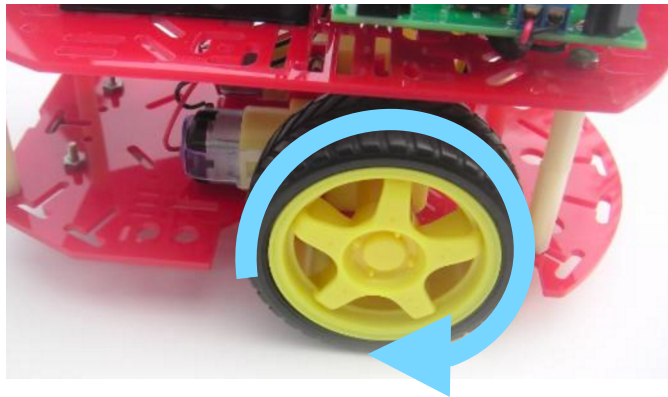
ideal (noiseless) world

We can estimate motion between consecutive poses from motion controls



Motion integration (odometry)

We can estimate motion between consecutive poses from motion controls



real (noisy) world

Motion integration (odometry)

ScreenRecorder

Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point

Displays

- Global Options
 - Fixed Frame: odom
 - Background Color: 48; 48; 48
 - Frame Rate: 30
- Global Status: Ok
 - Fixed Frame: OK
- Grid
 - Status: OK
- Marker
 - Status: OK
 - Marker Topic: /ferrari/gtTrajectory
 - Queue Size: 100
- Marker
 - Status: OK
 - Marker Topic: /ferrari/odomTrajectory
 - Queue Size: 100
- Image
 - Status: OK

Add Remove Rename

Image

Time

ROS Time: 1453244346.33 ROS Elapsed: 14.70 Wall Time: 1460521339.20 Wall Elapsed: 14.70

Reset

ground truth

odometry from wheel cmd

SimpleScreenRecorder

Recording

Pause recording

Enable recording hotkey Enable sound notifications

Hotkey: Ctrl + Shift + Alt + Super + R

Information

Total time: 0:00:00

FPS in: 0.00

FPS out: 0.00

Size in: 1600x900

Size out: ?

File name: rec2.mkv

File size: 0 B

Preview

Preview frame rate: 10

Note: Previewing requires extra CPU time (especially at high frame rates).

Save Remove

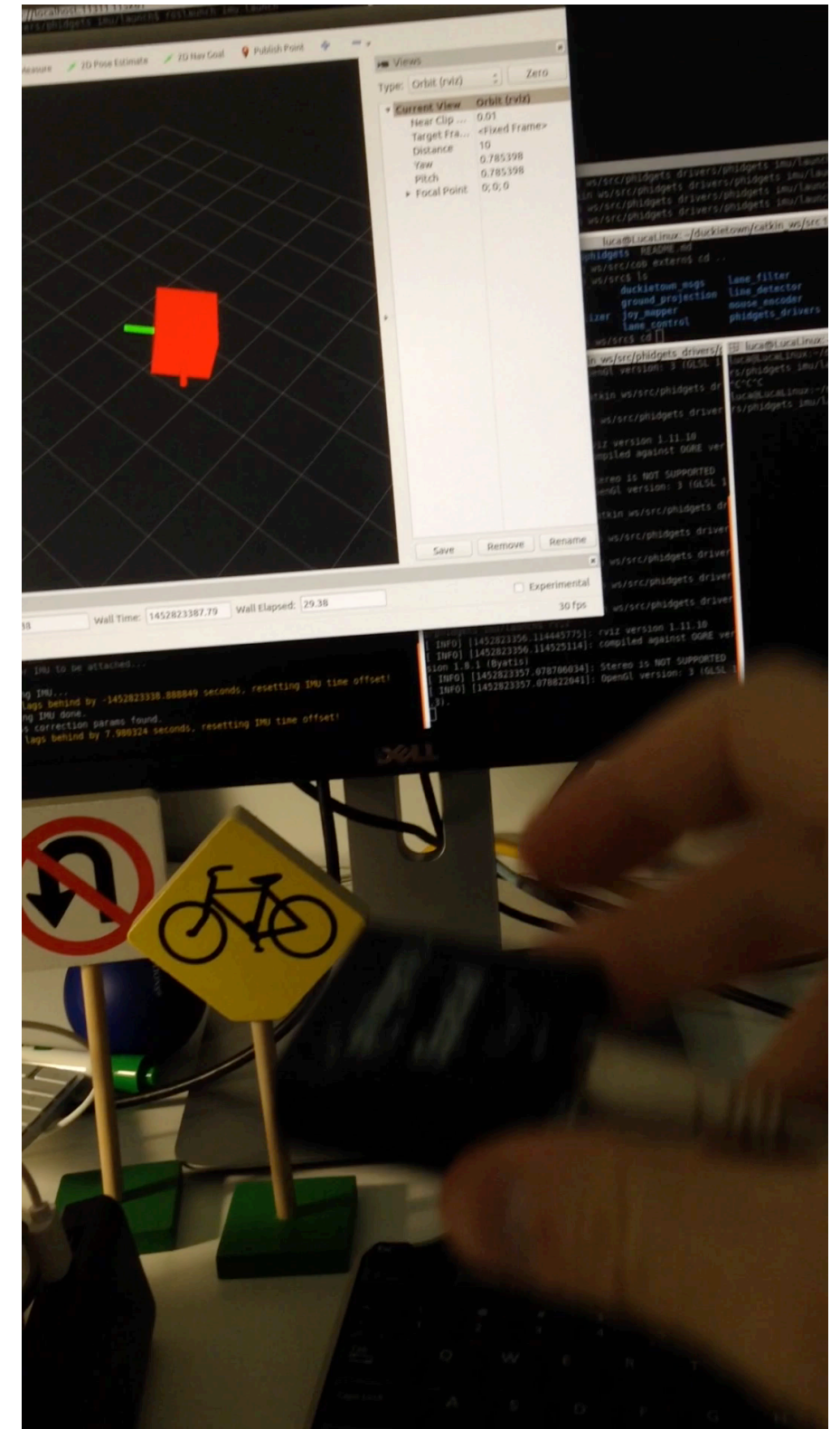
Motion integration (odometry)

what about using
better sensors?



Inertial Measurement Unit (IMU):

- accelerometer: accelerations
- gyroscope: angular velocities
- magnetometer: magnetic north direction



Motion integration (odometry)

The image shows a screenshot of a ROS environment with RViz and SimpleScreenRecorder. The RViz window displays a 2D grid map with a green dot representing the robot's current position. The SimpleScreenRecorder window is open, showing recording settings and a log of system messages.

Legend:

- ground truth (green line)
- odometry (wheel cmd + IMU) (blue line)

RViz Displays:

- Queue Size: 100
- Namespaces:
 - Marker:
 - Status: Ok
 - Marker Topic: /ferrari/odomTrajectory
 - Queue Size: 100
 - Image:
 - Status: Ok
 - Image Topic: /ferrari/apriltags/tags_im...
 - Transport Hint: compressed
 - Queue Size: 2
 - Marker:
 - Status: Ok
 - Marker Topic: /ferrari/slamTrajectory
 - Queue Size: 100
 - Marker:
 - Status: Ok

SimpleScreenRecorder Settings:

- Recording:
 - Pause recording
 - Enable recording hotkey: ☒ Ctrl + ☐ Shift + ☐ Alt + ☐ Super + R
 - Enable sound notifications: ☐
- Information:
 - Total time: 0:00:00
 - FPS in: 0.00
 - FPS out: 0.00
 - Size in: 1600x900
 - Size out: ?
 - File name: rec3.mkv
 - File size: 0 B
 - Bit rate: 0 bps
- Preview:
 - Preview frame rate: 10
 - Note: Previewing requires extra CPU time (especially at high frame rates).
- Log:
 - [X11Input::Init] Using X11 shared memory.
 - [X11Input::InputThread] Input thread started.
 - [PageRecord::StartInput] Started input.
 - [PulseAudioInput::InputThread] Input thread started.

System Information:

- ROS Time: 1453244341.03
- ROS Elapsed: 9.71
- Wall Time: 1460522285.55
- Wall Elapsed: 9.68

Motion integration (odometry)

DEAD RECKONING AT SEA



A navigator can take measurements of the ship speed and direction and the effects of wind and current. He can estimate the ship's location fairly accurately - at least over short distances.

Blindfolded people in the desert are not able to walk on a straight line

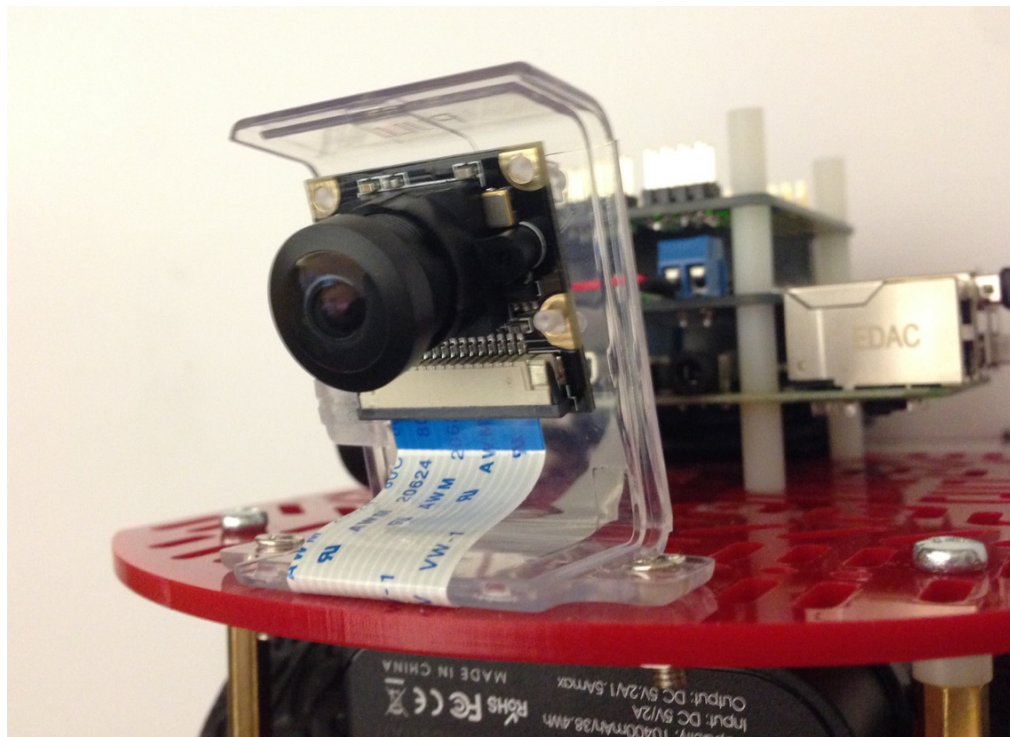


Souman et al. 2008

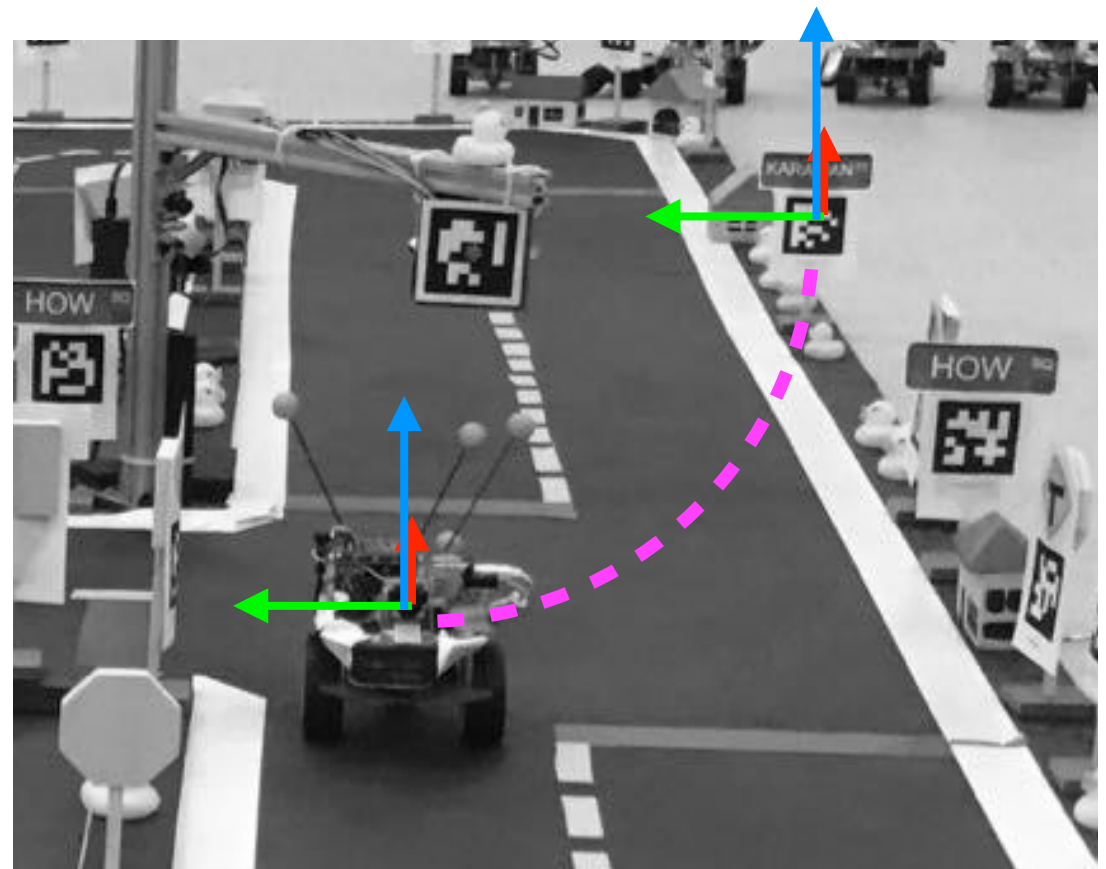
<http://news.nationalgeographic.com/news/2009/08/090820-walk-in-circles-video-ap.html>

Pose Estimation + Landmarks

- motion estimation (odometry) drifts over time, for any (real) proprioceptive sensor
- **Idea:** use external landmarks in the environment to reduce/avoid drift (**need exteroceptive sensors**)

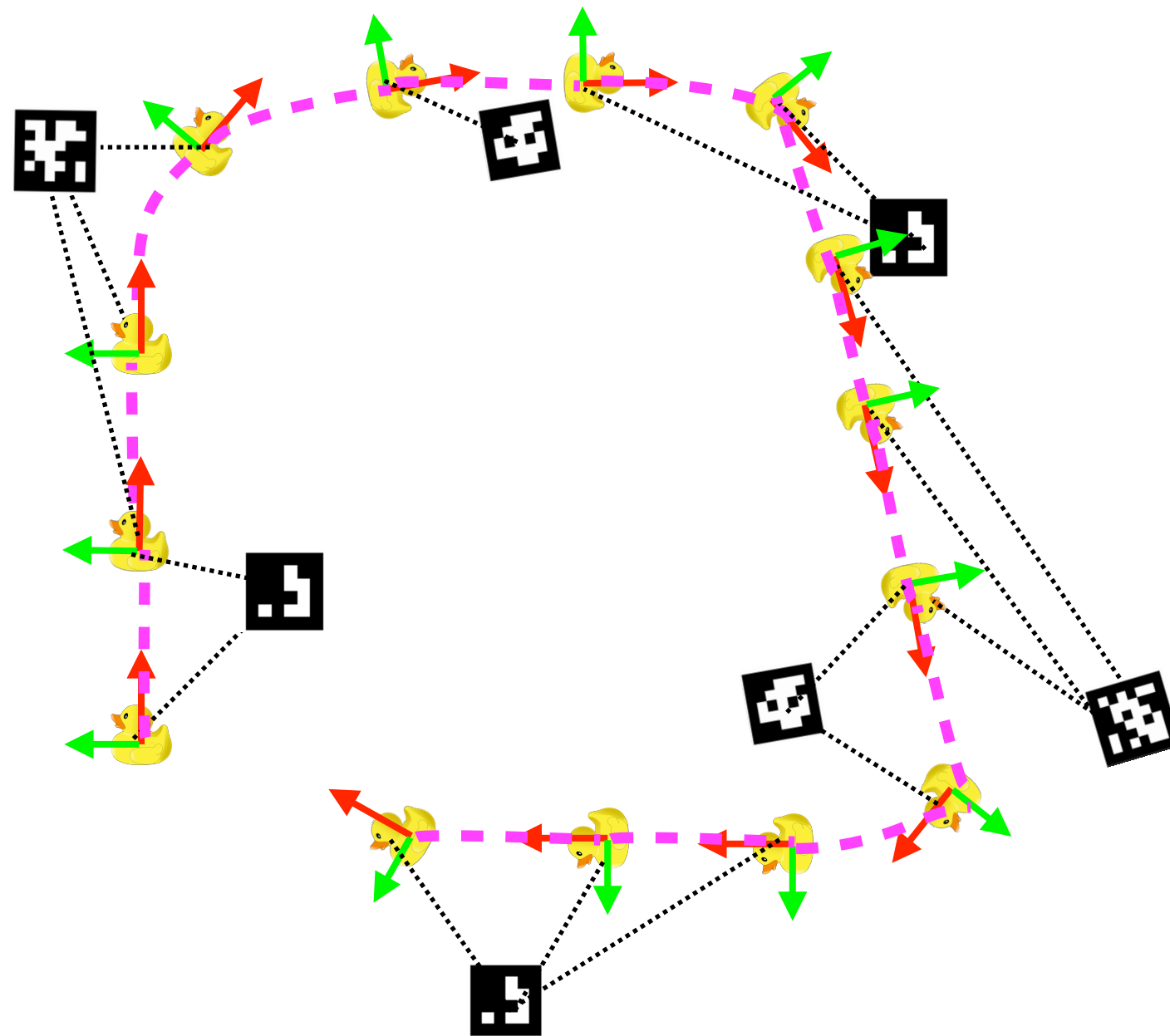


exteroceptive sensor:

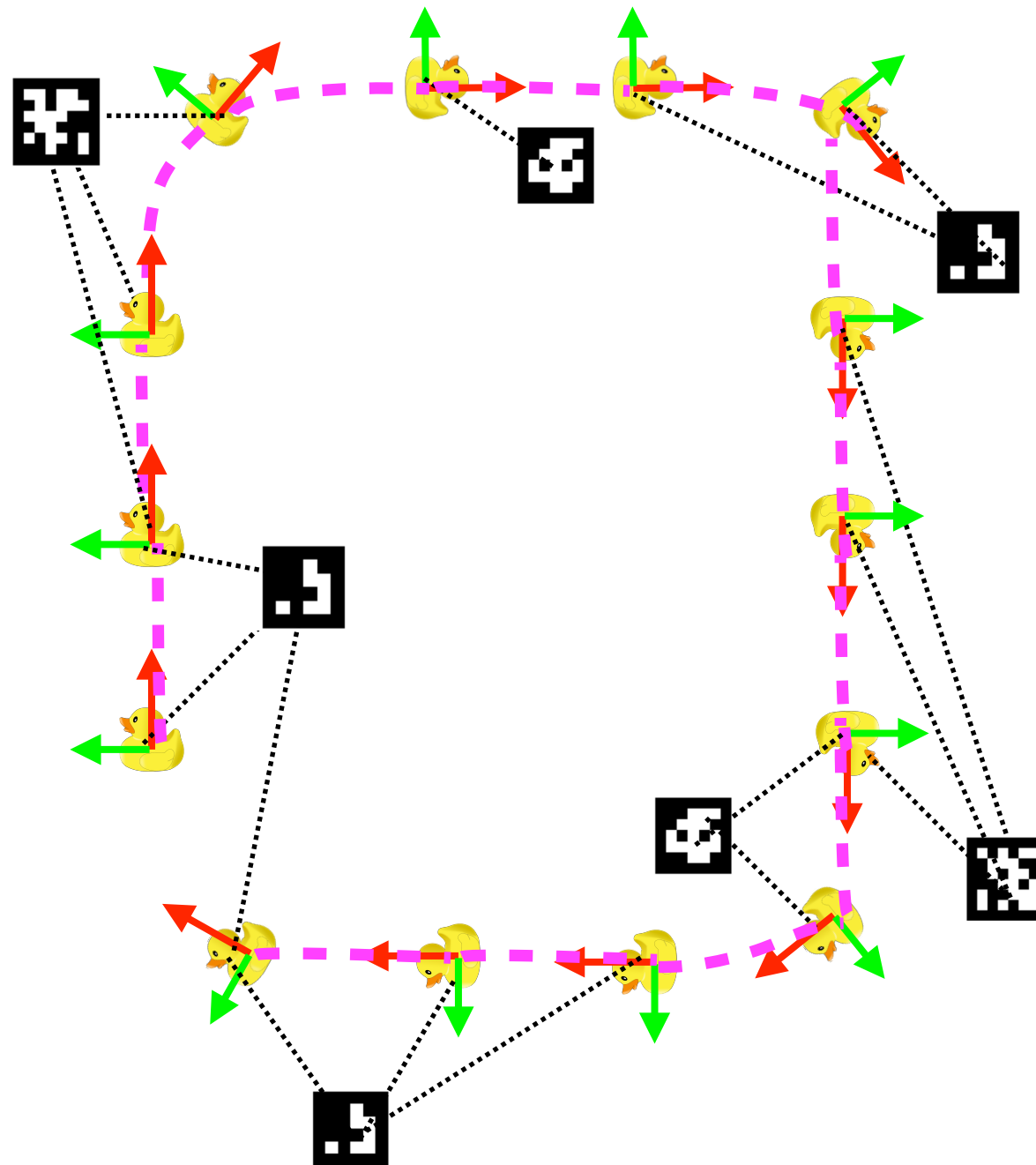


landmarks: april tags

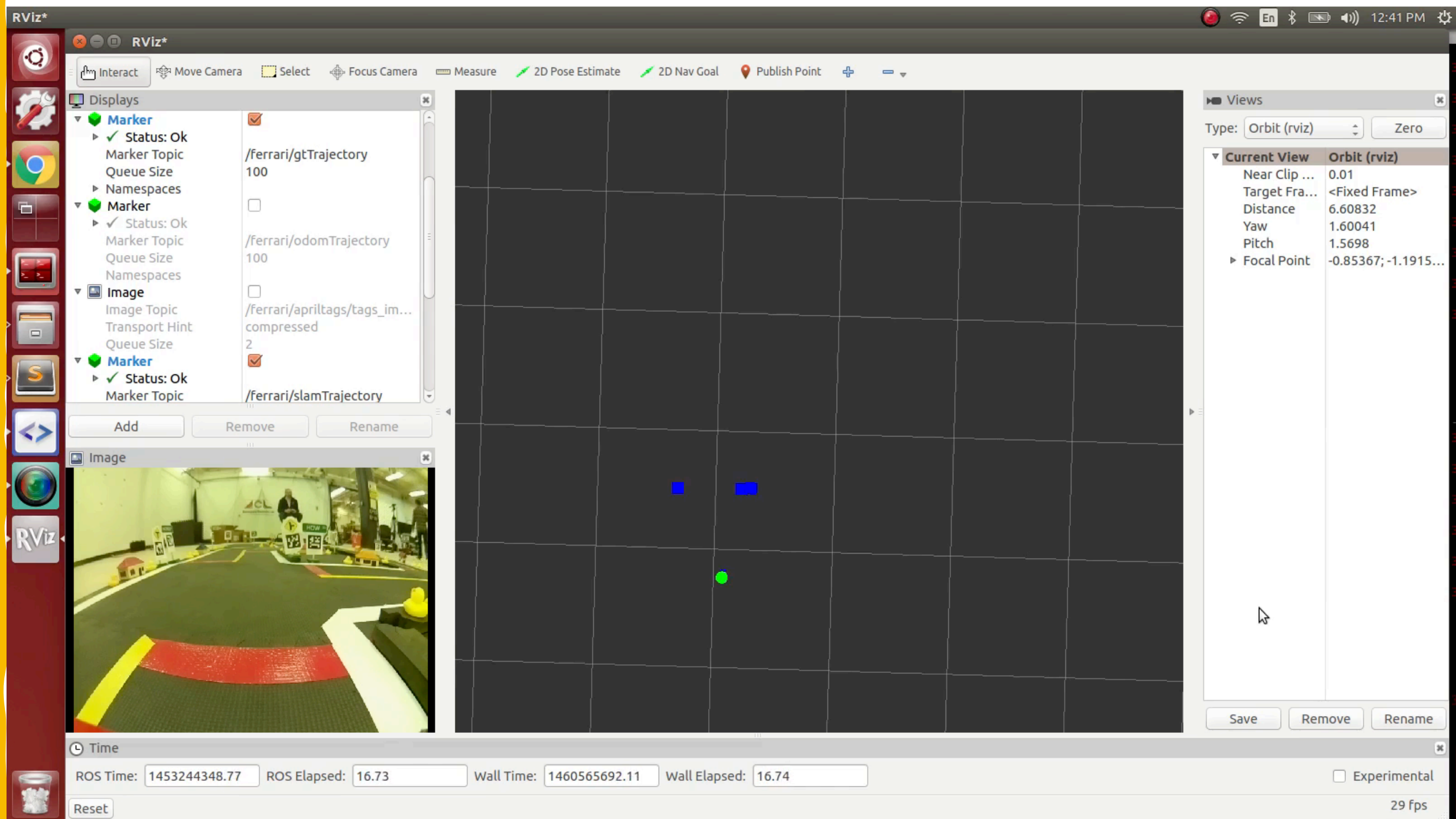
Combining odometry & landmarks: the SLAM problem



