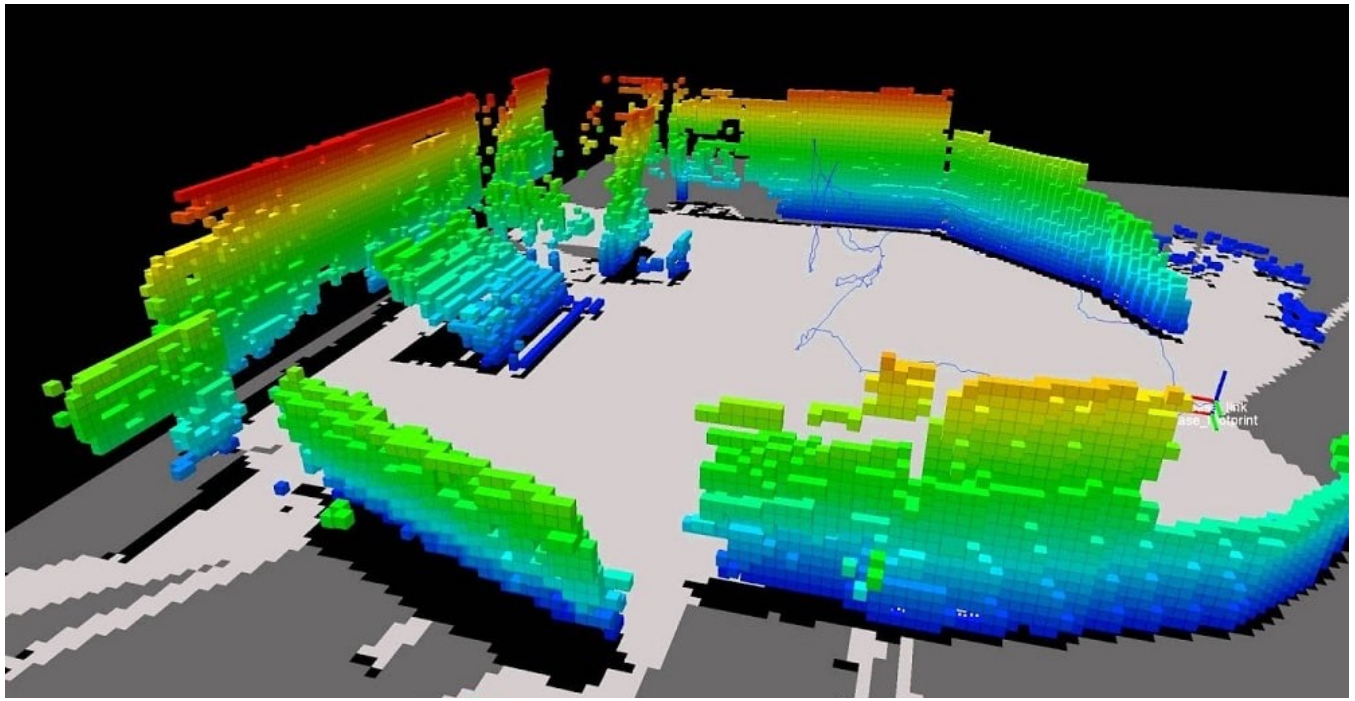
Autonomous Robotics

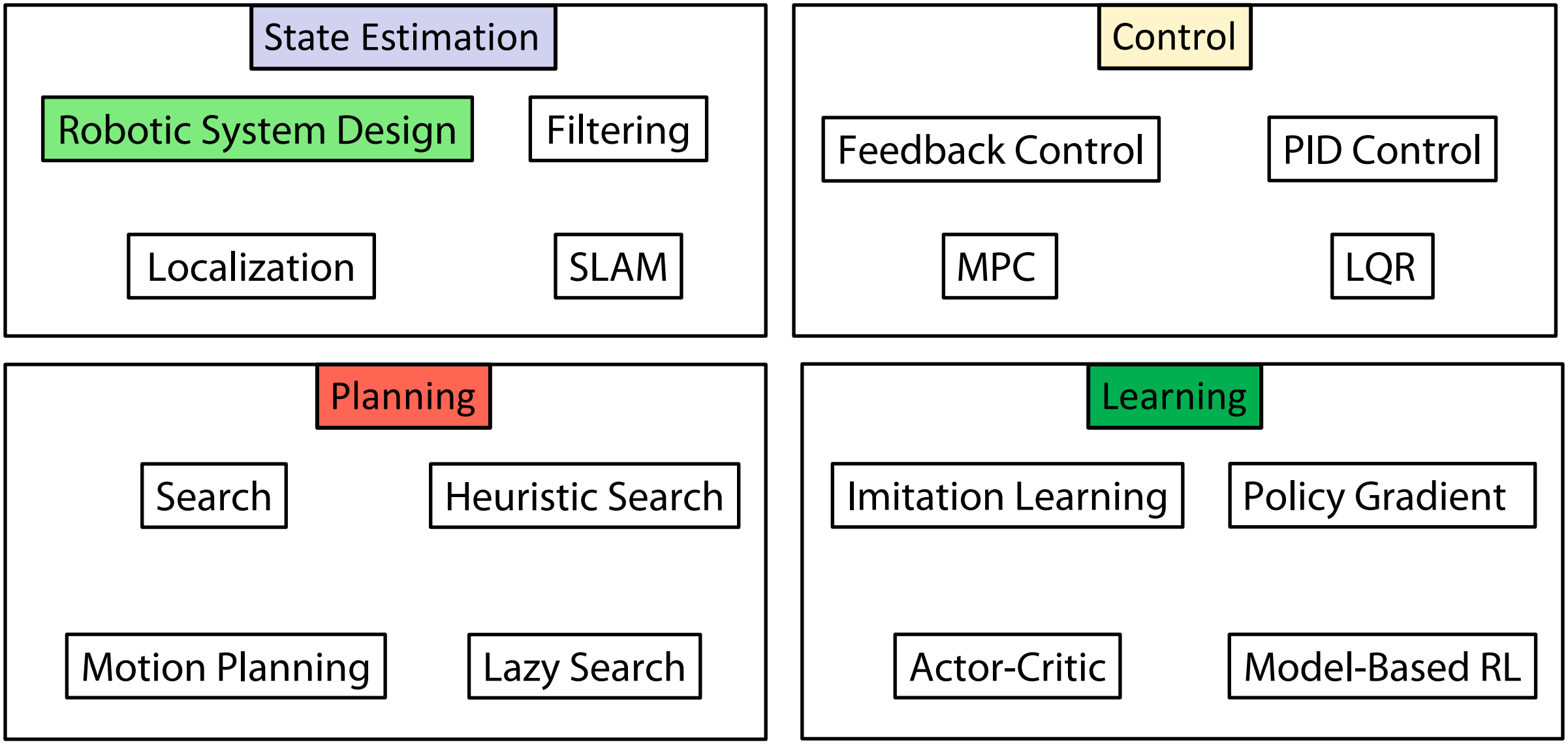
Winter 2025

Abhishek Gupta

TAs: Carolina Higuera, Entong Su, Bernie Zhu



Class Outline



Logistics

- Seeded paper discussion groups sent out, will be on Friday.

- Post questions, discuss any issues you are having on Ed.
- Students with **no** access to 002, e-mail us with your student ID.
- Students that have not been added to the class, email abhgupta@cs.washington.edu with the subject-line "Waitlisted for CSE478"

Recap

Important Identities: Gaussians

Forward propagation

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \\ \epsilon \sim \mathcal{N}(0, Q) \end{cases} \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q)$$