Combining odometry & landmarks: the SLAM problem



Feature-based SLAM in Duckietown



Simultaneous Localization and mapping

- Problem statement:
 - Jointly estimate the map and the robot pose within the map.
 - State is now comprised of robot poses and map representation
 - $x_t \triangleq (p_t, m)$
 - feature-based representation: $m = \{l_j\}$

SLAM: a bit of history

Randall C. Smith*

SRI International
Menlo Park, California 94025

Peter Cheeseman

NASA Ames
Moffett Field, California 94025

On the Representation and Estimation of Spatial Uncertainty

Mobile Robot Localization by Tracking Geometric Beacons

John J. Leonard and Hugh F. Durrant-Whyte

Abstract—This short paper presents the application of the extended Kalman filter (EKF) to the problem of mobile robot navigation in a known environment. We have developed an algorithm for model-based localization that relies on the concept of a *geometric beacon*—a natu-

EKF-SLAM

1990 1995 2000 2005 2010 2015

SLAM - a bit of history

Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks

Arnaud Doucet[‡]

Nando de Freitas[†]

Kevin Murphy[†]

Stuart Russell[†]

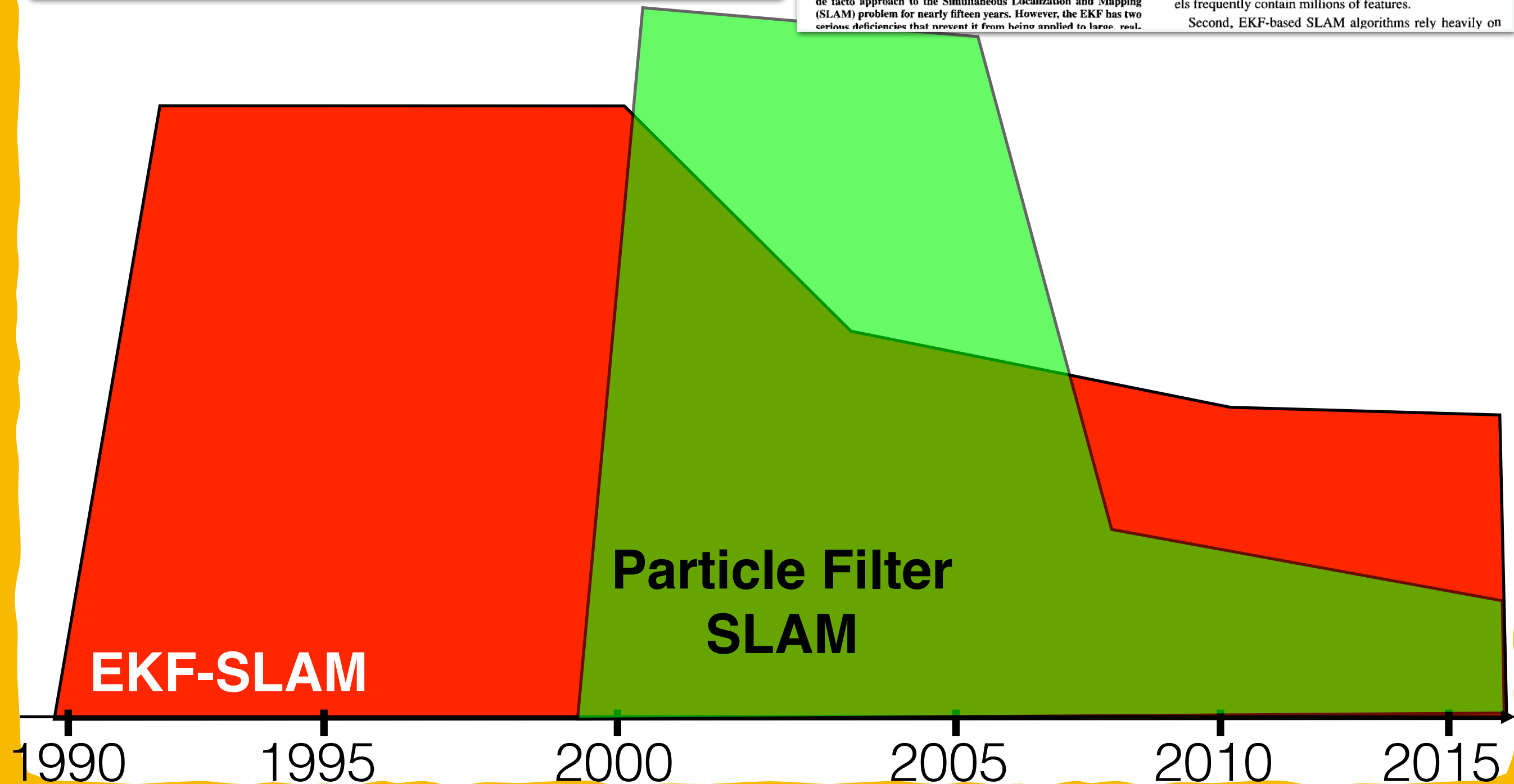
Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM

Michael Montemerlo, Sebastian Thrun

Abstract— The Extended Kalman Filter (EKF) has been the de facto approach to the Simultaneous Localization and Mapping (SLAM) problem for nearly fifteen years. However, the EKF has two serious deficiencies that prevent it from being applied to large, real-

to only a few hundred—whereas natural environment models frequently contain millions of features.

Second, EKF-based SLAM algorithms rely heavily on



EKF-SLAM

**Particle Filter
SLAM**

1990 1995 2000 2005 2010 2015

SLAM: a bit of history

Globally Consistent Range Scan Alignment for Environment Mapping

F. Lu*, E. Milios
Department of Computer Science,
York University,
North York, Ontario, Canada
{lufeng, eem}@cs.yorku.ca

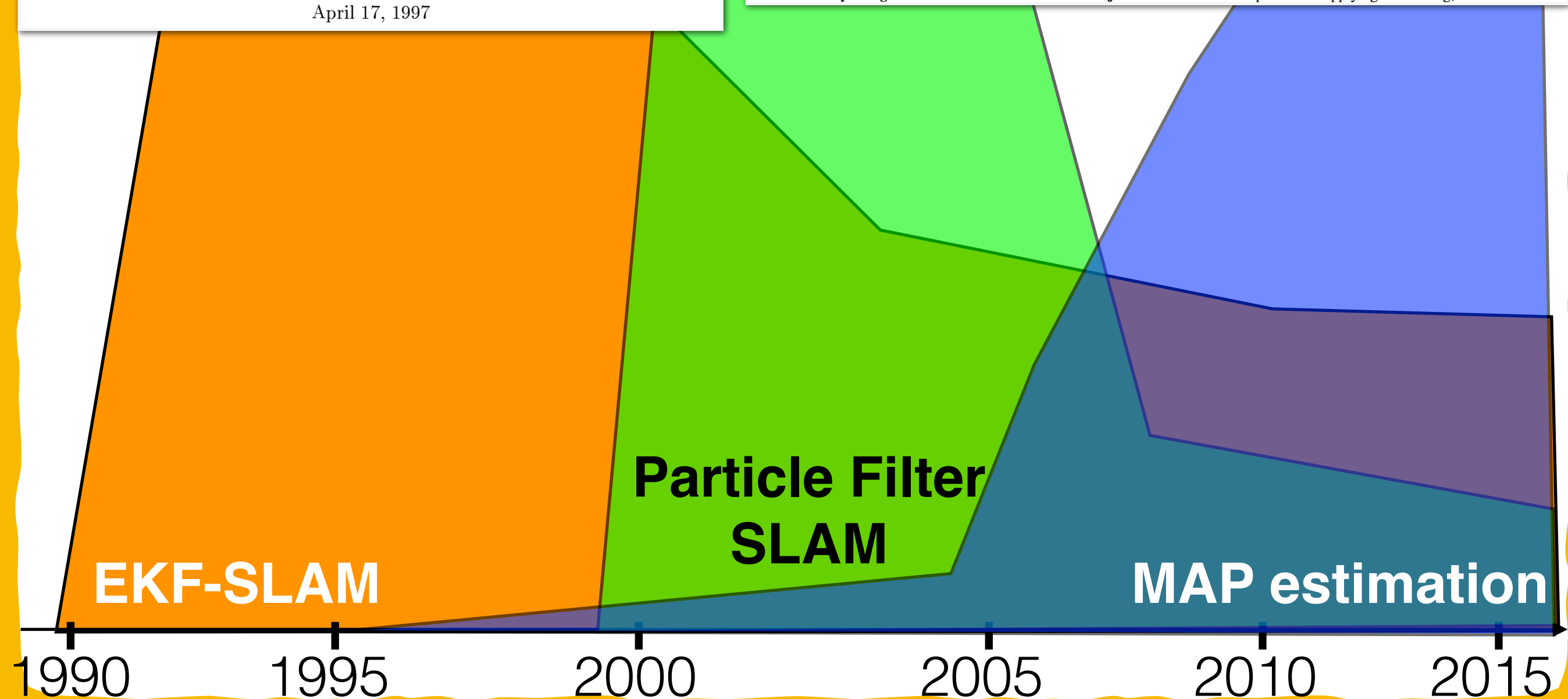
April 17, 1997

iSAM: Incremental Smoothing and Mapping

Michael Kaess, *Student Member, IEEE*, Ananth Ranganathan, *Student Member, IEEE*,
and Frank Dellaert, *Member, IEEE*

Abstract—We present incremental smoothing and mapping (iSAM), a novel approach to the simultaneous localization and mapping problem that is based on fast incremental matrix factorization. iSAM provides an efficient and exact solution by updating a QR factorization of the naturally sparse smoothing information matrix, therefore recalculating only the matrix entries that actually change. iSAM is efficient even for robot trajectories

counterintuitive at first, because more variables are added to the estimation problem, the simplification arises from the fact that the smoothing information matrix is naturally sparse. In contrast, in filtering approaches the information matrix becomes dense when marginalizing out robot poses. As a consequence of applying smoothing, we are able to



EKF-SLAM

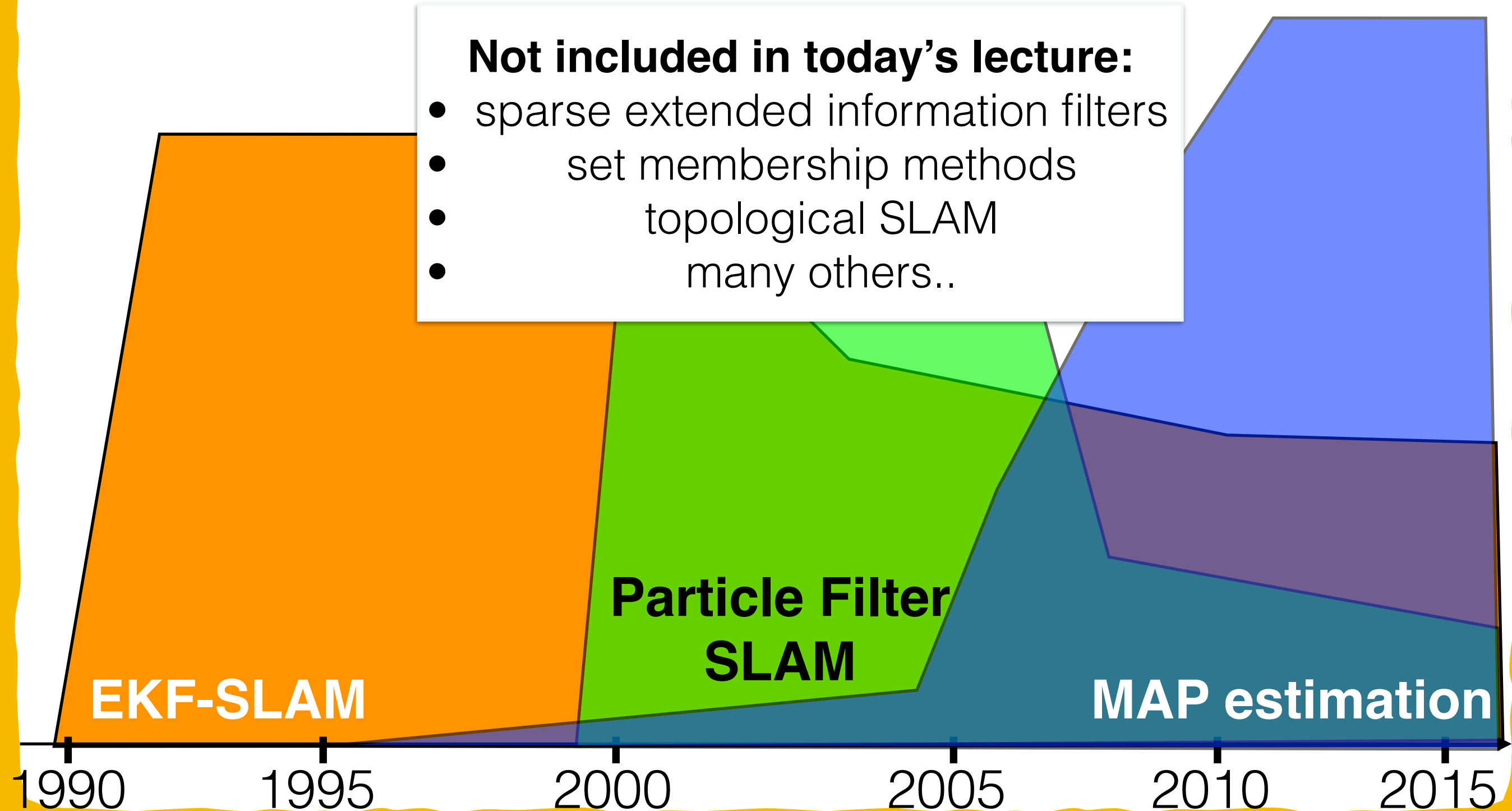
Particle Filter
SLAM

MAP estimation

SLAM: a bit of history

Not included in today's lecture:

- sparse extended information filters
- set membership methods
- topological SLAM
- many others..



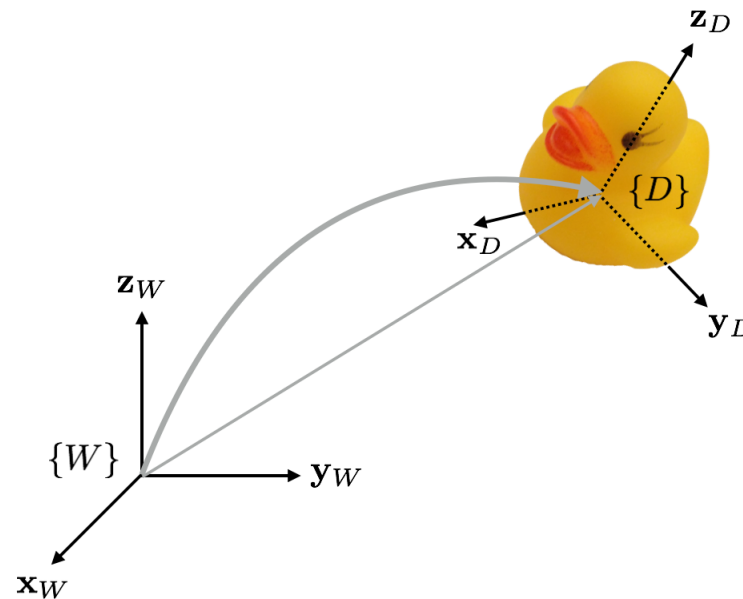
EKF-SLAM

**Particle Filter
SLAM**

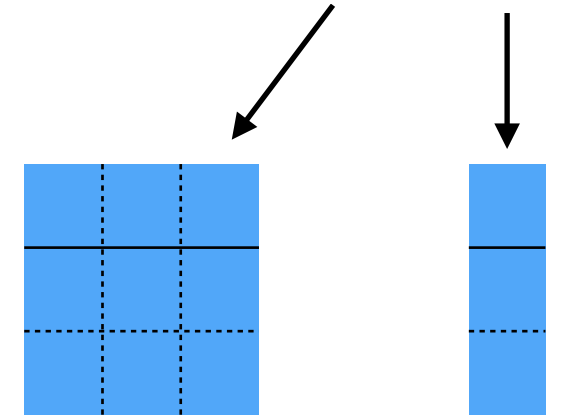
MAP estimation

Recall: the Special Euclidean Group

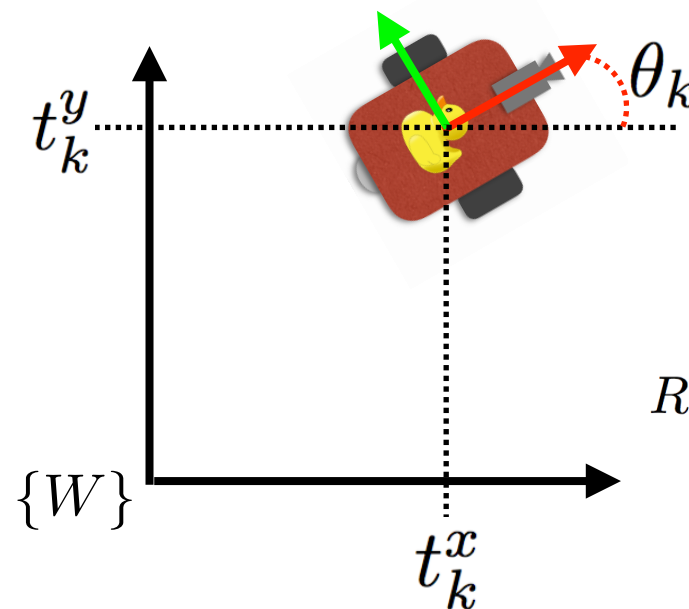
SE(3): 3D poses,
rigid motion in 3D



$$p_k = (R_k, t_k)$$



SE(2): 2D pose,
rigid motion in 2D

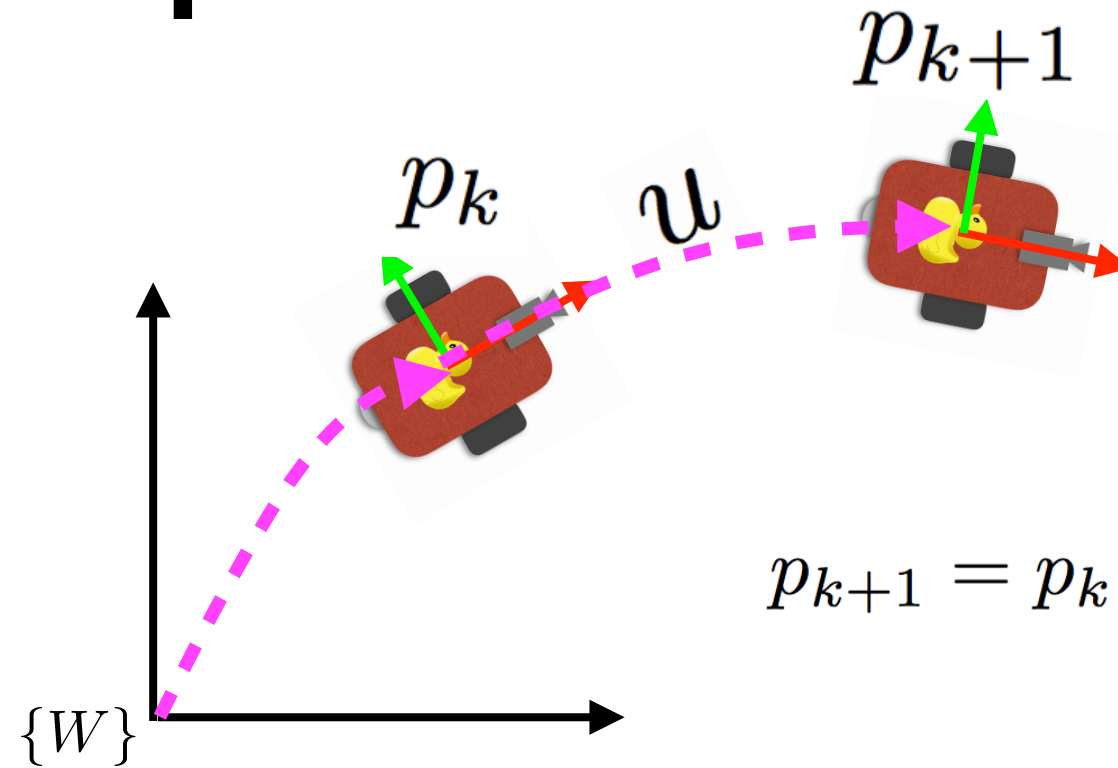


$$p_k = (R_k, t_k)$$

$$R_k = \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \quad t_k = \begin{bmatrix} t_k^x \\ t_k^y \end{bmatrix}$$

Recall: the Special Euclidean Group

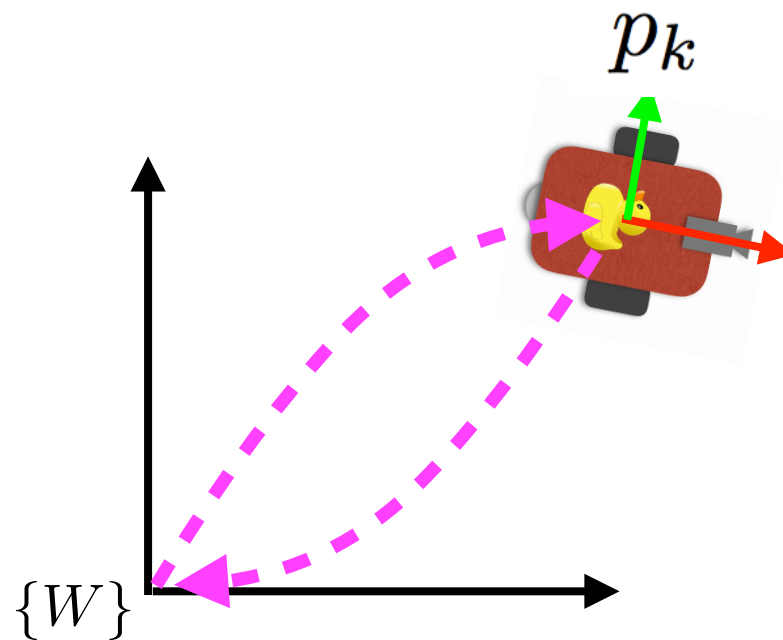
2D pose
composition



$$p_{k+1} = p_k \cdot u = \begin{bmatrix} t_k + R_k t_u \\ R_k R_u \end{bmatrix}$$

$\downarrow$
 $\theta_k + \theta_u$

2D pose
inverse



$$p_k^{-1} = \begin{bmatrix} -R_k^T t_k \\ R_k^T \end{bmatrix}$$

SLAM: inputs

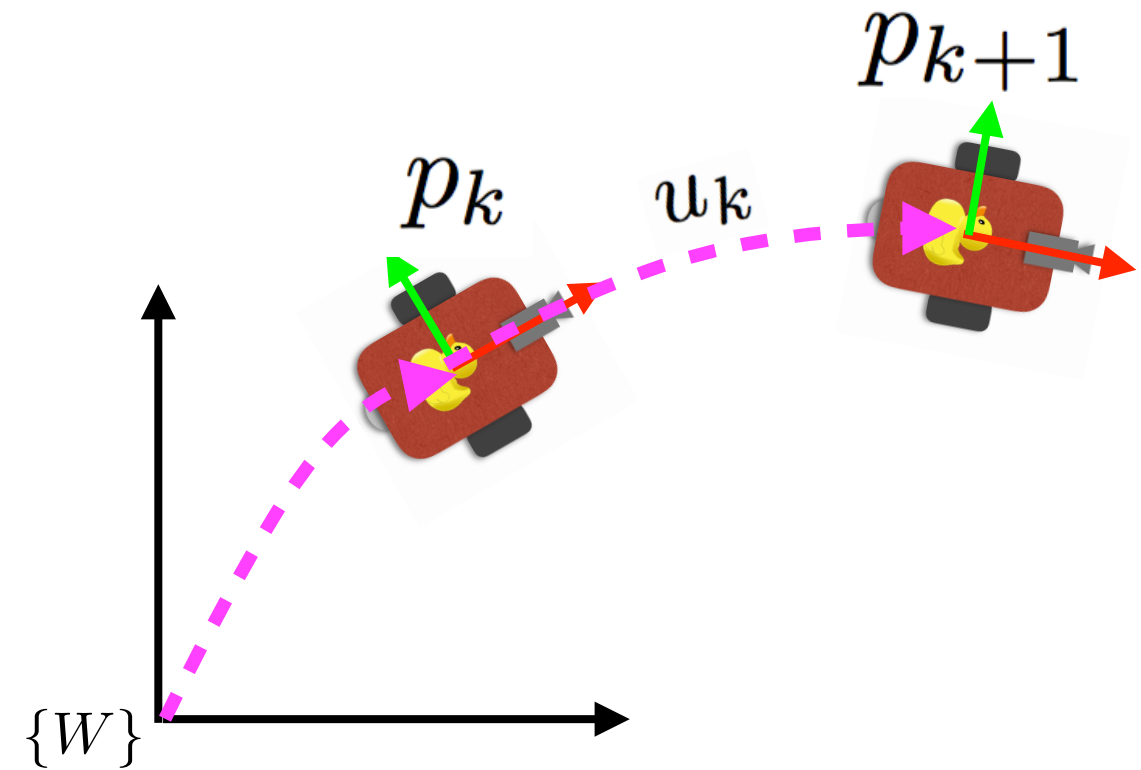
- **Input:** sensor measurements

1. **odometry:**

motion
noise

$$p_{k+1} = p_k \cdot u_k \cdot \epsilon_u$$

$$\hookrightarrow p_{k+1} = f(p_k, u_k, \epsilon_u)$$

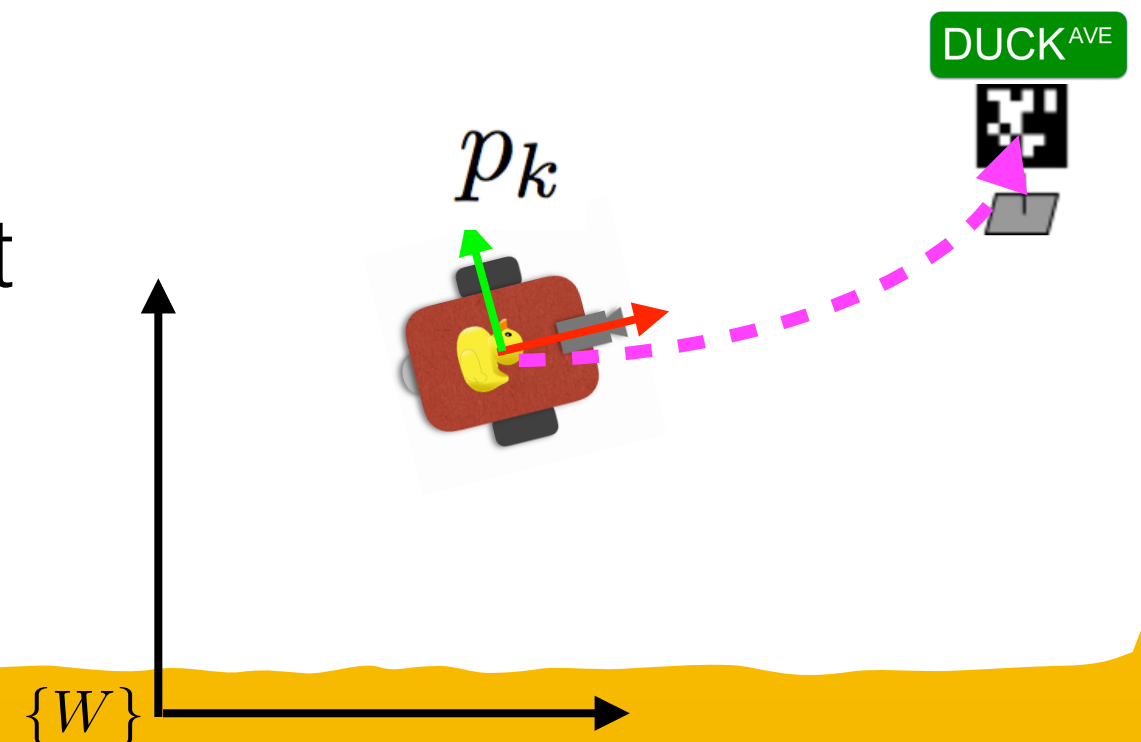


2. **landmark measurements:**

measurement
noise

$$z_{kj} = p_k^{-1} \cdot l_j \cdot \epsilon_z$$

$$\hookrightarrow z_{kj} = h(p_k, l_j, \epsilon_z)$$



SLAM: outputs

- **Output:** state estimate

1. Robot poses:

current pose: p_k

or

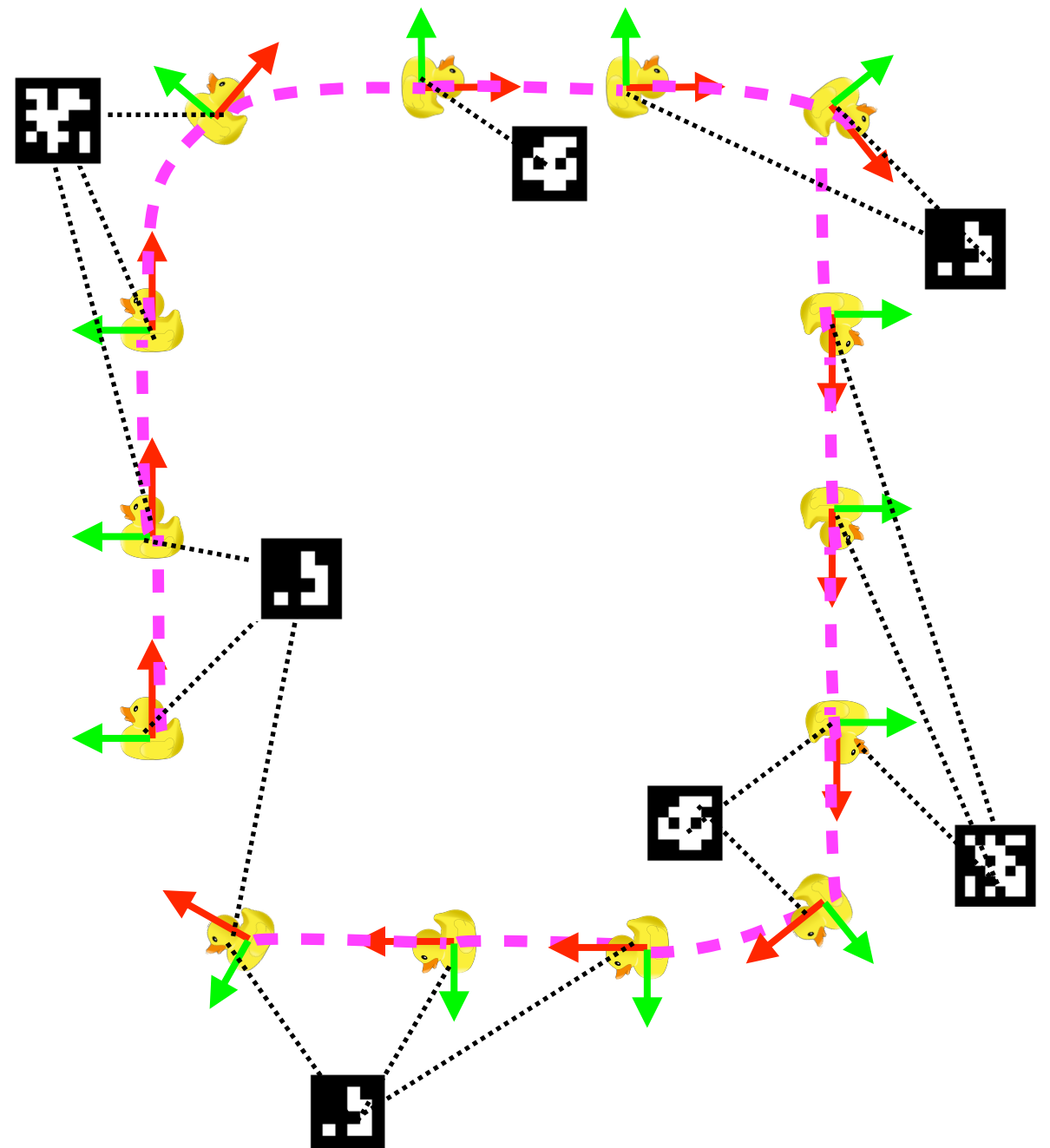
entire trajectory:

$$\{p_1, p_2, \dots, p_k\}$$

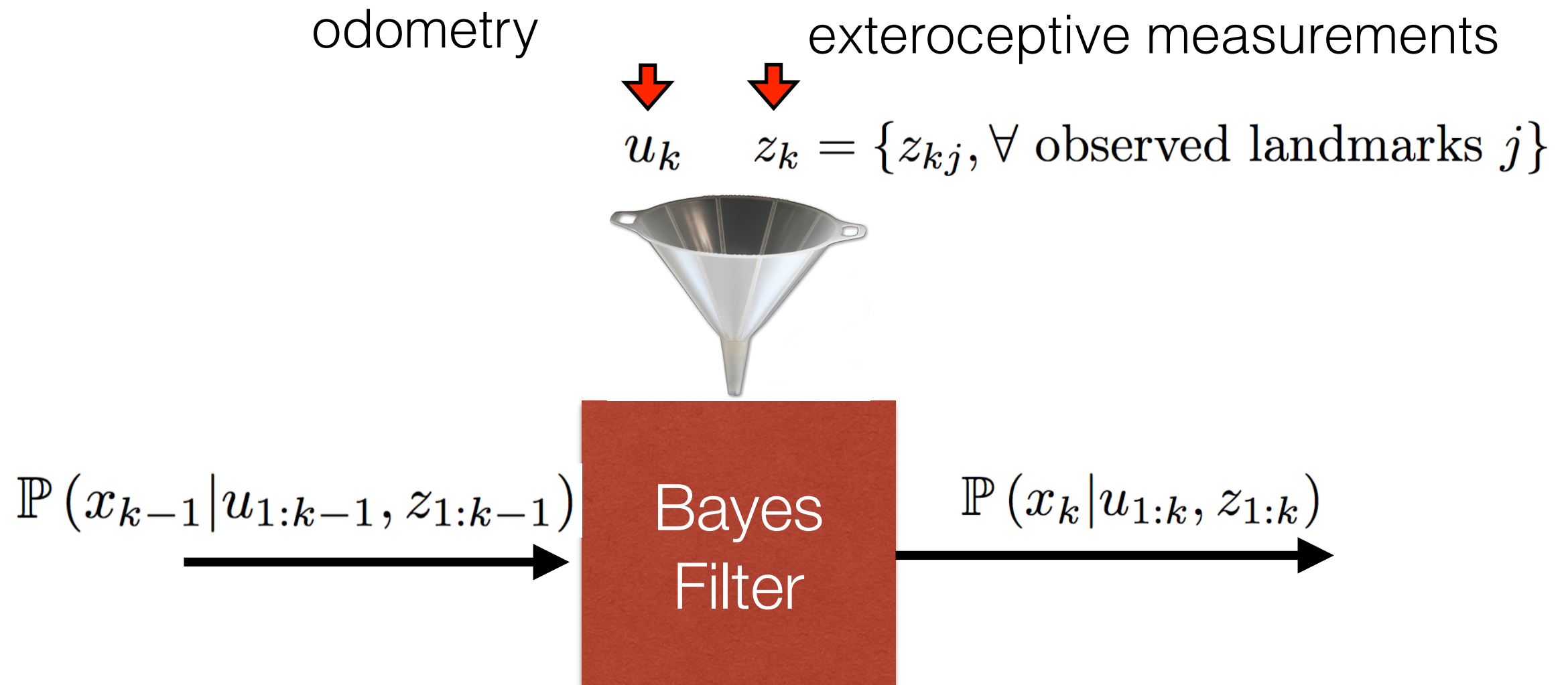
2. Landmark poses:

$$\{l_1, l_2, \dots, l_M\}$$

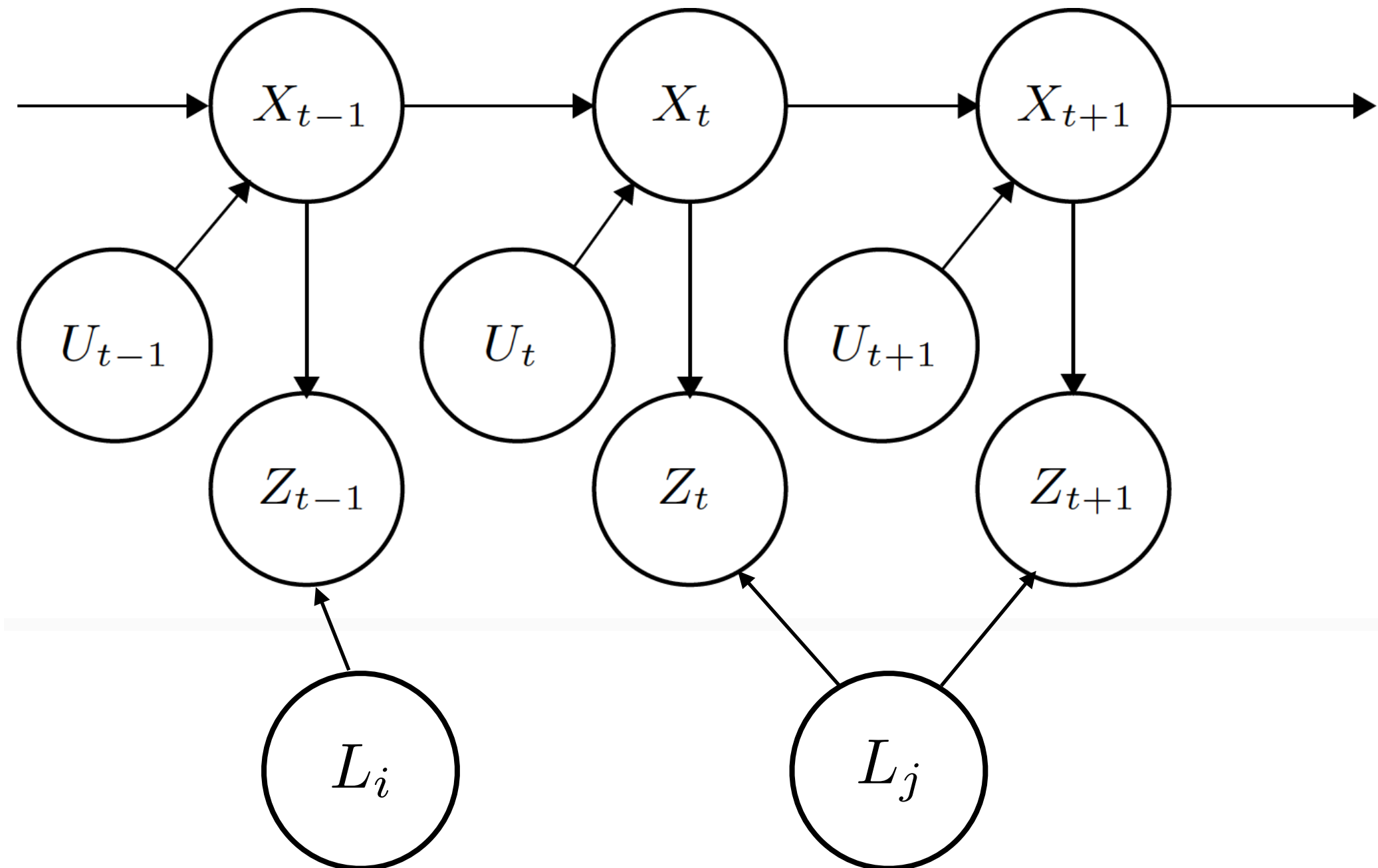
STATE: $x_k = \{p_k, l_1, l_2, \dots, l_M\}$



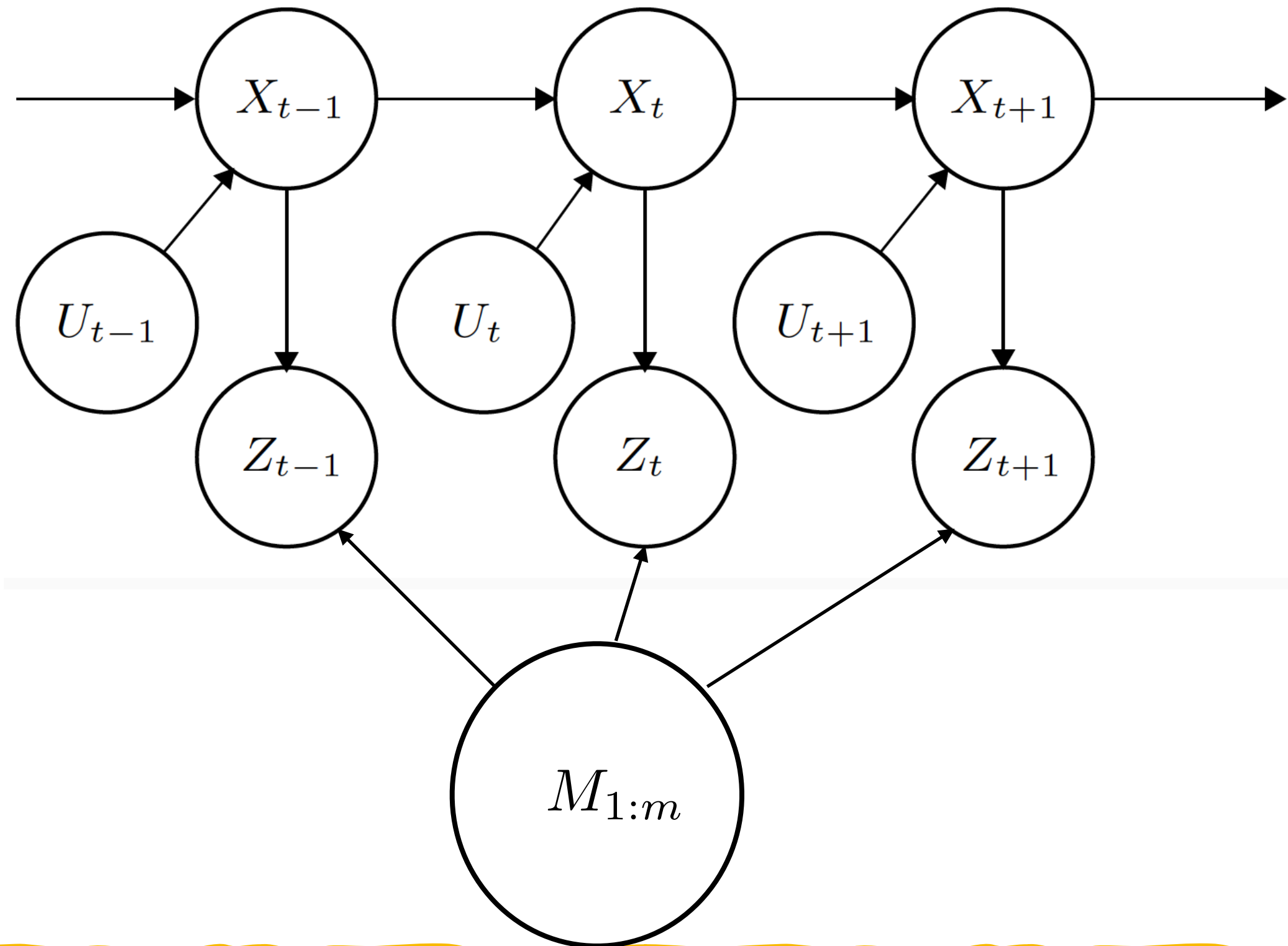
A General Solution: The Bayes Filter



Feature-Based SLAM Bayes' Net



View-based SLAM Bayes' Net



A General Solution: The Bayes Filter

Prediction (includes odometry in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k-1}) = \int \underbrace{\mathbb{P}(x_k | x_{k-1}, u_k)}_{\text{motion model}} \mathbb{P}(x_{k-1} | u_{1:k-1}, z_{1:k-1}) dx_{k-1}$$

$$x_{k+1} = f(x_k, u_k, \epsilon_u)$$

Update (includes other measurements in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k}) = \alpha \underbrace{\mathbb{P}(z_k | x_k)}_{\text{measurement model}} \mathbb{P}(x_k | u_{1:k}, z_{1:k-1})$$

$$z_k = h(x_k, \epsilon_z)$$



feasible for the lane filter (2 parameters to estimate)