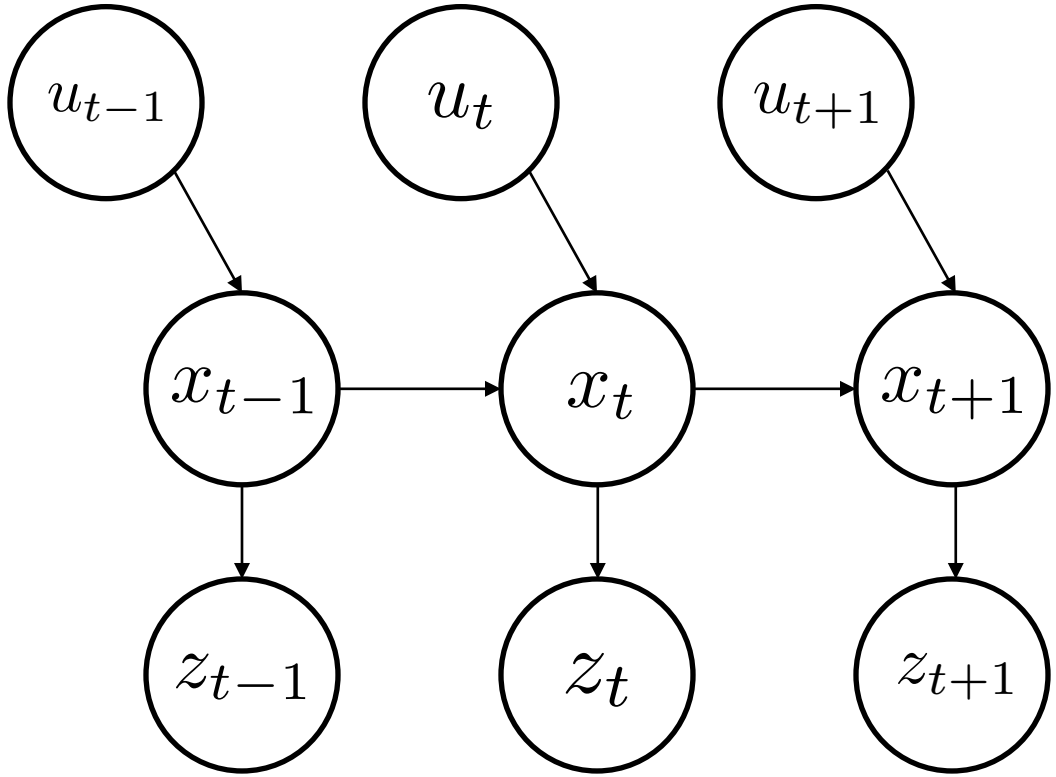
Conditioning

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma)$$

- Marginalization and conditioning in Gaussians results in Gaussians
- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations.

Discrete Kalman Filter

Kalman filter = Bayes filter with Linear Gaussian dynamics and sensor models



Discrete Kalman Filter: Matrix Version

Estimates the state ***x*** of a discrete-time controlled process that is governed by the linear stochastic difference equation

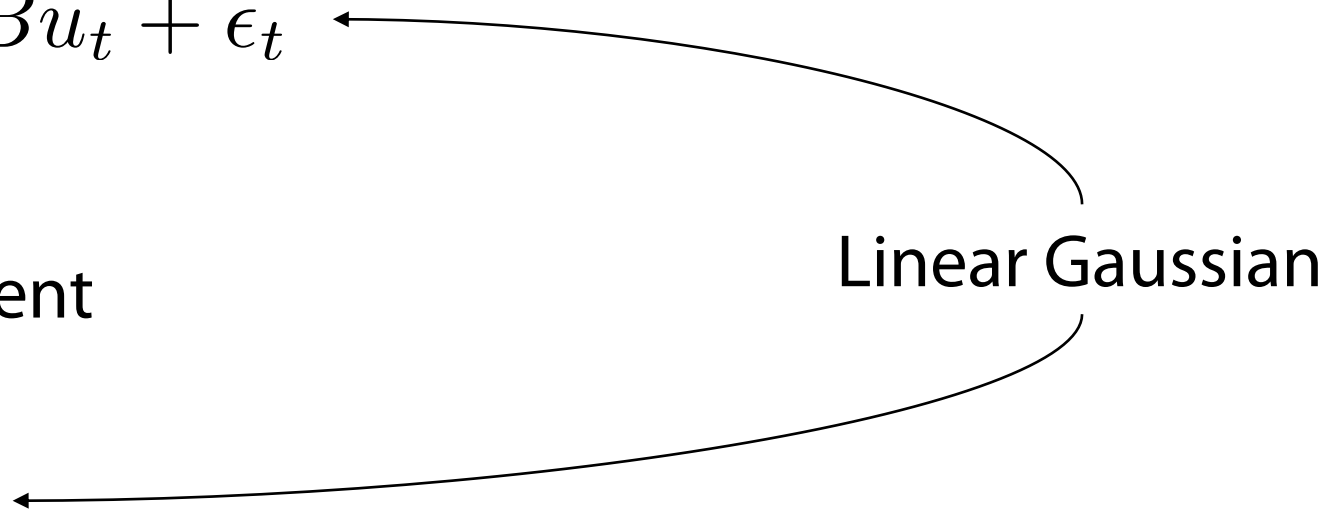
$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