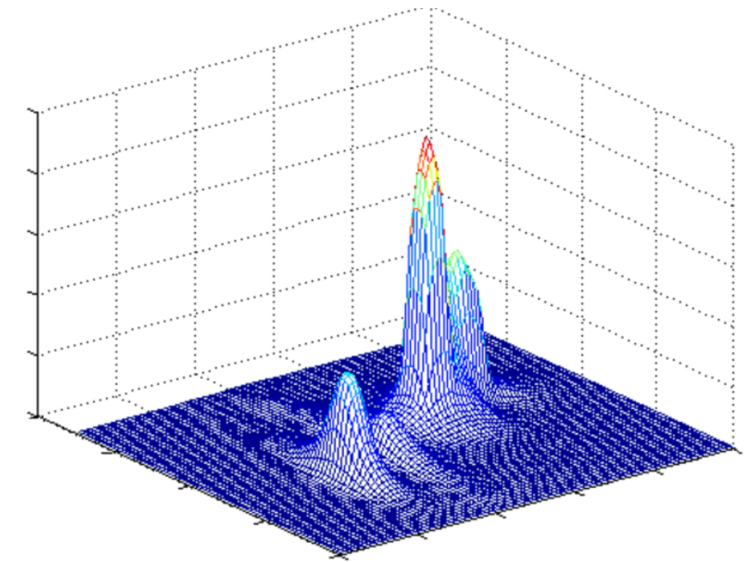
intractable in high-dimension (as in SLAM)

Kalman Filter



Idea: add assumptions and parametrize the pdf to make Bayes Filter tractable

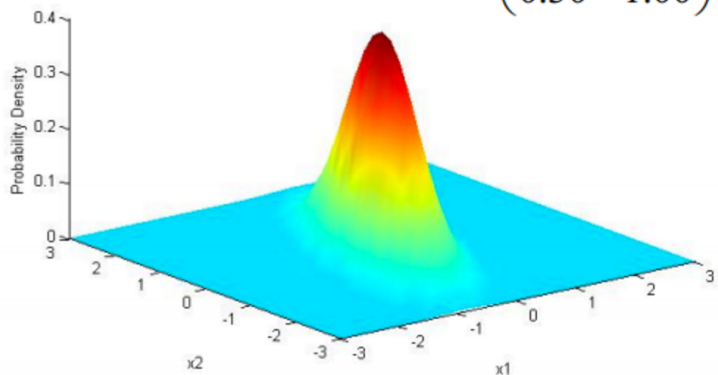
Generic pdf: $\mathbb{P}(x_k | u_{1:k}, z_{1:k})$



Gaussian pdf:

$$\mathcal{N}(\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(\Sigma_k)^{\frac{1}{2}}} \exp \left((x_k - \mu_k)^T \Sigma_k^{-1} (x_k - \mu_k) \right)$$

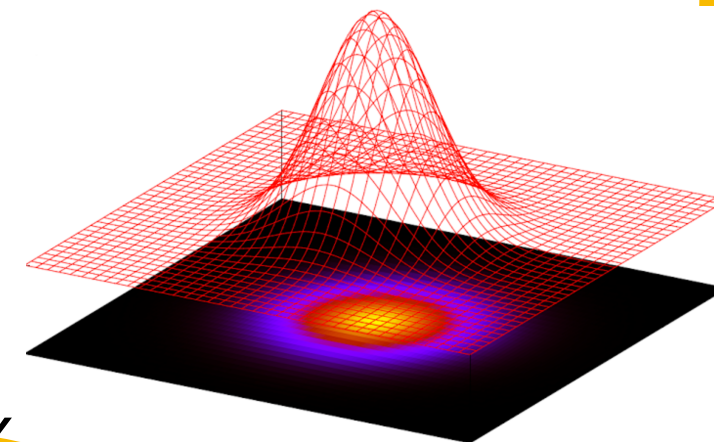
$$\mu = (0,0)^T \quad \Sigma = \begin{pmatrix} 0.25 & 0.30 \\ 0.30 & 1.00 \end{pmatrix}$$



μ_k mean

Σ_k covariance

Σ_k^{-1} information matrix



Kalman Filter



Idea: add assumptions and parametrize the pdf to make Bayes Filter tractable

$$\underline{x_{k+1} = f(x_k, u_k, \epsilon_u)}$$

motion model



$$x_{k+1} = Ax_k + Bu_k + B_u \epsilon_u$$

$$\underline{z_k = h(x_k, \epsilon_z)}$$

measurement model



$$z_k = Cx_k + D_z \epsilon_z$$

Linear Gaussian Model: Gaussian pdf + linear model

Kalman Filter

In the linear gaussian model,
the Bayes Filter admits a closed-form solution

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + B_u \epsilon_u \\z_k &= Cx_k + D_z \epsilon_z\end{aligned}$$

Initial
state: $\mathcal{N}(\hat{x}_1, \Sigma_1)$

Motion
noise: $\epsilon_u \sim \mathcal{N}(0, Q)$

Measurement.
noise: $\epsilon_z \sim \mathcal{N}(0, R)$

Prediction (includes odometry in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k-1}) = \int \mathbb{P}(x_k | x_{k-1}, u_k) \mathbb{P}(x_{k-1} | u_{1:k-1}, z_{1:k-1}) dx_{k-1}$$

Update (includes other measurements in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k}) = \alpha \mathbb{P}(z_k | x_k) \mathbb{P}(x_k | u_{1:k}, z_{1:k-1})$$

Kalman Filter

In the linear gaussian model,
the Bayes Filter admits a closed-form solution

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + B_u \epsilon_u \\z_k &= Cx_k + D_z \epsilon_z\end{aligned}$$

Initial
state: $\mathcal{N}(\hat{x}_1, \Sigma_1)$

Motion
noise: $\epsilon_u \sim \mathcal{N}(0, Q)$

Measurement.
noise: $\epsilon_z \sim \mathcal{N}(0, R)$

Prediction (includes odometry in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k-1}) = \mathcal{N}(\hat{x}_k^-, \Sigma_k^-) \quad \rightarrow \quad \begin{aligned}\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ \Sigma_k^- &= A\Sigma_{k-1}A^\top + B_u Q B_u^\top\end{aligned}$$

Update (includes other measurements in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k}) = \alpha \mathbb{P}(z_k | x_k) \mathbb{P}(x_k | u_{1:k}, z_{1:k-1})$$

Kalman Filter

In the linear gaussian model,
the Bayes Filter admits a closed-form solution

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + B_u \epsilon_u \\ z_k &= Cx_k + D_z \epsilon_z\end{aligned}$$

Initial
state: $\mathcal{N}(\hat{x}_1, \Sigma_1)$

Motion
noise: $\epsilon_u \sim \mathcal{N}(0, Q)$

Measurement.
noise: $\epsilon_z \sim \mathcal{N}(0, R)$

Prediction (includes odometry in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k-1}) = \mathcal{N}(\hat{x}_k^-, \Sigma_k^-) \quad \Rightarrow \quad \begin{aligned}\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ \Sigma_k^- &= A\Sigma_{k-1}A^\top + B_u Q B_u^\top\end{aligned}$$

Update (includes other measurements in belief)

$$\mathbb{P}(x_k | u_{1:k}, z_{1:k}) = \mathcal{N}(\hat{x}_k, \Sigma_k) \quad \Rightarrow \quad \begin{aligned}K &= \Sigma_k^- C^\top (C \Sigma_k^- C^\top + B_z R B_z^\top)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K (z_k - C \hat{x}_k^-) \\ \Sigma_k &= (I - KC) \Sigma_k^-\end{aligned}$$

but our system is nonlinear...

Extended Kalman Filter

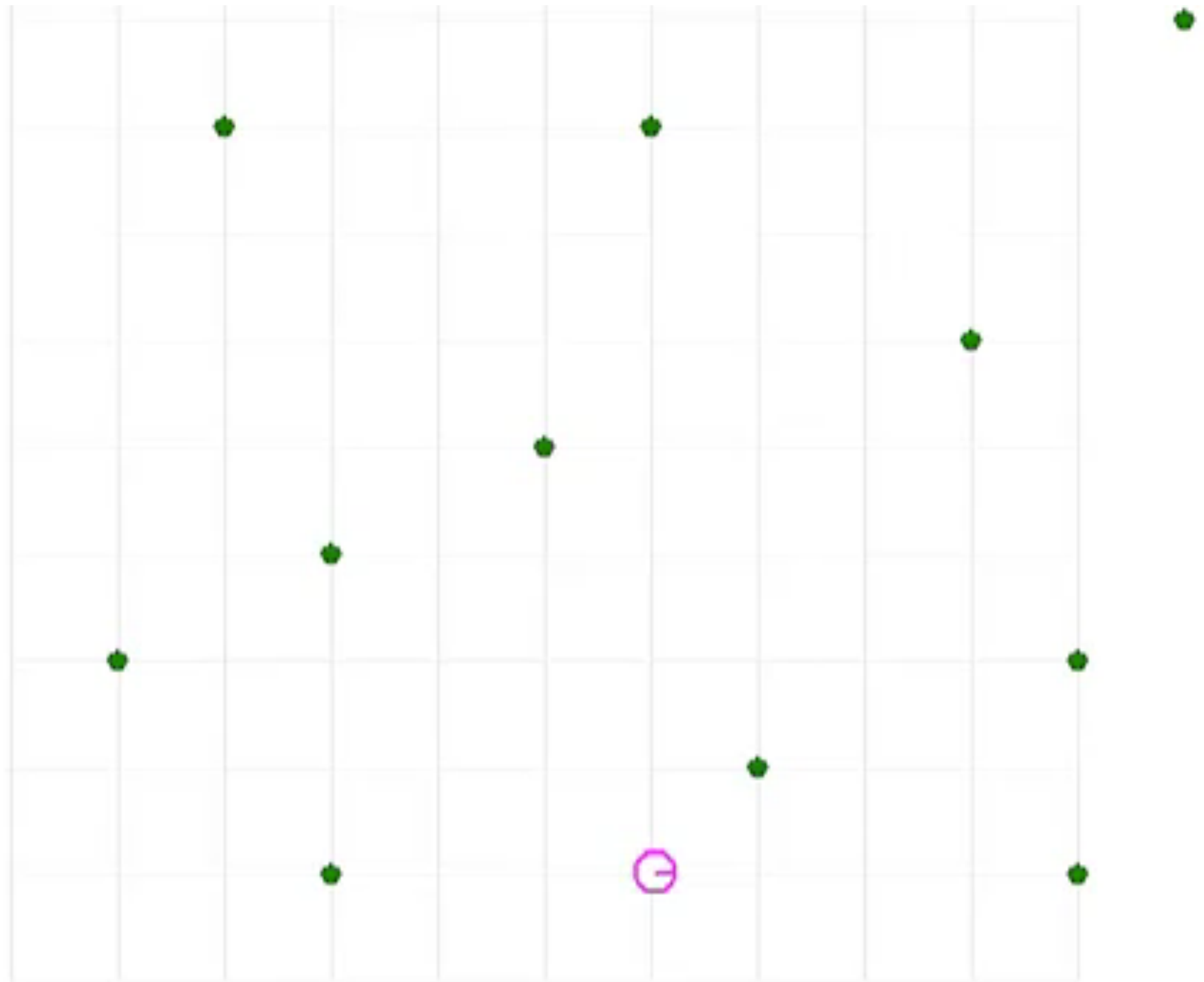
- take linear approximation around current state estimate

$$\underbrace{x_{k+1} = f(x_k, u_k, \epsilon_u)}_{\text{motion model}} \approx \tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k + B_u\tilde{\epsilon}_u$$

$$\underbrace{z_k = h(x_k, \epsilon_z)}_{\text{measurement model}} \approx \tilde{z}_k = C\tilde{x}_k + D_z\tilde{\epsilon}_z$$

- apply Kalman filter recursion to linearized model
(but propagate mean using nonlinear model)

EKF-SLAM



Limitations

- **Filter divergence:** linearization may be a poor approximation of original nonlinear system

$$\underbrace{x_{k+1} = f(x_k, u_k, \epsilon_u)}_{\text{motion model}} \approx \tilde{x}_{k+1} = A\tilde{x}_k + B\tilde{u}_k + B_u\tilde{\epsilon}_u$$

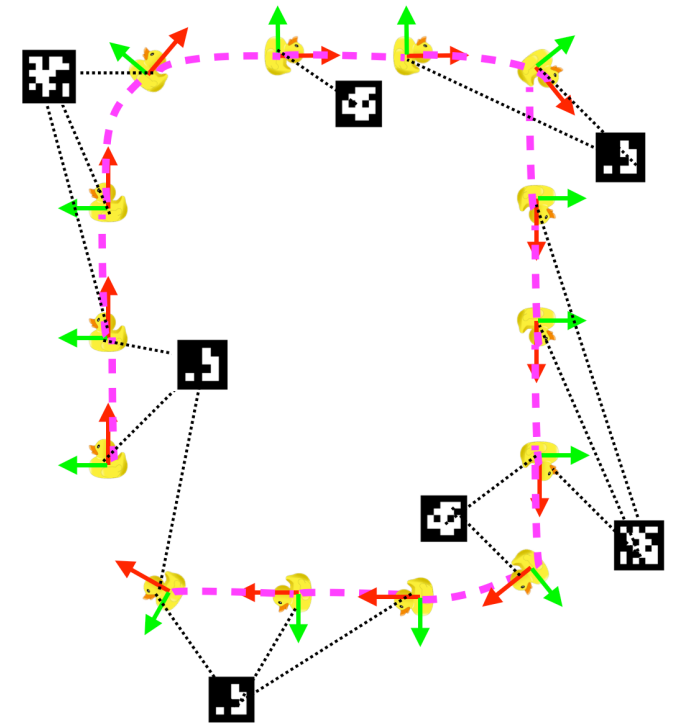
$$\underbrace{z_k = h(x_k, \epsilon_z)}_{\text{measurement model}} \approx \tilde{z}_k = C\tilde{x}_k + D_z\tilde{\epsilon}_z$$

- **Robustness:** noise, outliers
- **Scalability:** complexity is quadratic in the number of landmarks
- **Convergence? Consistency?**
hard to provide performance guarantees

Particle Filter-based SLAM

- estimate entire trajectory (+ landmarks):

$$x_k = \{p_1, \dots, p_k, l_1, \dots, l_M\}$$

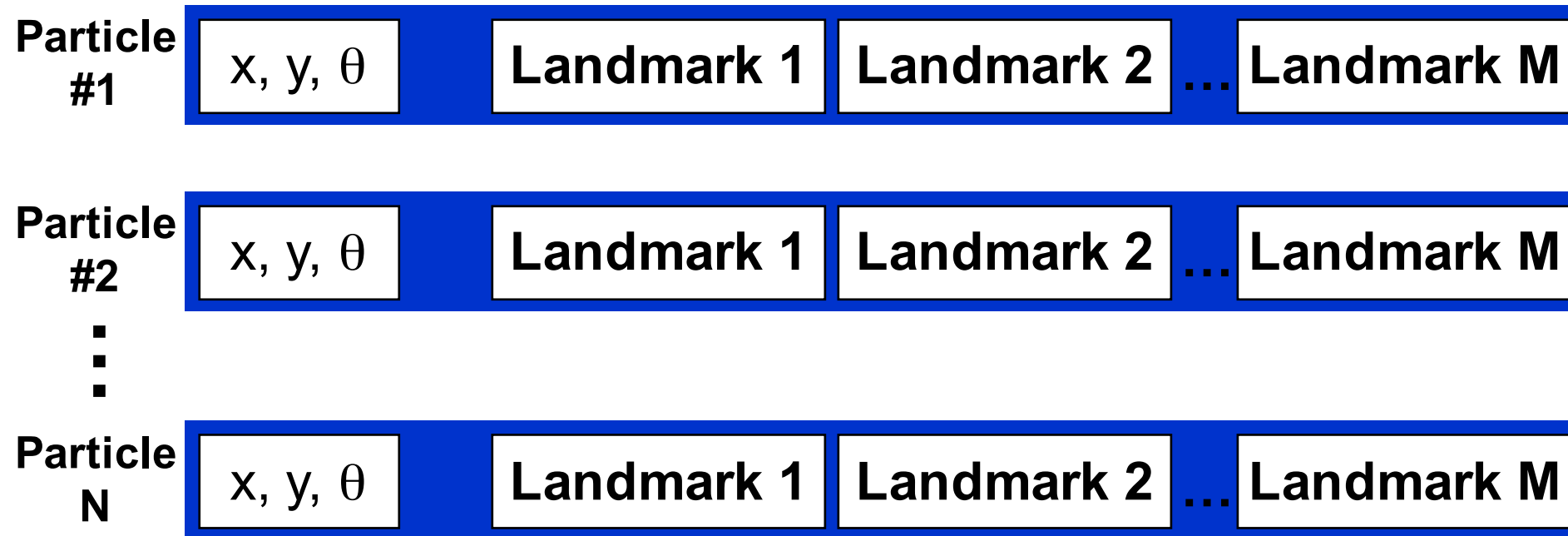


- posterior factorizes:

$$\begin{aligned} \mathbb{P}(x_k | u_{1:k}, z_{1:k}) &= \mathbb{P}(p_1, \dots, p_k, l_1, \dots, l_M | u_{1:k}, z_{1:k}) = \\ &= \underbrace{\mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k})}_{\text{trajectory posterior}} \underbrace{\mathbb{P}(l_1, \dots, l_M | p_1, \dots, p_k, u_{1:k}, z_{1:k})}_{\text{mapping from known poses}} \end{aligned}$$

Rao-Blackwellization

$$\begin{aligned}
 \mathbb{P}(x_k | u_{1:k}, z_{1:k}) &= \mathbb{P}(p_1, \dots, p_k, l_1, \dots, l_M | u_{1:k}, z_{1:k}) = \\
 &= \mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k}) \mathbb{P}(l_1, \dots, l_M | p_1, \dots, p_k, u_{1:k}, z_{1:k}) \\
 &= \mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k}) \prod_{i=1}^M \mathbb{P}(l_i | p_1, \dots, p_k, u_{1:k}, z_{1:k})
 \end{aligned}$$



Particle Filter-based SLAM

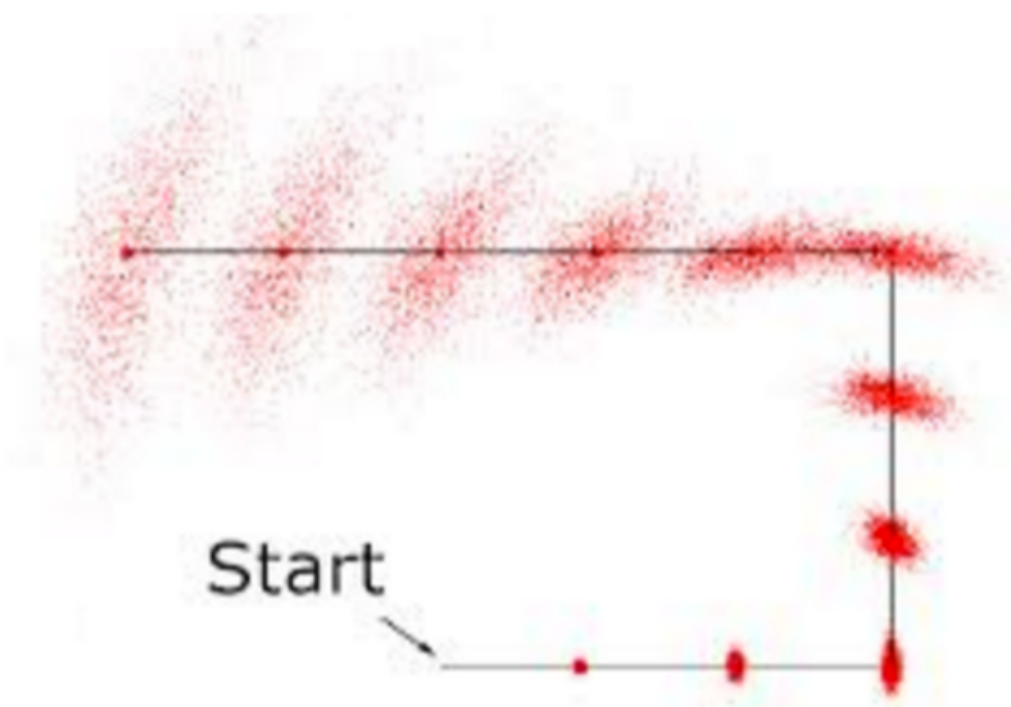
$$\begin{aligned}\mathbb{P}(x_k | u_{1:k}, z_{1:k}) &= \mathbb{P}(p_1, \dots, p_k, l_1, \dots, l_M | u_{1:k}, z_{1:k}) = \\ &= \underbrace{\mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k})}_{\text{trajectory posterior}} \underbrace{\mathbb{P}(l_1, \dots, l_M | p_1, \dots, p_k, u_{1:k}, z_{1:k})}_{\text{mapping from known poses}}\end{aligned}$$



Idea 1: use a set of weighted particles to approximate trajectory posterior



Idea 2: for each particle, do mapping from known poses



Particle Filter-based SLAM

$$\begin{aligned}\mathbb{P}(x_k | u_{1:k}, z_{1:k}) &= \mathbb{P}(p_1, \dots, p_k, l_1, \dots, l_M | u_{1:k}, z_{1:k}) = \\ &= \underbrace{\mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k})}_{\text{trajectory posterior}} \underbrace{\mathbb{P}(l_1, \dots, l_M | p_1, \dots, p_k, u_{1:k}, z_{1:k})}_{\text{mapping from known poses}}\end{aligned}$$