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

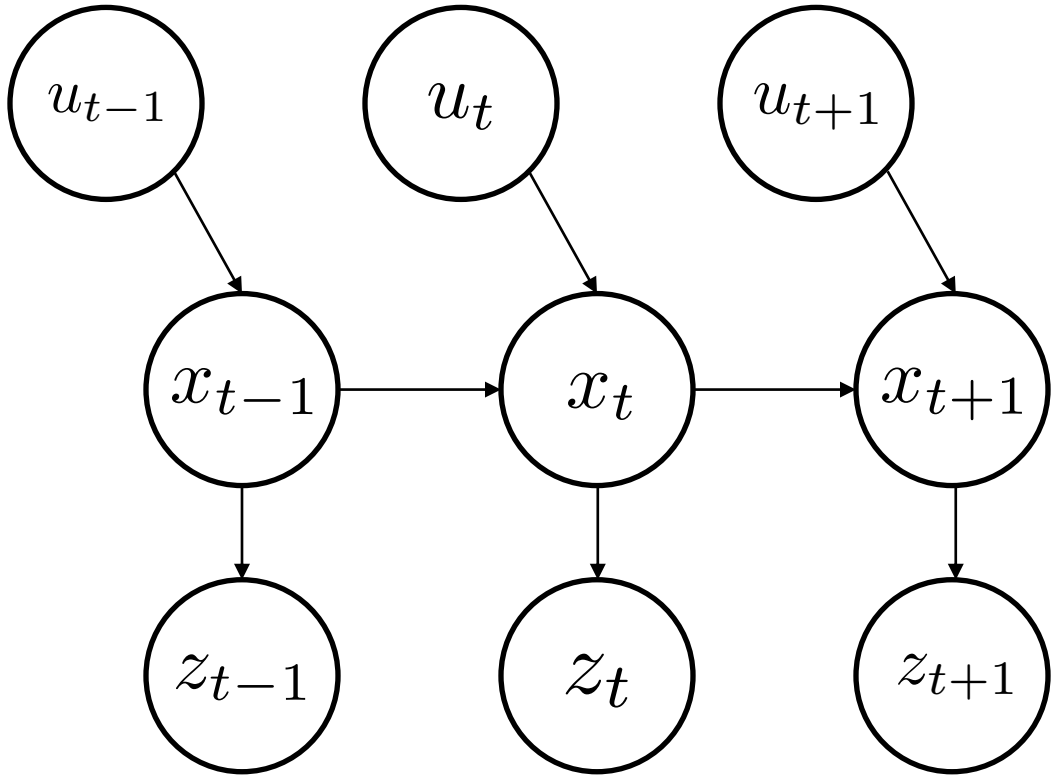
with a measurement

$$z_t = Cx_t + \delta_t$$

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



Goal of the Kalman Filter: Same as Bayes Filter



Belief

$$p(x_t | z_{0:t}, u_{0:t})$$

Idea: recursive update

$$\propto p(z_t|x_t) \int p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{0:t-1}, u_{0:t-1})$$

Measurement

Dynamics

Recursive Belief

2 step process:

- Dynamics update (incorporate action)
- Measurement update (incorporate sensor reading)

Bayes Filters

Key Idea: Apply Markov to get a recursive update!

Step 0. Start with the belief at time step t-1

$$bel(x_{t-1})$$

Step 1: Prediction - push belief through dynamics given action

$$\overline{bel}(x_t) = \sum P(x_t|u_t, x_{t-1})bel(x_{t-1})$$

Linear Gaussian

Step 2: Correction - apply Bayes rule given measurement

$$bel(x_t) = \eta P(z_t|x_t)\overline{bel}(x_t)$$

Linear Gaussian Systems: Initialization

- Initial belief is normally distributed:

$$Bel(x_0) = \mathcal{N}(\mu_0, \Sigma_0)$$

- $Bel(x_t)$ at any step t is: $\mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$
- $\overline{Bel}(x_t)$ at any step t is: $\mathcal{N}(\mu_{t|0:t-1}, \Sigma_{t|0:t-1})$

Linear Gaussian Systems: Prediction

- Integrate the effect of one action under the dynamics, before measurement comes in

$$x_{t+1} = Ax_t + Bu_{t+1} + \epsilon_{t+1} \qquad \epsilon_{t+1} \sim \mathcal{N}(0, Q_{t+1})$$

$$p(x_{t+1}|x_t, u_{t+1}) = \mathcal{N}(Ax_t + Bu_{t+1}, Q_{t+1})$$

$$\overline{Bel}(x_{t+1}) \qquad\qquad Bel(x_t)$$

$$p(x_{t+1}|z_{0:t}, u_{0:t+1}) = \int p(x_t|u_{0:t}, z_{0:t}) p(x_{t+1}|u_{t+1}, x_t) dx_t$$

Gaussian, easy!

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q) \\ \epsilon \sim \mathcal{N}(0, Q) \end{cases}$$

Linear Gaussian Systems: Prediction

- Integrate the effect of one action under the dynamics, before measurement comes in

$$p(x_t|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$$

$$x_{t+1} = Ax_t + Bu_{t+1} + \epsilon_{t+1}$$

$$\epsilon_{t+1} \sim \mathcal{N}(0, Q_{t+1})$$

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q) \\ \epsilon \sim \mathcal{N}(0, C) \end{cases}$$

Previous belief	$p(x_t\|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t\|0:t}, \Sigma_{t\|0:t})$
Belief Update	$p(x_{t+1}\|u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t\|0:t} + Bu_{t+1}, A\Sigma_{t\|0:t}A^T + Q_{t+1})$

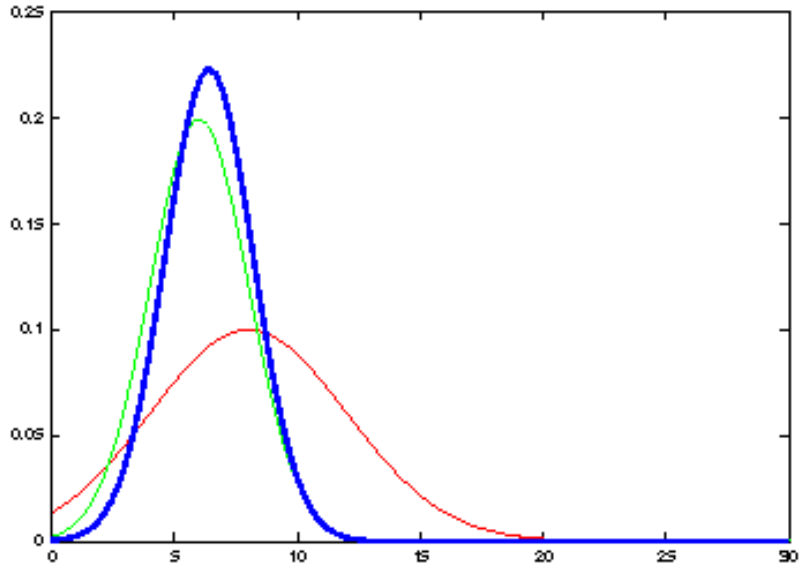
Intuition: Scale and shift the mean according to dynamics, uncertainty grows quadratically!