Idea 1: use a set of weighted particles to approximate trajectory posterior



Idea 2: for each particle, do mapping from known poses



Particle Filter-based SLAM

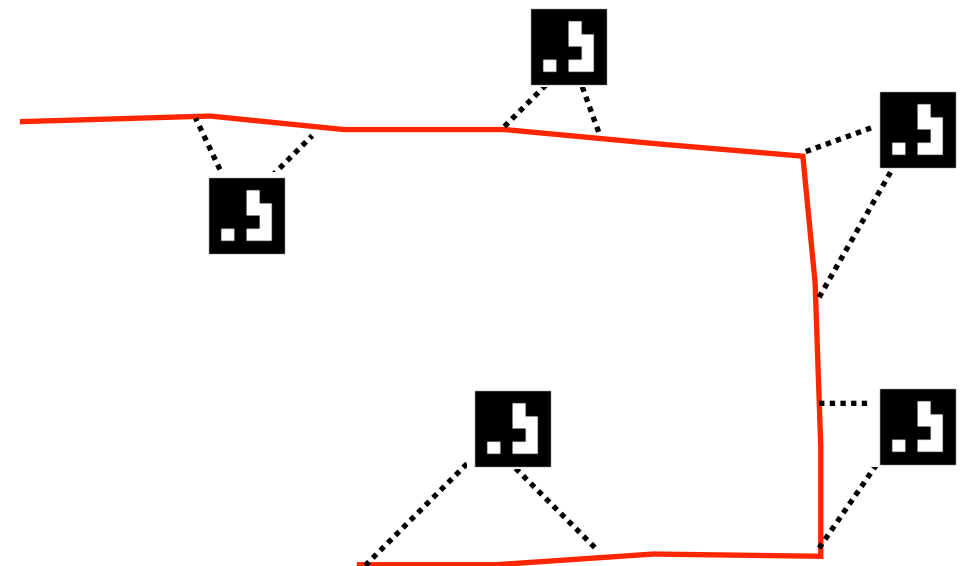
$$\begin{aligned}\mathbb{P}(x_k | u_{1:k}, z_{1:k}) &= \mathbb{P}(p_1, \dots, p_k, l_1, \dots, l_M | u_{1:k}, z_{1:k}) = \\ &= \underbrace{\mathbb{P}(p_1, \dots, p_k | u_{1:k}, z_{1:k})}_{\text{trajectory posterior}} \underbrace{\mathbb{P}(l_1, \dots, l_M | p_1, \dots, p_k, u_{1:k}, z_{1:k})}_{\text{mapping from known poses}}\end{aligned}$$



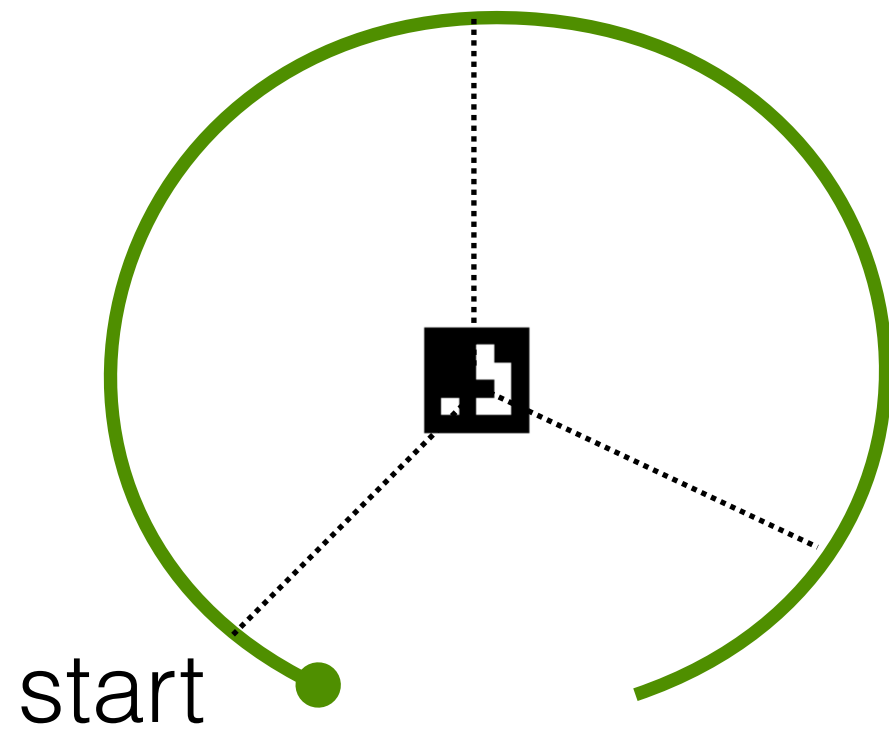
Idea 1: use a set of weighted particles to approximate trajectory posterior



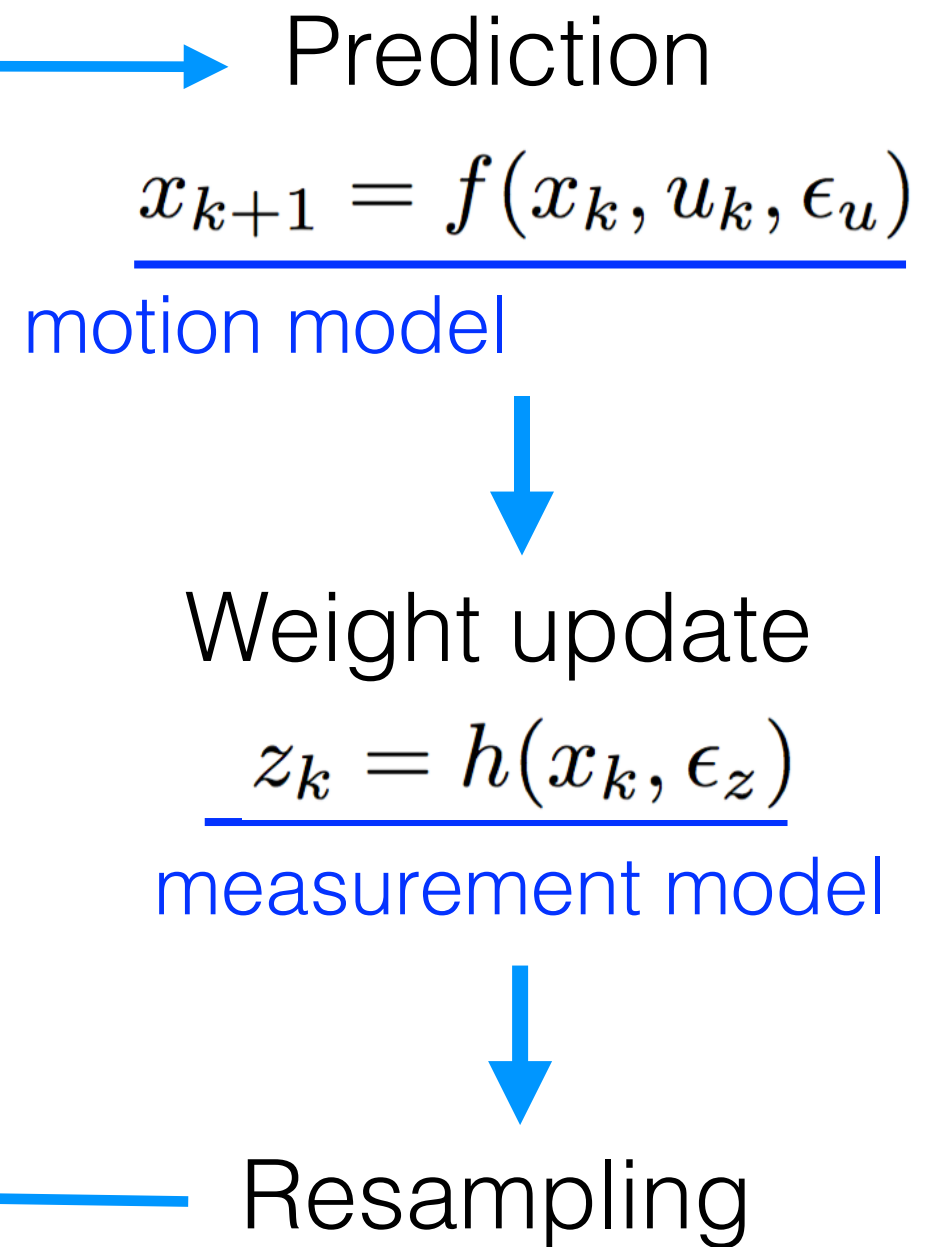
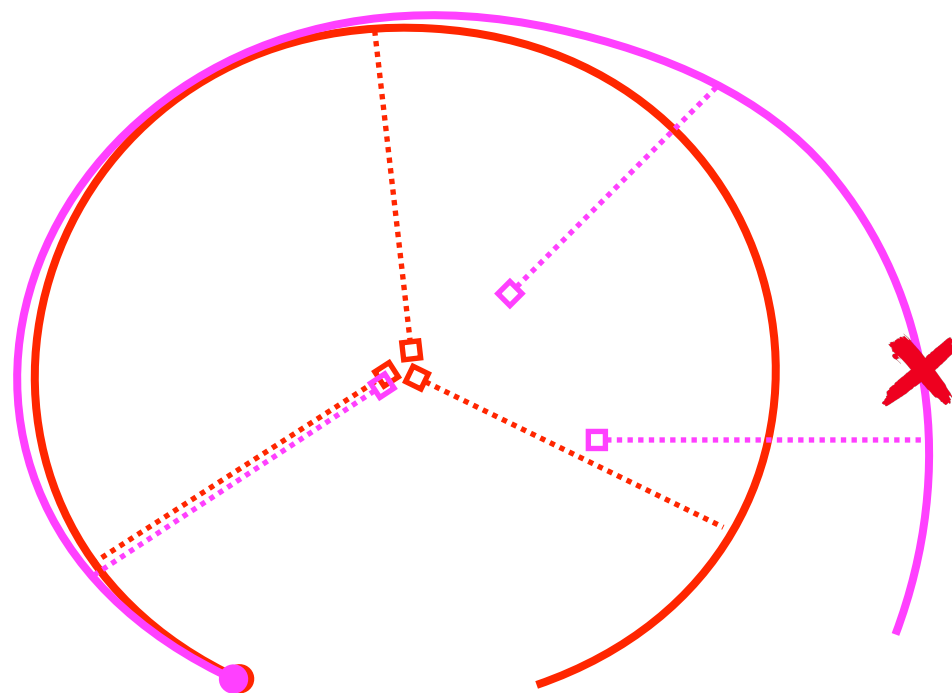
Idea 2: for each particle, do mapping from known poses



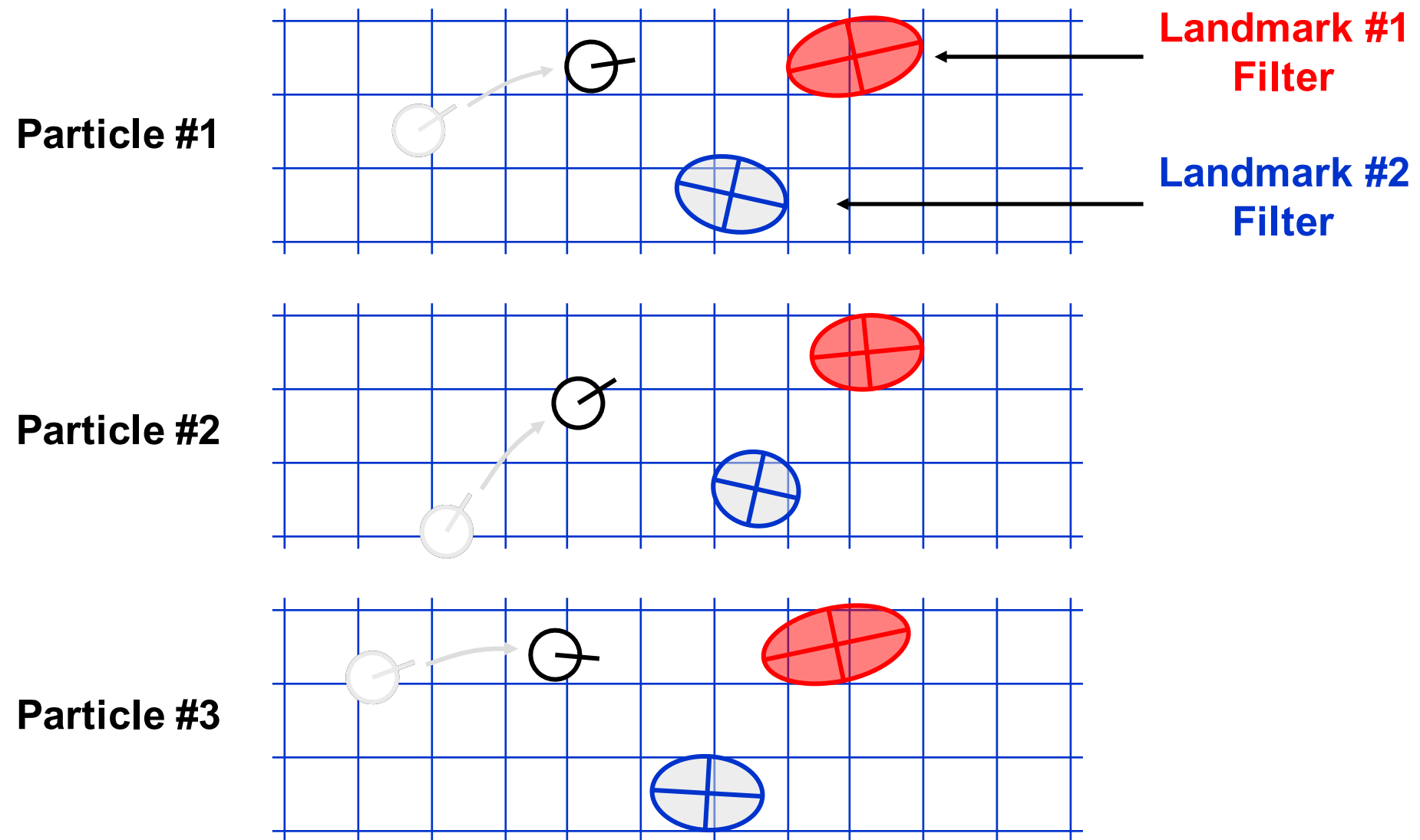
Particle Filter-based SLAM



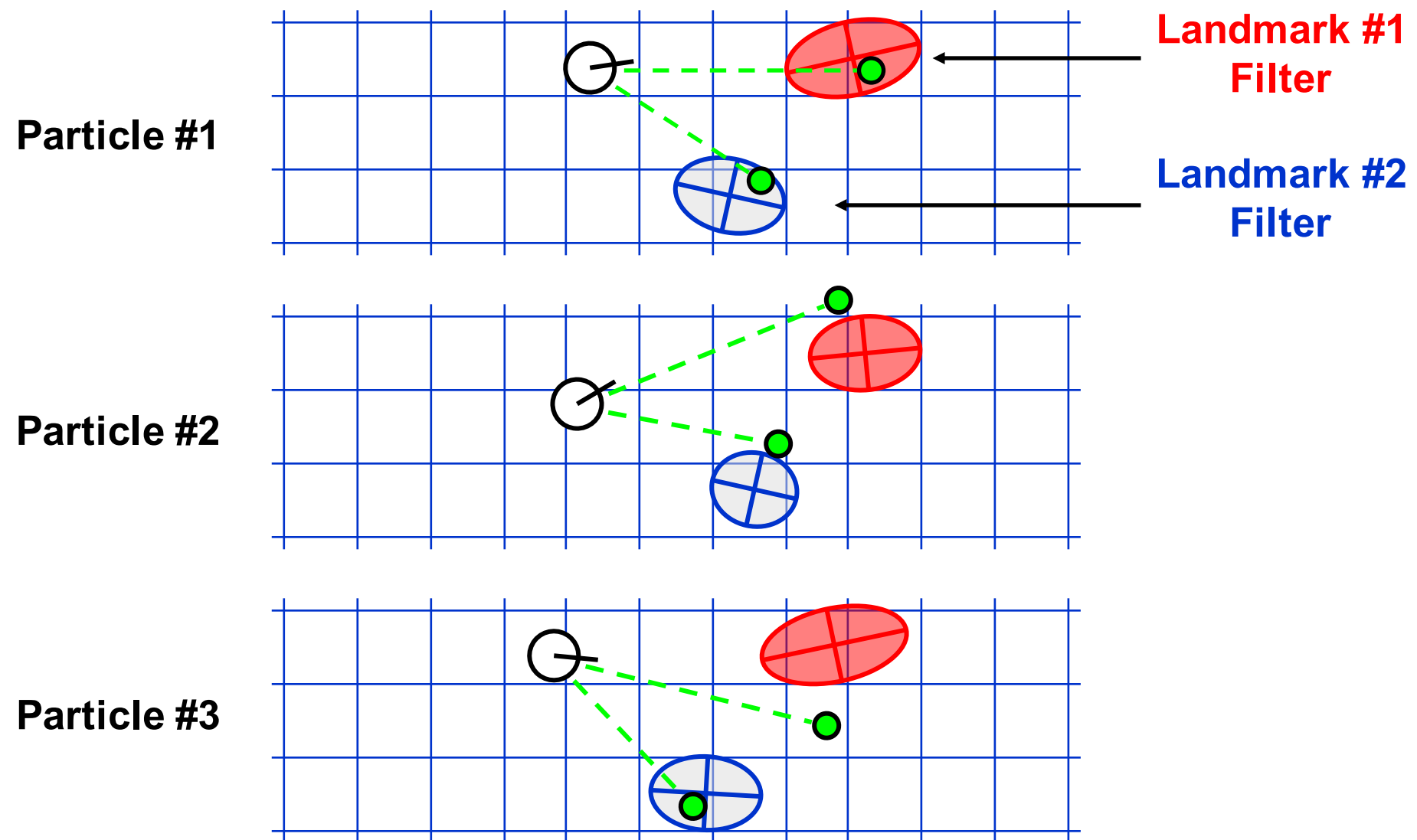
2 samples



FAST-SLAM - Prediction

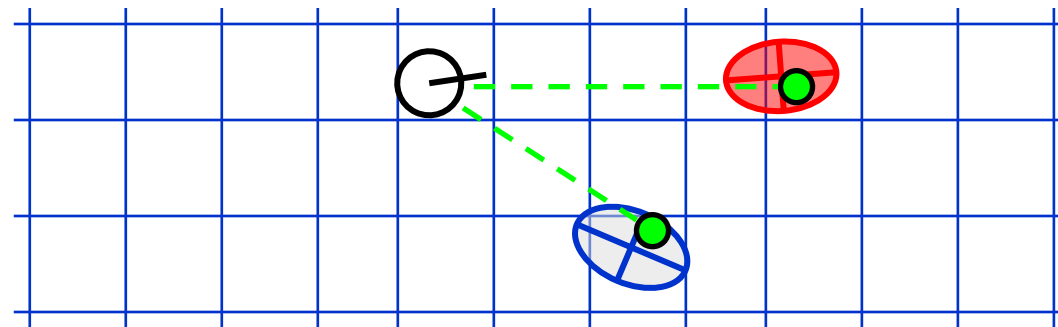


FAST-SLAM Update (pt. 1)



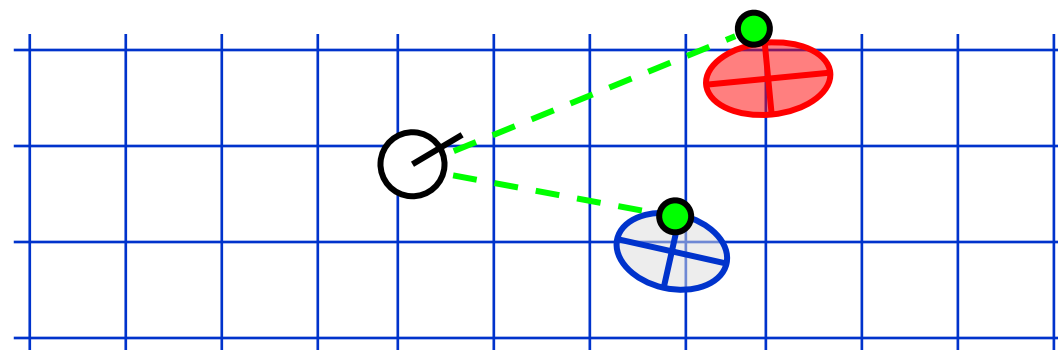
FAST-SLAM Update (pt. 2)

Particle #1



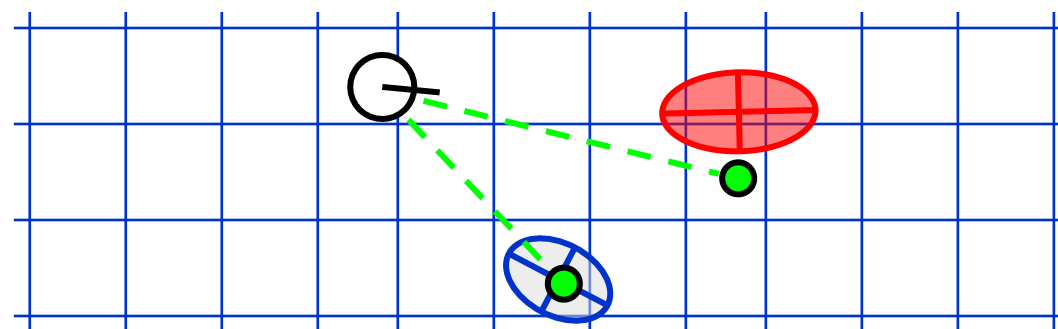
Weight = 0.8

Particle #2



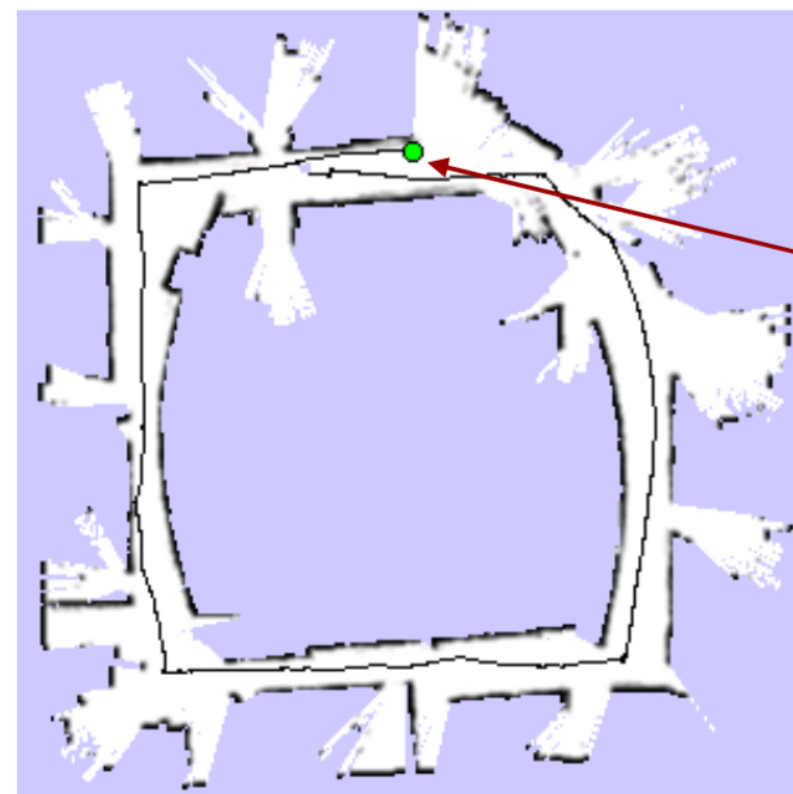
Weight = 0.4

Particle #3



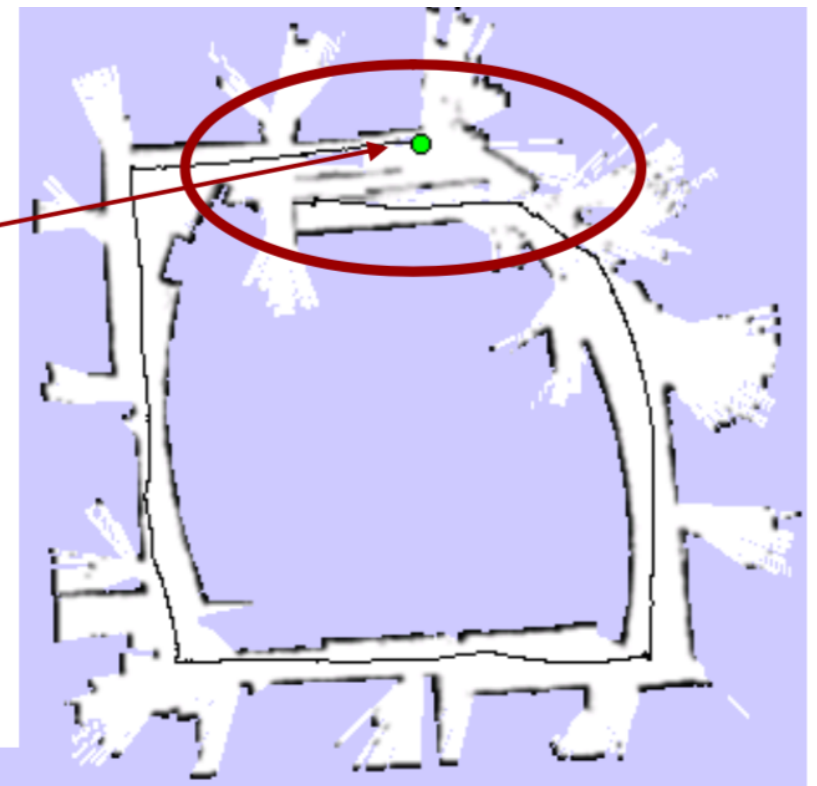
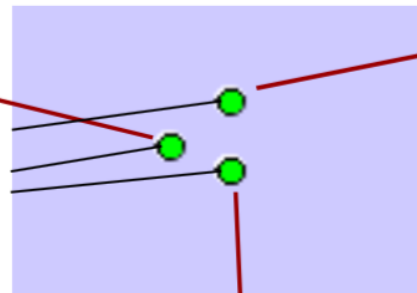
Weight = 0.1

Particle Filter View-based SLAM

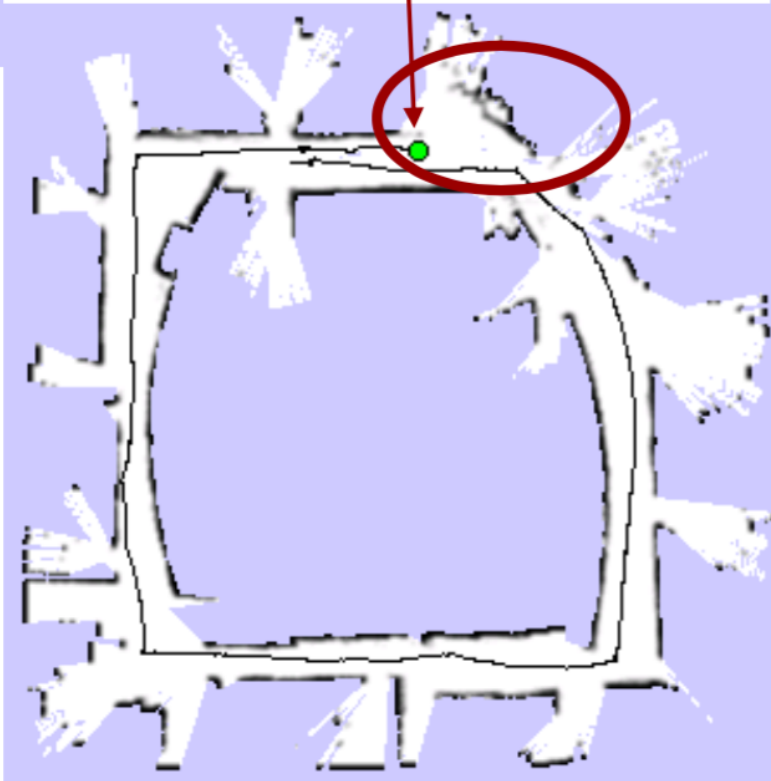


map of particle 1

3 particles



map of particle 3

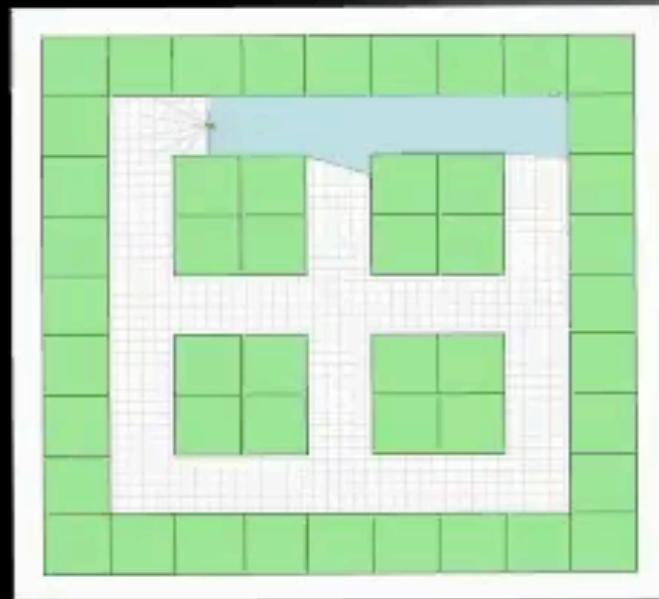


map of particle 2

Grisetti, 2007

Particle Filter view-based SLAM

samples represent possible trajectories of the robot



Top:
ground truth

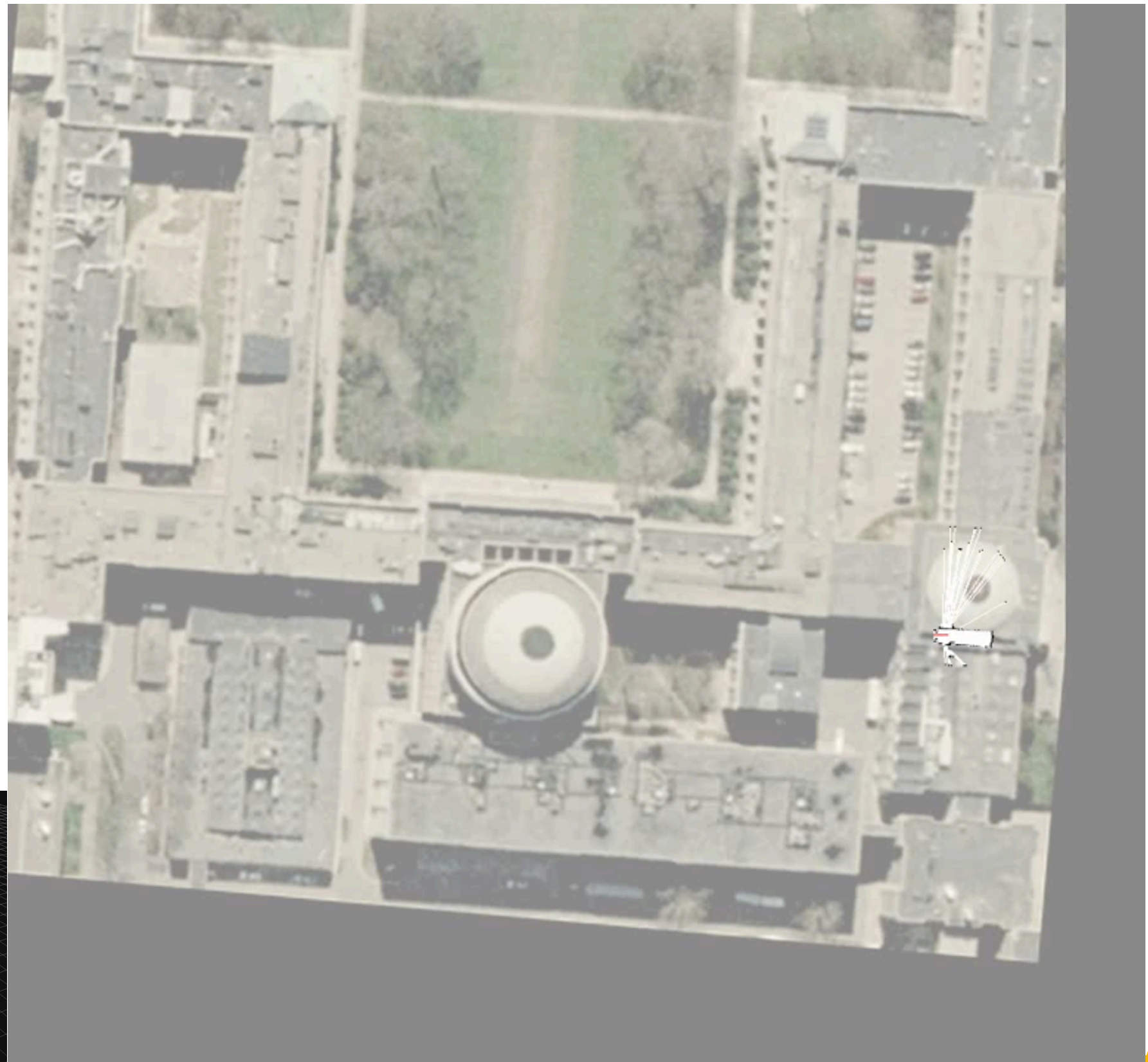
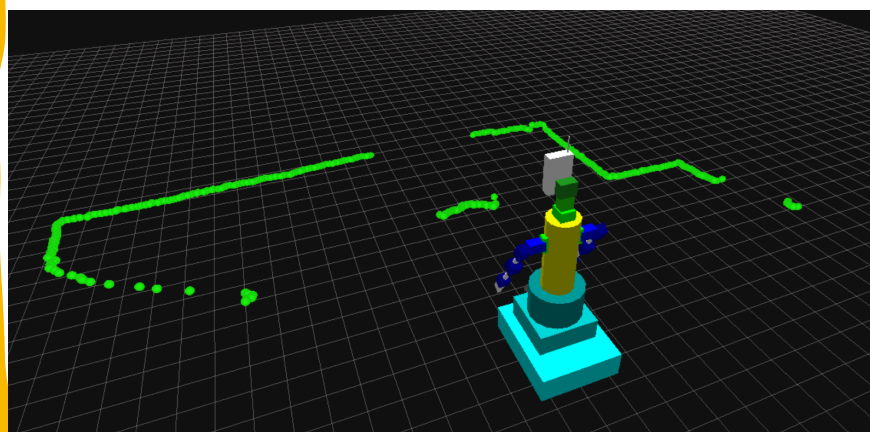
Left:
PF-SLAM

Right:
mapping
from
odometry

PF-SLAM in ROS

gmapping: ROS package for PF-SLAM

Plug & play, but requires a 2D laser scanner and a decent odometry



[courtesy of Cyrill Stachniss]

Advantages & Limitations

Relax EKF assumption ✓

nonlinear, non-Gaussian,
less strict requirements on measurement model

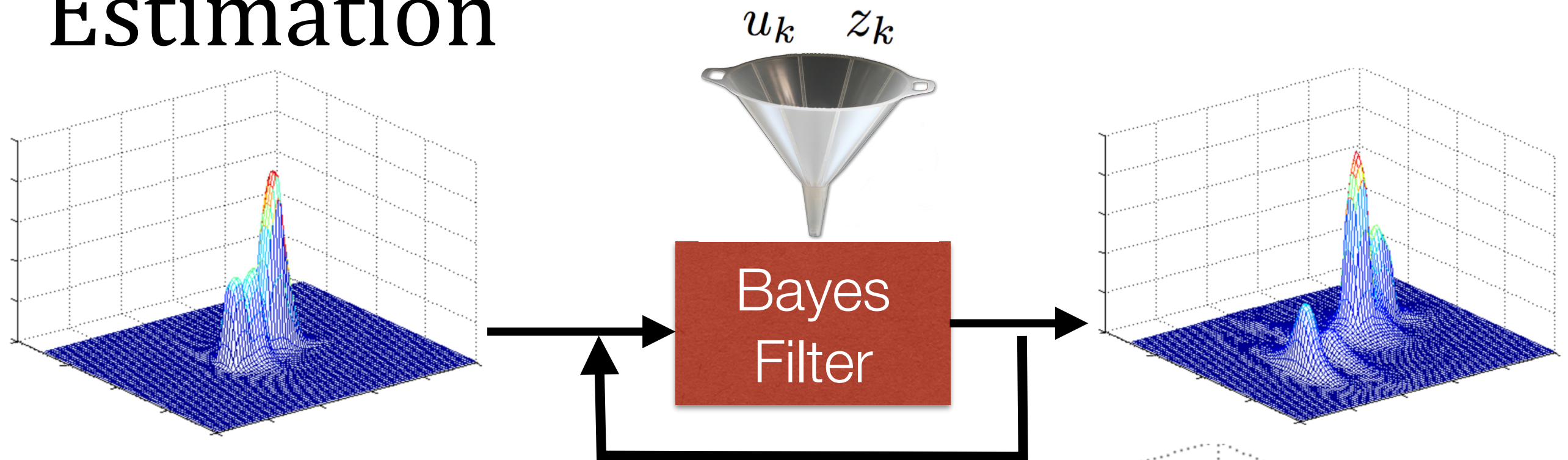
Particle depletion ✗

when uncertainty is large, a limited number of particles is a poor approximation of the posterior (poor performance in 3D)

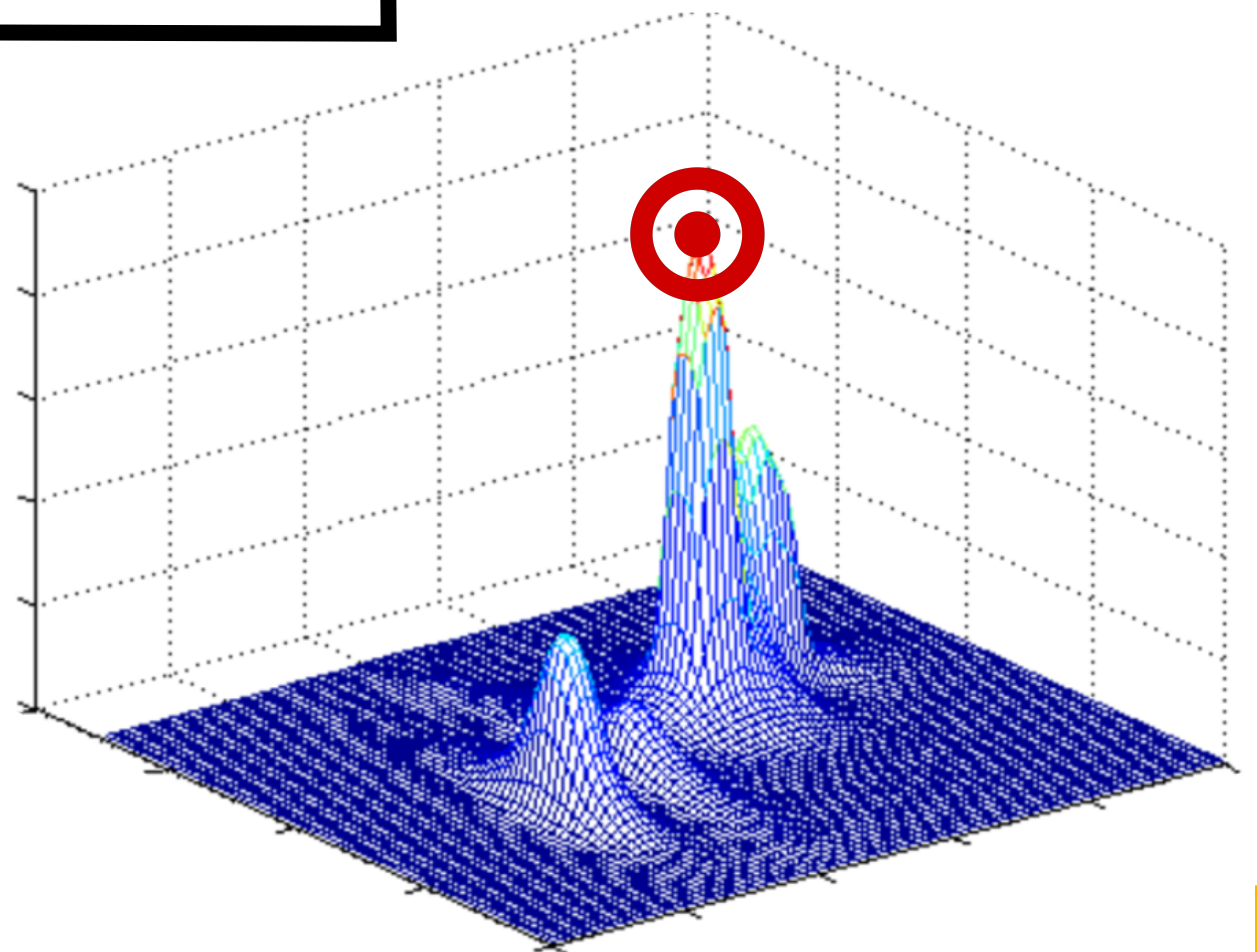
particles VS computational cost ✗

the more particles, the better the estimate, but computationally expensive

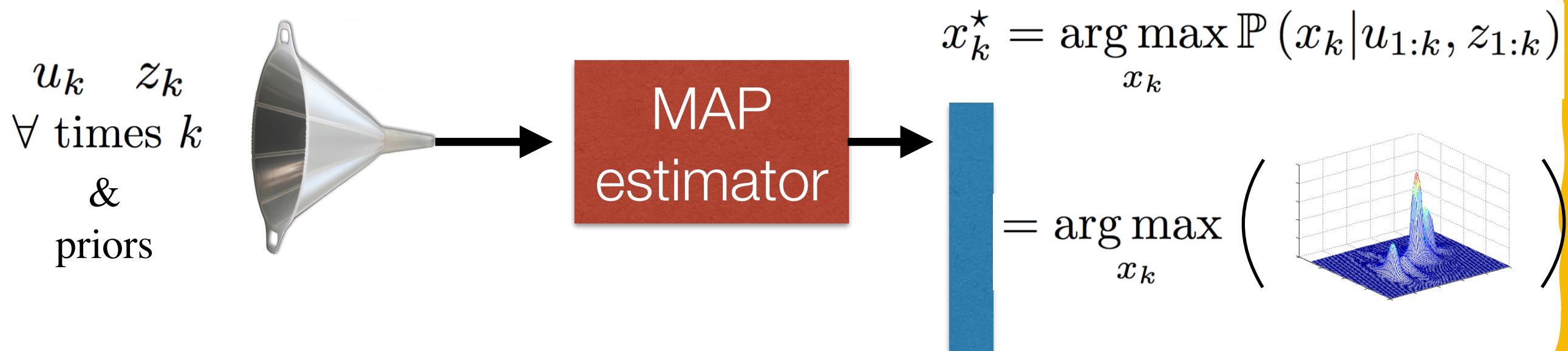
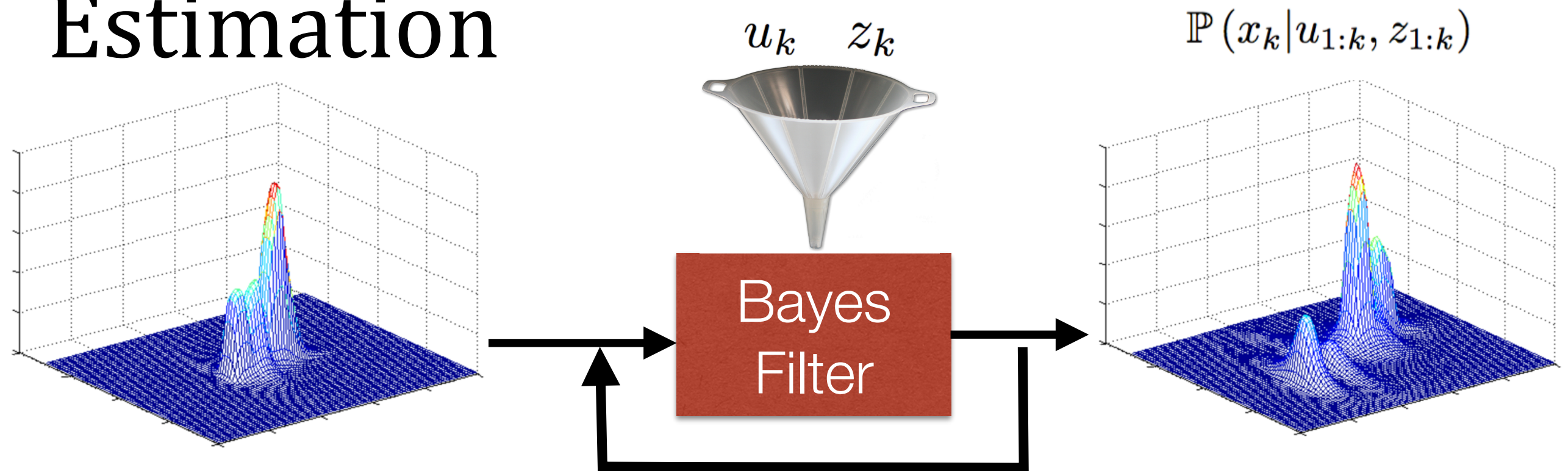
Maximum a posteriori (MAP) Estimation



- MAP estimation is an alternative to Bayes filter
- rather than computing/approximating posterior, we look for its peak



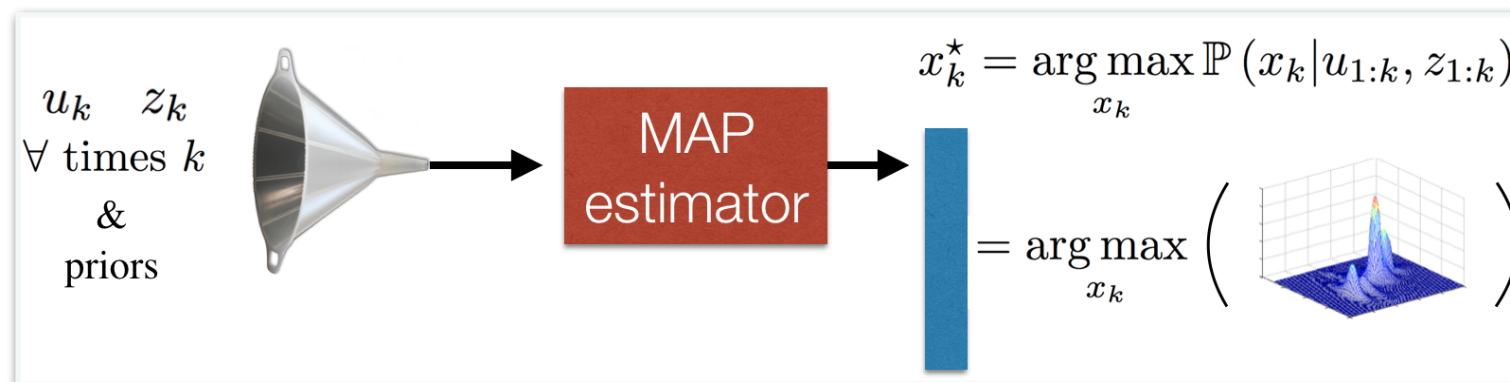
Maximum a posteriori (MAP) Estimation



Maximum a posteriori (MAP) Estimation

- we estimate the entire trajectory and the landmark poses

$$x_k = \{p_1, \dots, p_k, l_1, \dots, l_M\}$$



Posterior Probability

- Posterior probability:

$$\mathbb{P}(x|u_{1:k}, z_{1:k}) = \alpha \underbrace{\mathbb{P}(u_{1:k}, z_{1:k}|x)}_{\text{measurement likelihood}} \underbrace{\mathbb{P}(x)}_{\text{prior}}$$

- Measurements are independent (recall, $\mathbb{P}(A, B) = \mathbb{P}(A)\mathbb{P}(B)$)

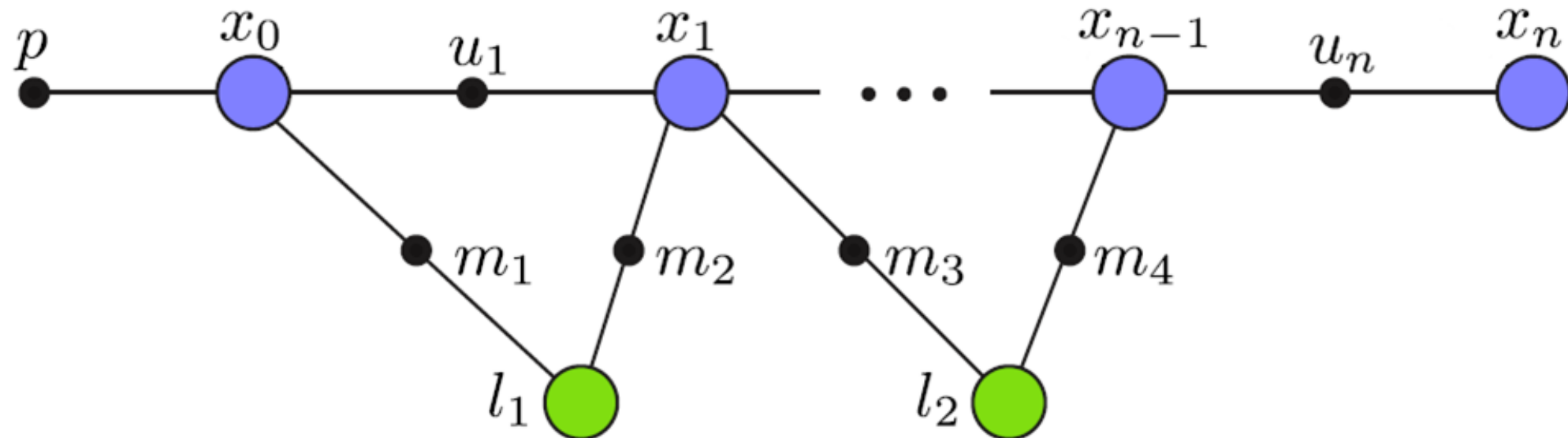
$$\mathbb{P}(x|u_{1:k}, z_{1:k}) = \alpha \mathbb{P}(u_1|x) \mathbb{P}(z_1|x) \dots \mathbb{P}(u_k|x) \mathbb{P}(z_k|x) \mathbb{P}(x)$$

- **MAP estimate:**

$$x^* = \arg \max_x \mathbb{P}(u_1|x) \mathbb{P}(z_1|x) \dots \mathbb{P}(u_k|x) \mathbb{P}(z_k|x) \mathbb{P}(x)$$

without prior becomes **maximum likelihood** estimation

The Factor Graph



Linear Gaussian (LG) Model

- Linear measurements with additive Gaussian noise:

$$z_k = C_k x + \epsilon_z \quad \epsilon_z \sim \mathcal{N}(0, R)$$

$$u_k = A_k x + \epsilon_u \quad \epsilon_u \sim \mathcal{N}(0, Q)$$

- Measurement Likelihoods:

$$\mathbb{P}(z_k|x) = \gamma_u \exp \left((z_k - C_k x)^\top Q^{-1} (z_k - C_k x) \right) = \gamma_z \exp \|z_k - C_k x\|_R^2$$

$$\mathbb{P}(u_k|x) = \gamma_z \exp \left((u_k - A_k x)^\top R^{-1} (u_k - A_k x) \right) = \gamma_u \exp \|u_k - A_k x\|_Q^2$$

- Gaussian prior: $\mathbb{P}(x) = \gamma_x \exp \|x - \mu\|_\Omega^2$

MAP Estimation in LG Models

- MAP estimate:

$$x^* = \arg \max_x \mathbb{P}(u_1|x) \mathbb{P}(z_1|x) \dots \mathbb{P}(u_k|x) \mathbb{P}(z_k|x) \mathbb{P}(x)$$



$$x^* = \arg \max_x \prod_k \left(\exp \|z_k - C_k x\|_R^2 \exp \|u_k - A_k x\|_Q^2 \right) \exp \|x - \mu\|_\Sigma^2$$

- Max posterior = min negative log-posterior

$$x^* = \arg \max_x \mathbb{P}(x|u_{1:k}, z_{1:k}) = \arg \min -\log \mathbb{P}(x|u_{1:k}, z_{1:k})$$



$$x^* = \arg \min_x \sum_k \left(\|z_k - A_z x\|_R^2 + \|u_k - A_u x\|_Q^2 \right) + \|x - \mu\|_\Sigma^2$$

linear least squares problem - admits closed form solution!

but world is nonlinear...

$$z_k = h_k(x) + \epsilon_z$$

$$u_k = f_k(x) + \epsilon_u$$

nonlinear
functions

$$\epsilon_z \sim \mathcal{N}(0, R)$$

$$\epsilon_u \sim \mathcal{N}(0, Q)$$

✓ The derivation is similar to previous slides and we get:

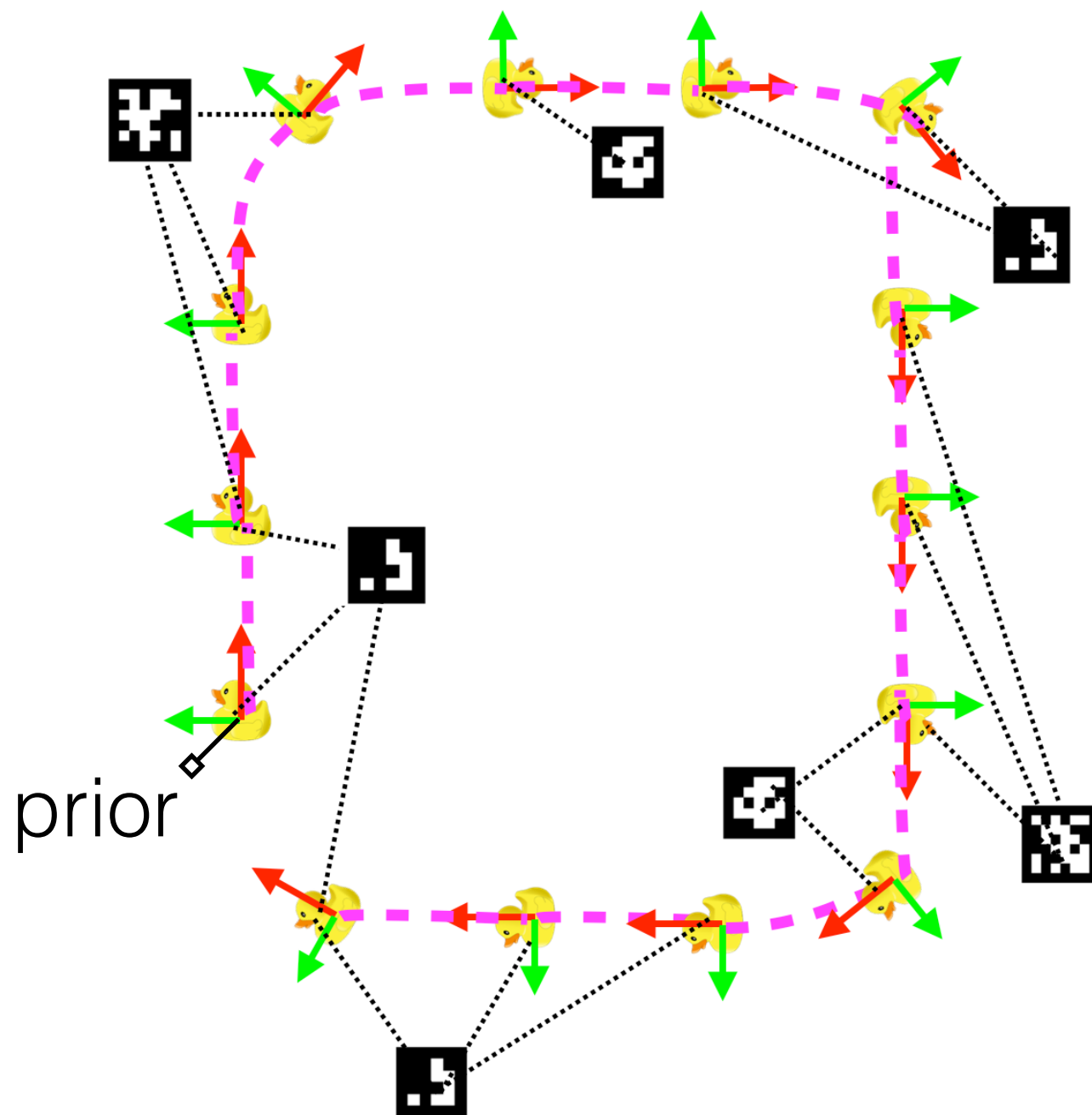
$$x^* = \arg \min_x \sum_k \left(\|z_k - h_k(x)\|_R^2 + \|u_k - f_k(x)\|_Q^2 \right) + \|\mu - g(x)\|_\Sigma^2$$

nonlinear least squares problem

it can be solved efficiently using iterative solvers

Graph interpretation

$$x^* = \arg \min_x \sum_k \left(\|z_k - h_k(x)\|_R^2 + \|u_k - f_k(x)\|_Q^2 \right) + \|\mu - g(x)\|_\Sigma^2$$



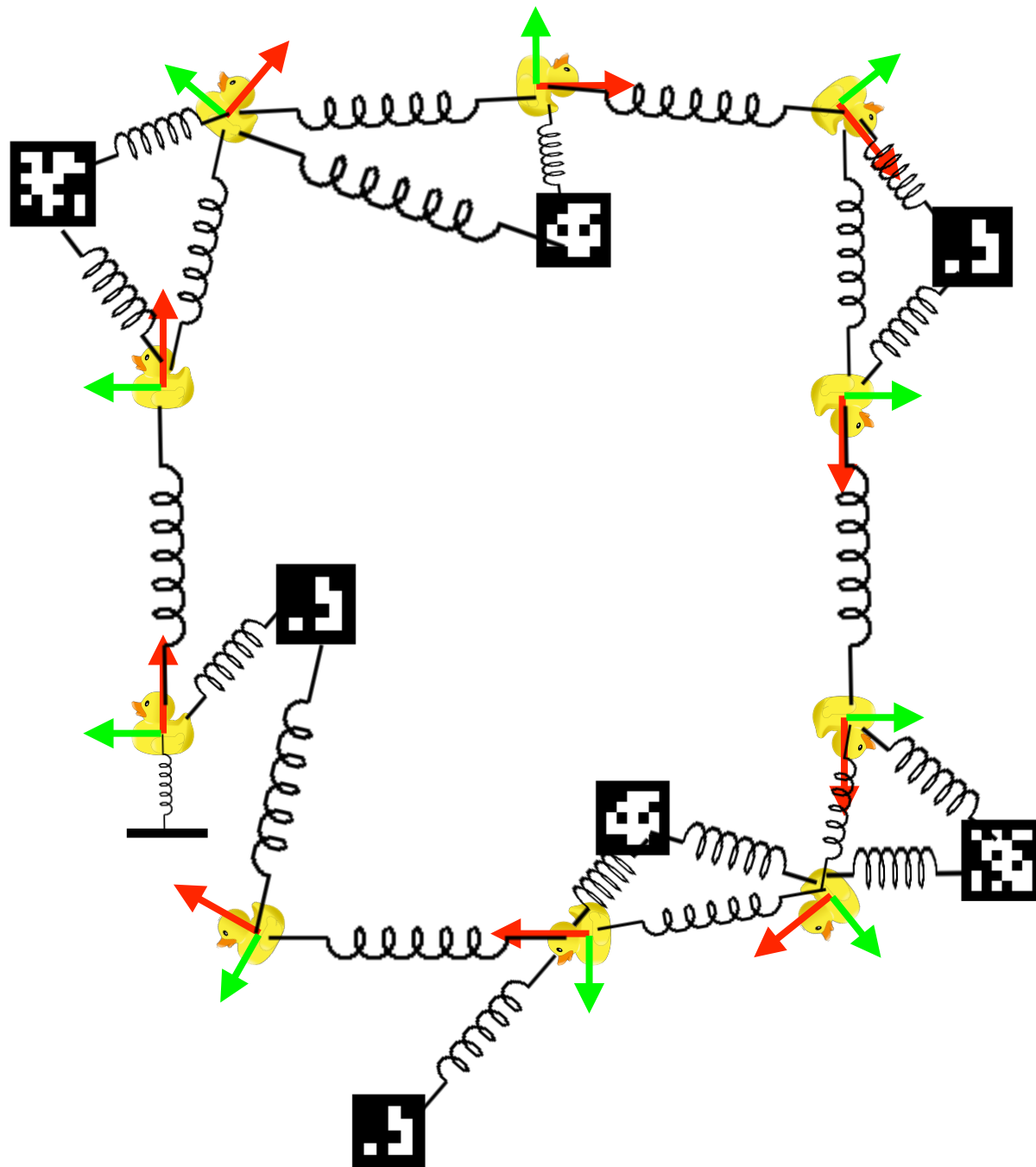
Graph:

- **vertices**: poses to be estimated
- **edges**: measurements between pair of poses

Pose Graph Optimization

Mechanical interpretation

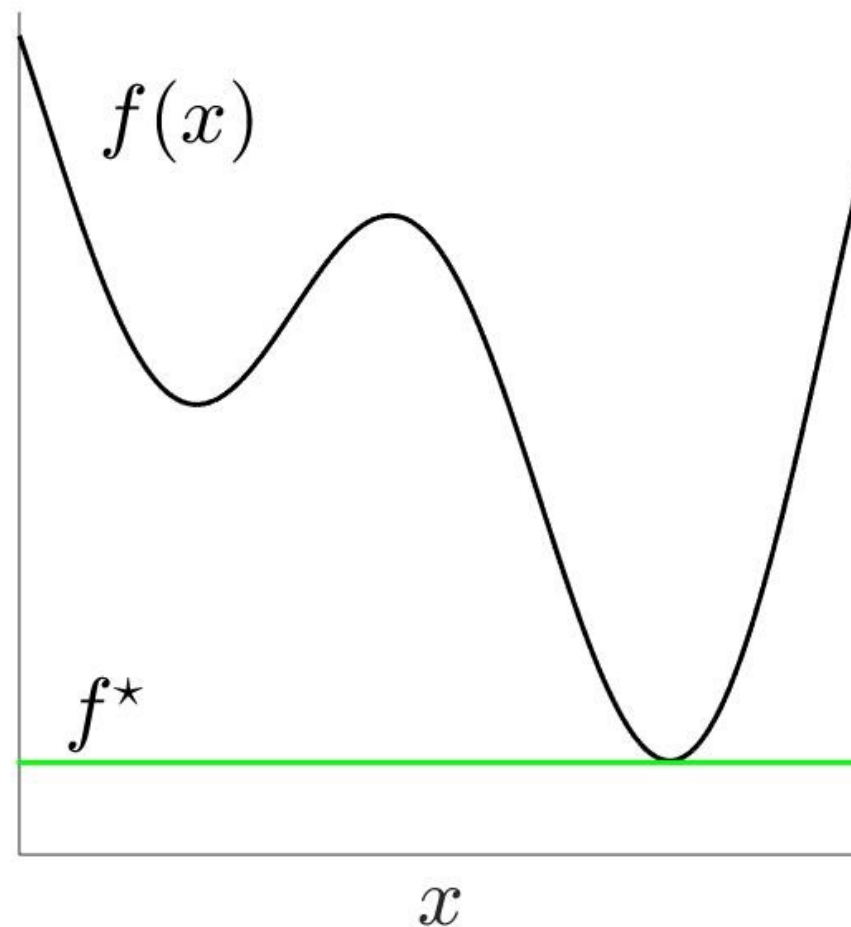
$$x^* = \arg \min_x \sum_k \left(\|z_k - h_k(x)\|_R^2 + \|u_k - f_k(x)\|_Q^2 \right) + \|\mu - g(x)\|_\Sigma^2$$



- measurements as **springs** connecting pairs of masses
- MAP estimate is the position of the masses minimizing energy

Iterative optimization: Gauss-Newton Method

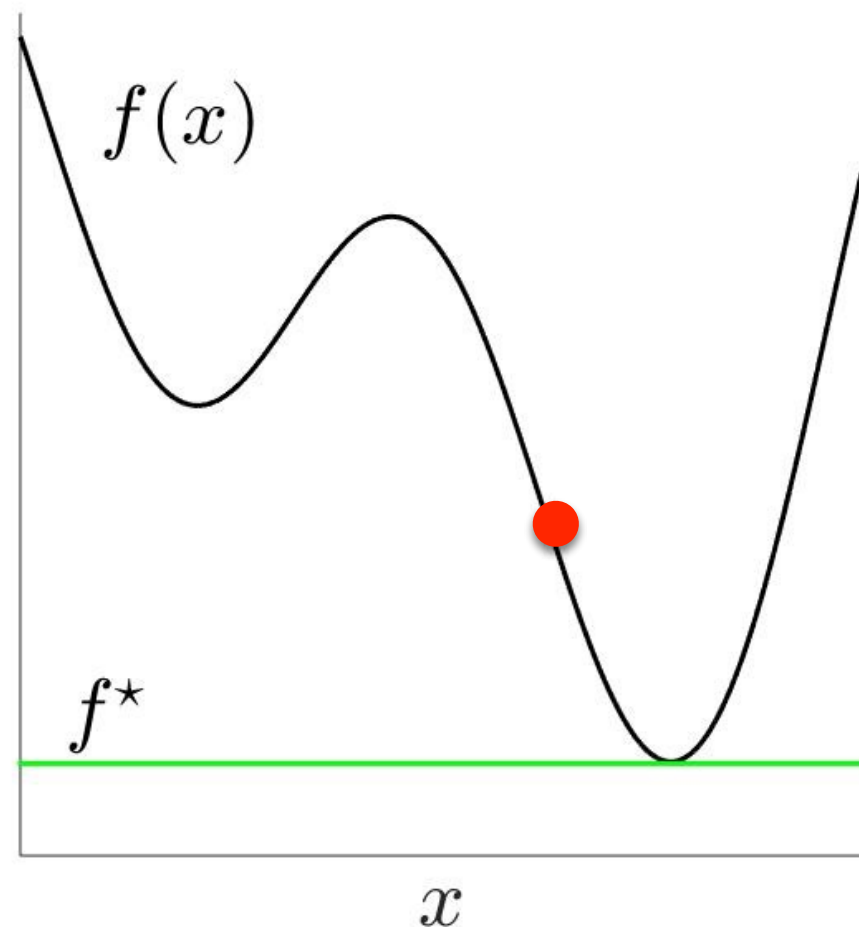
$$\min_x \sum_i \|r_i(x)\|^2$$



Imagine this,
but in 100k
dimensional
space

Iterative optimization: Gauss-Newton Method

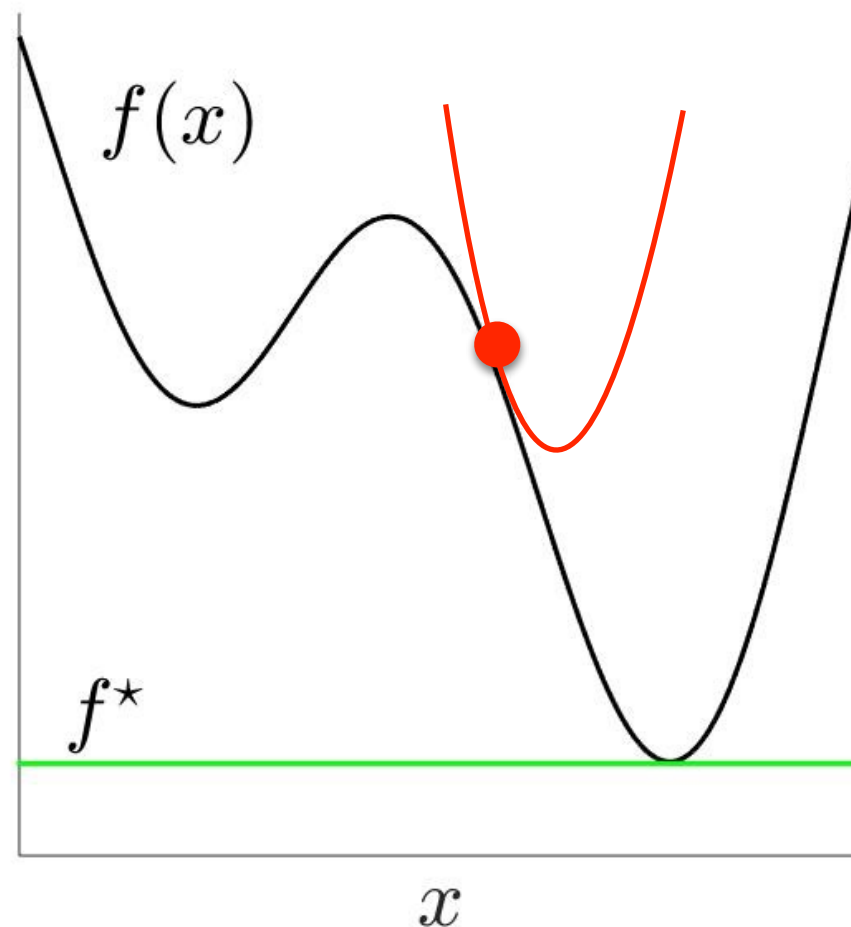
$$\min_x \sum_i \|r_i(x)\|^2 \rightarrow \hat{x}$$



Iterative optimization: Gauss-Newton Method

$$\min_x \sum_i \|r_i(x)\|^2 \rightarrow \hat{x} \quad \text{linearization}$$

$$\min_{\tilde{x}} \|A\tilde{x} - b\|^2$$

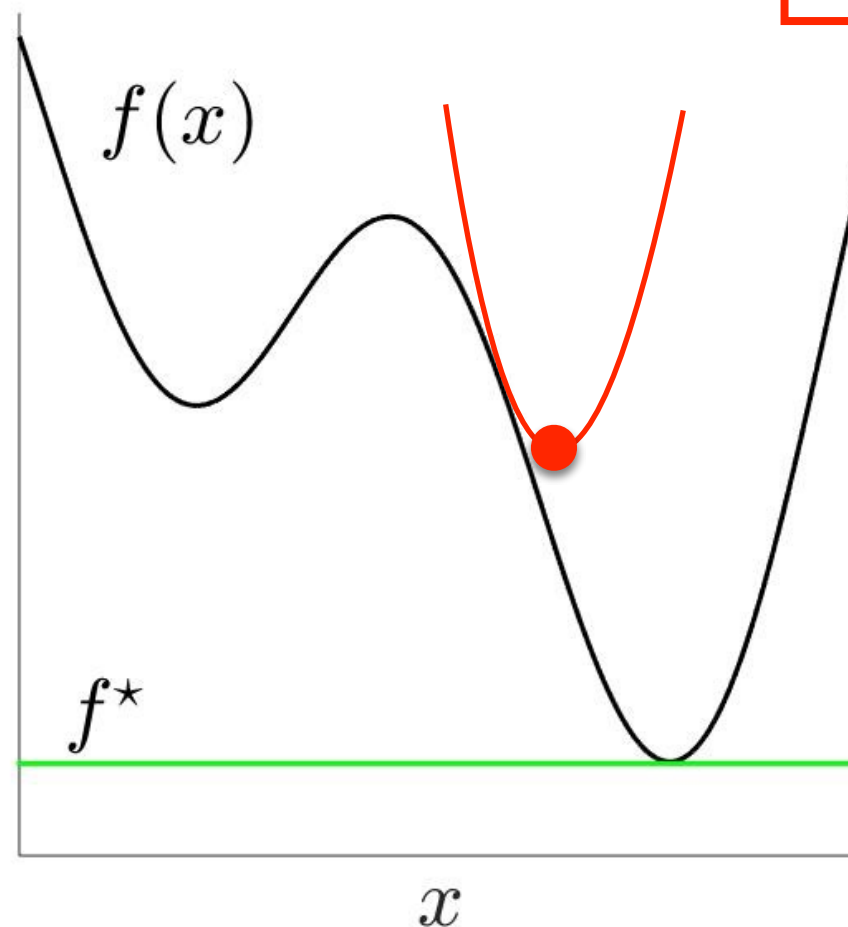


Iterative optimization: Gauss-Newton Method

$$\min_x \sum_i \|r_i(x)\|^2 \rightarrow \hat{x} \quad \text{linearization}$$

$$\min_{\tilde{x}} \|A\tilde{x} - b\|^2$$

$$\tilde{x}^* = (A^T A)^{-1} A^T b$$



Iterative optimization: Gauss-Newton Method

$$\min_x \sum_i \|r_i(x)\|^2$$

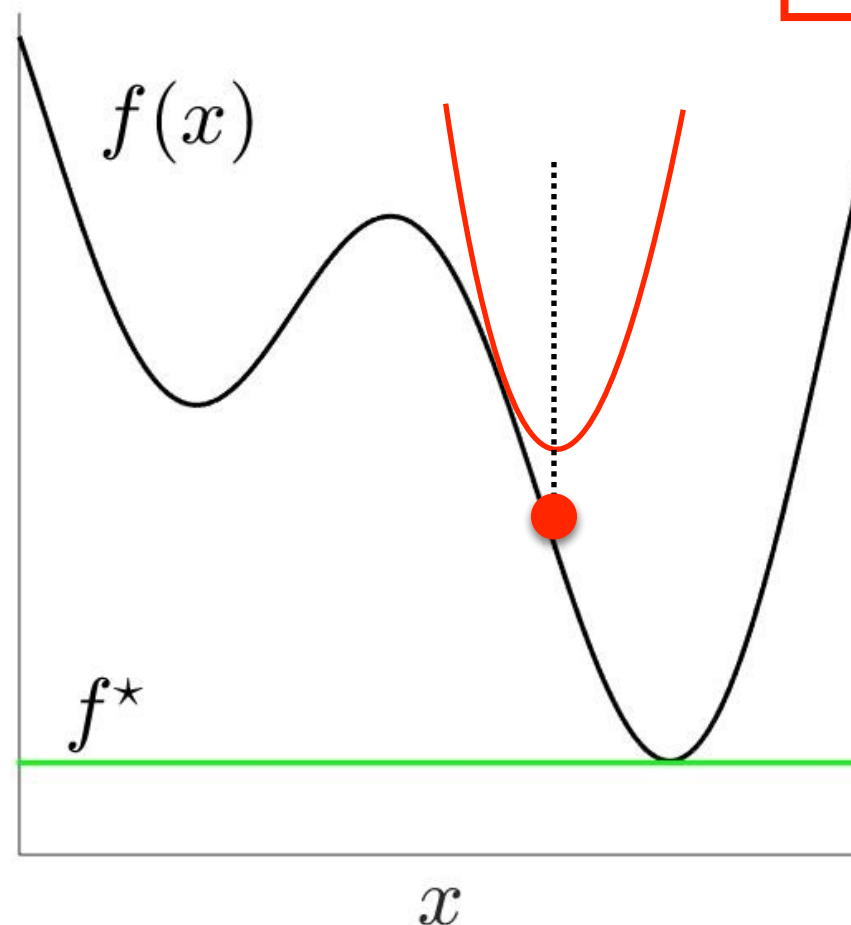
$\rightarrow \hat{x}$ linearization

$$\hat{x} \leftarrow \hat{x} + \tilde{x}^*$$

correction

$$\min_{\tilde{x}} \|A\tilde{x} - b\|^2$$

$$\tilde{x}^* = (A^T A)^{-1} A^T b$$



Iterative optimization: Gauss-Newton Method

$$\min_x \sum_i \|r_i(x)\|^2$$

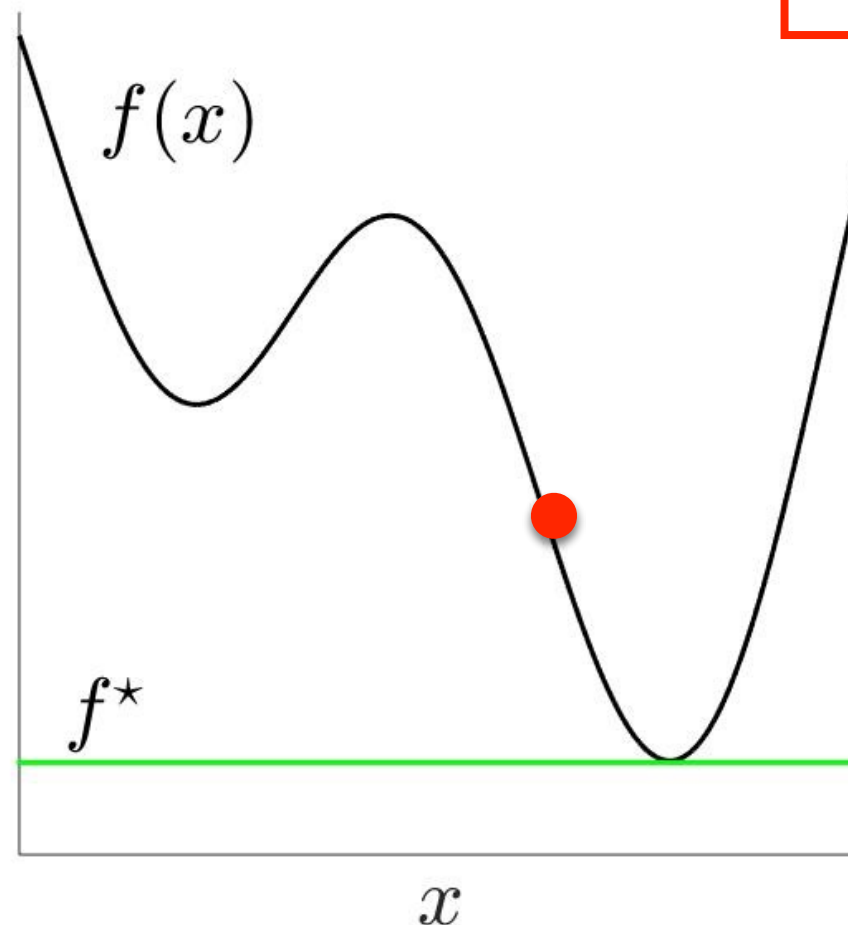
$\rightarrow \hat{x}$ linearization

$$\hat{x} \leftarrow \hat{x} + \tilde{x}^*$$

correction

$$\min_{\tilde{x}} \|A\tilde{x} - b\|^2$$

$$\tilde{x}^* = (A^T A)^{-1} A^T b$$



Iterative optimization: Gauss-Newton Method

$$\min_x \sum_i \|r_i(x)\|^2$$

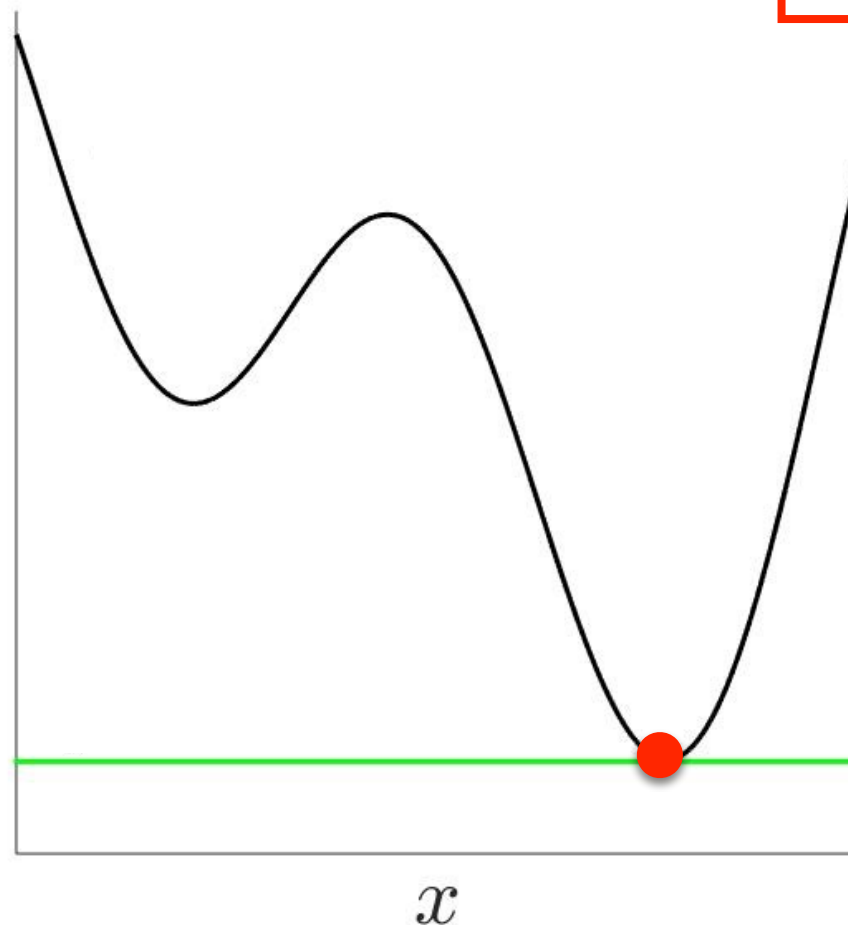
$\rightarrow \hat{x}$ linearization

$$\hat{x} \leftarrow \hat{x} + \tilde{x}^*$$

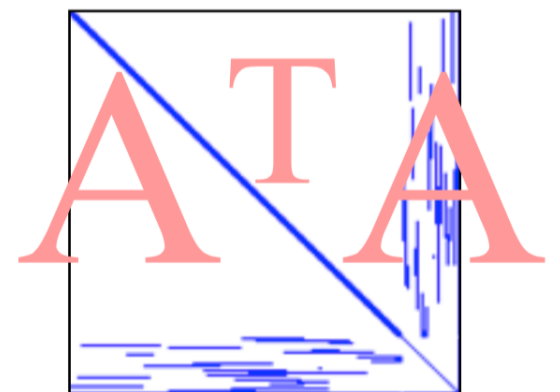
correction

$$\min_{\tilde{x}} \|A\tilde{x} - b\|^2$$

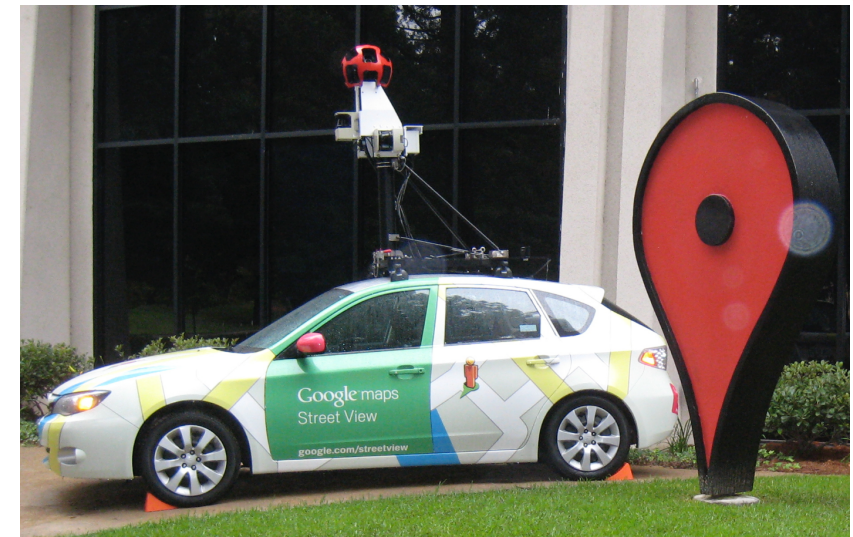
$$\tilde{x}^* = (A^T A)^{-1} A^T b$$



$$(A^T A)\tilde{x}^* = A^T b$$



Google Street View



Challenges: MAP Estimation for SLAM

- **Large scale optimization:**

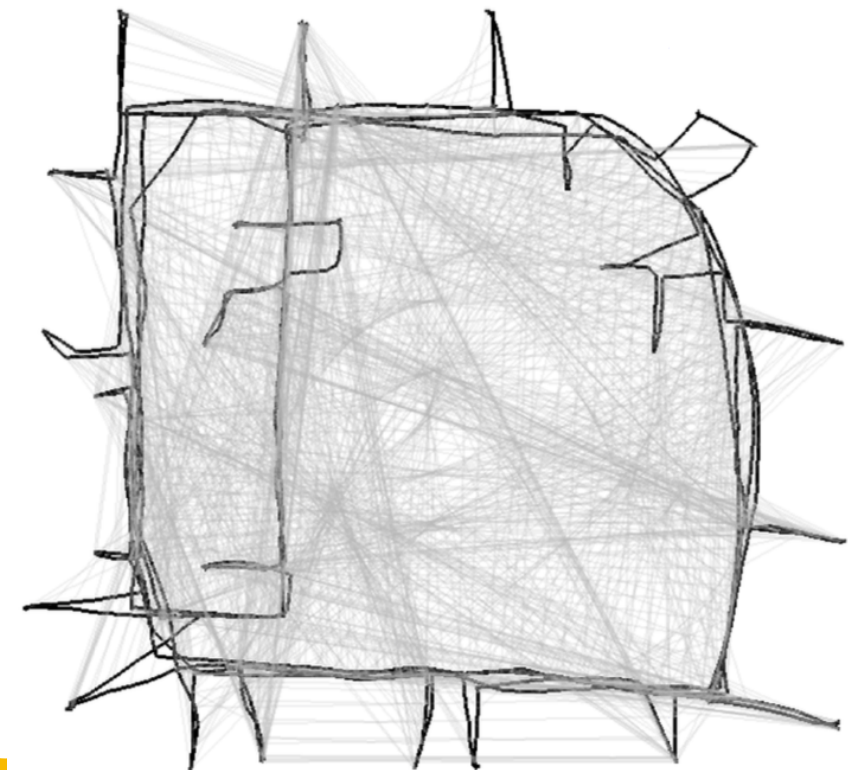
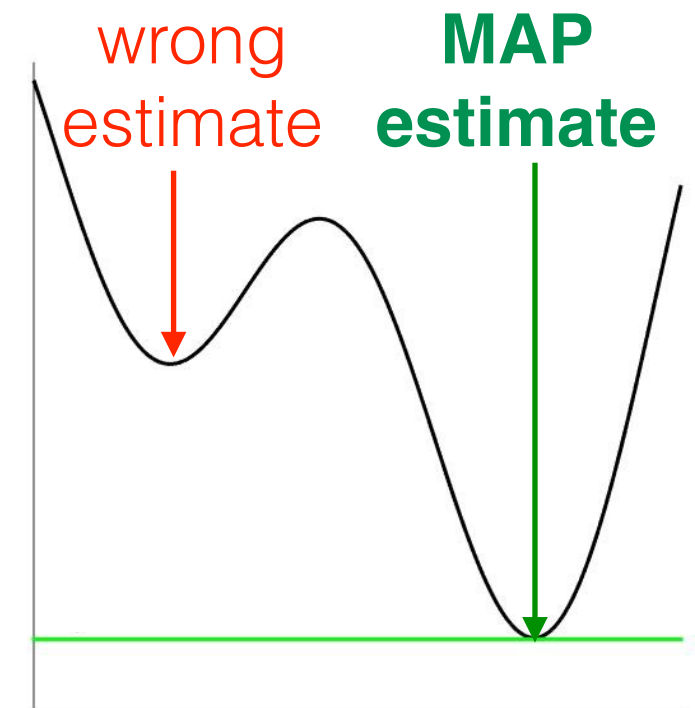
- ✓ efficient & incremental solvers exist
- ✗ problem keeps growing over time

- **Non-convex optimization**

- ✓ available initialization techniques (new!)
- ✓ possible to study convergence (new!)

- **Not robust to outliers**

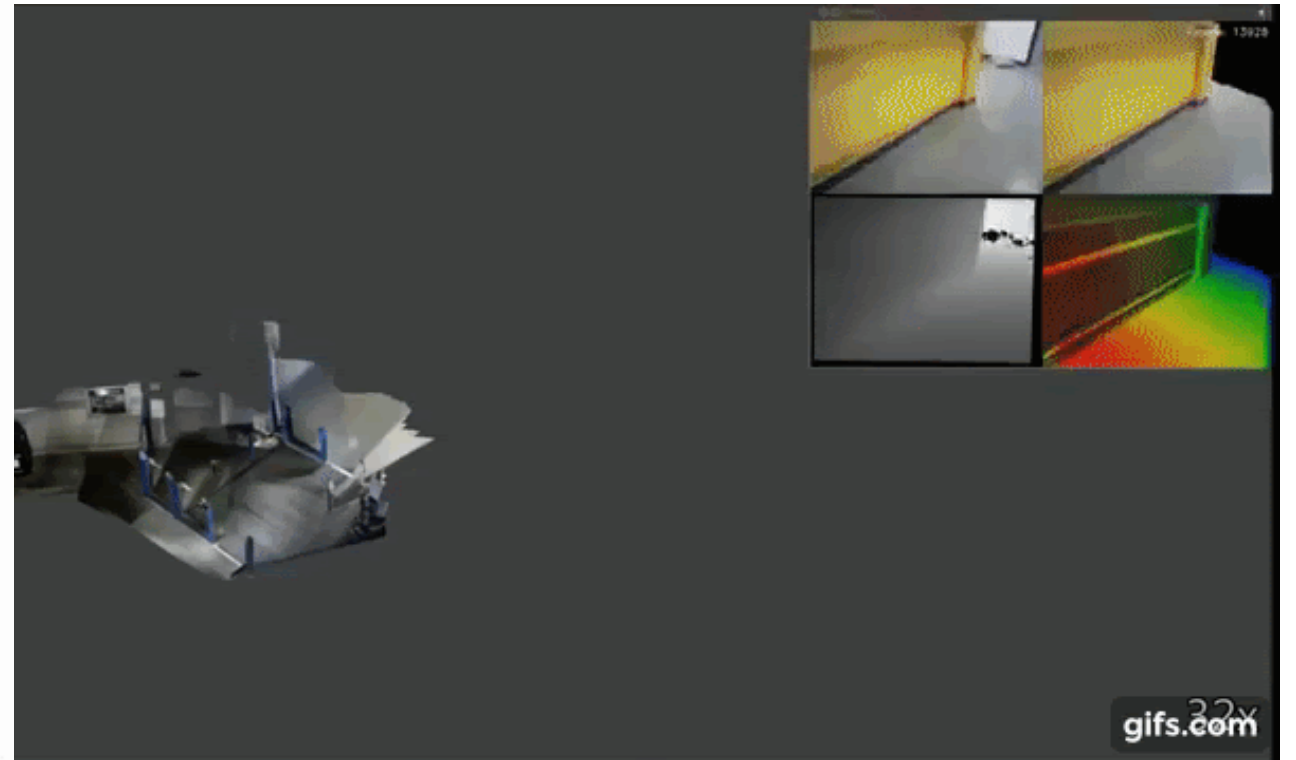
- ✓ possible to improve robustness using robust cost functions
- ✗ need more research on performance guarantees
- ✗ dynamic environments



Modern SLAM systems



gifs.com



- [1] Keller et al. *Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion*
- [2] Whelan et al. *Kintinuous: Spatially Extended KinectFusion*

SLAM: Challenges

- Robustness



SLAM
solution

- Scalability



- Representation



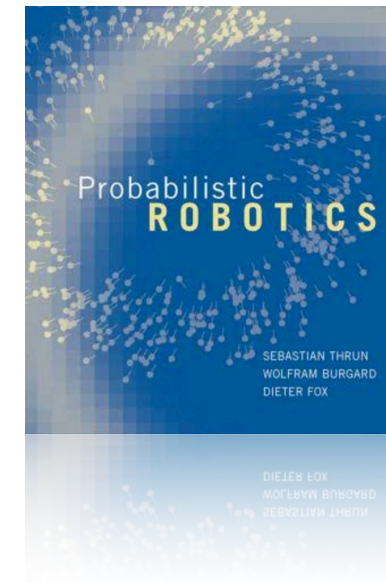
- Theoretical guarantees



References

Books

- S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. The MIT press, Cambridge, MA, 2005.



Surveys & tutorials

- S. Thrun, “Robotic mapping: a survey,” in Exploring artificial intelligence in the new millennium. Morgan Kaufmann, Inc., 2003, pp. 1–35.
- H. Durrant-Whyte and T. Bailey, “Simultaneous localisation and mapping (SLAM): Part I the essential algorithms,” Robotics & Automation Magazine, 2006.
- T. Bailey and H. Durrant-Whyte, “Simultaneous localisation and mapping (SLAM): Part II state of the art,” Robotics & Automation Magazine, 2006.
- G. Grisetti, R. Kummerle, C. Stachniss, W. Burgard, A Tutorial on Graph-Based SLAM, IEEE Intelligent Transportation Systems Magazine, (2)4:31-43, 2010.

Online references & slides

- Visual SLAM Tutorial, CVPR 2014 (http://frc.ri.cmu.edu/~kaess/vslam_cvpr14/)
- Future goals and indicator or progress in SLAM, RSS 2015 (<http://ylatif.github.io/movingsensors/>)