Intuition Behind Prediction Step

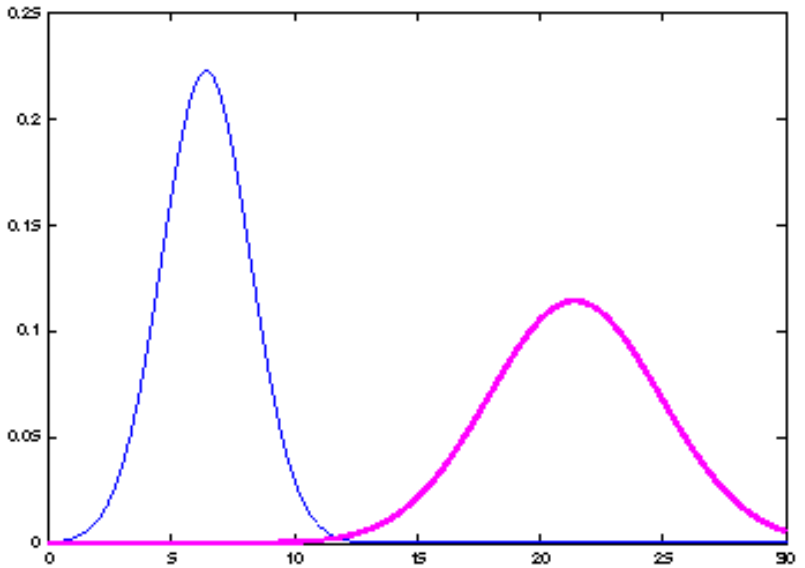
Previous belief $$p(x_t|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$$

Belief Update $$p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t|0:t} + Bu_{t+1}, A\Sigma_{t|0:t}A^T + Q_{t+1})$$

Intuition: Scale and shift the mean according to dynamics, uncertainty grows!



Belief at x_t



Belief post dynamics → shifted mean, scaled and shifted variance

Linear Gaussian Systems: Observations

- Integrate the effect of an observation using sensor model, after dynamics

$$z_{t+1} = Cx_{t+1} + \delta_{t+1} \qquad \delta_{t+1} \sim \mathcal{N}(0, R_{t+1})$$

$$p(z_{t+1}|x_{t+1}) = \mathcal{N}(Cx_{t+1}, R_{t+1})$$

$$p(x_{t+1}|u_{0:t+1}, z_{0:t+1}) \propto p(z_{t+1}|x_{t+1}) p(x_{t+1}|u_{0:t+1}, z_{0:t}) \qquad Bel(x_{t+1}) \qquad \overline{Bel}(x_{t+1})$$

Conditioning

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma)$$

$$K = \Sigma C^T (C\Sigma C^T + R)^{-1}$$

Linear Gaussian Systems: Observations

- Integrate the effect of an observation using sensor model, after dynamics

$$p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$$
$$z_{t+1} = Cx_{t+1} + \delta_{t+1}$$
$$\delta_{t+1} \sim \mathcal{N}(0, R_{t+1})$$

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma)$$
$$K = \Sigma C^T (C\Sigma C^T + R)^{-1}$$

Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$

$$K_{t+1} = \Sigma_{t+1|0:t} C^T (C\Sigma_{t+1|0:t} C^T + R)^{-1}$$

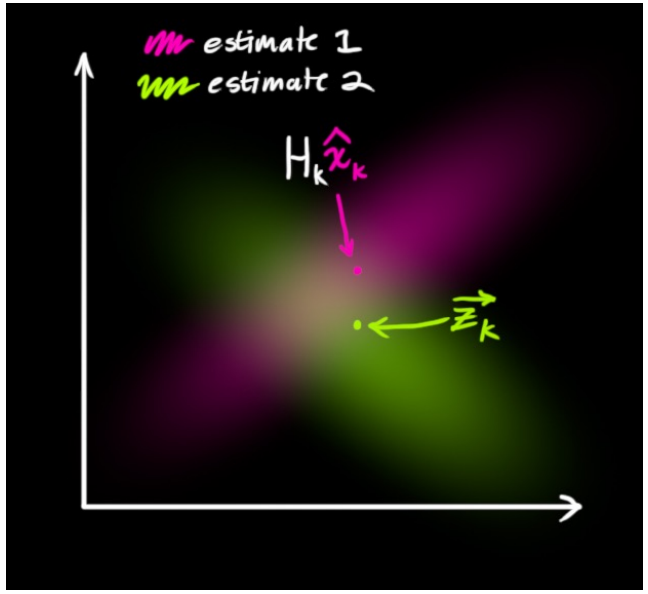
Linear Gaussian Systems: Observations

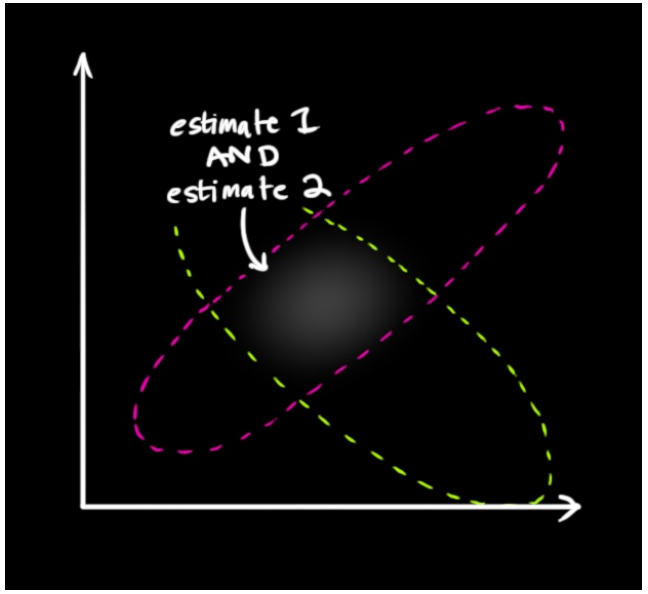
Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

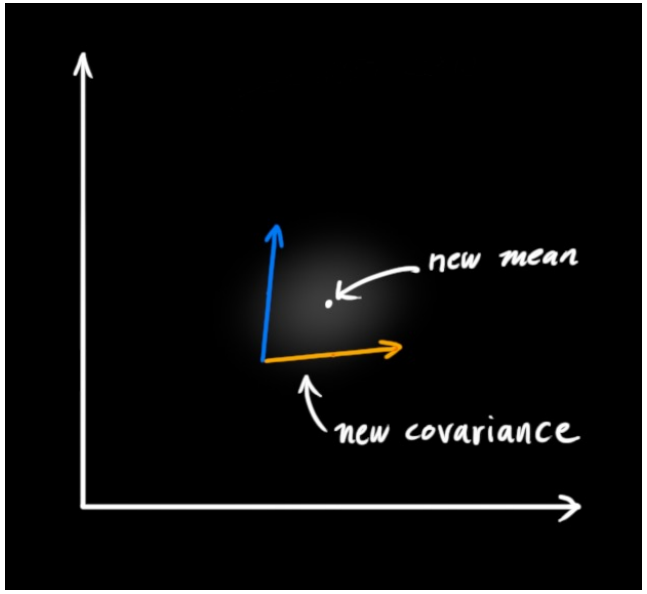
Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$

Intuition: Correct the update linearly according to measurement error from expectation, shrink uncertainty accordingly



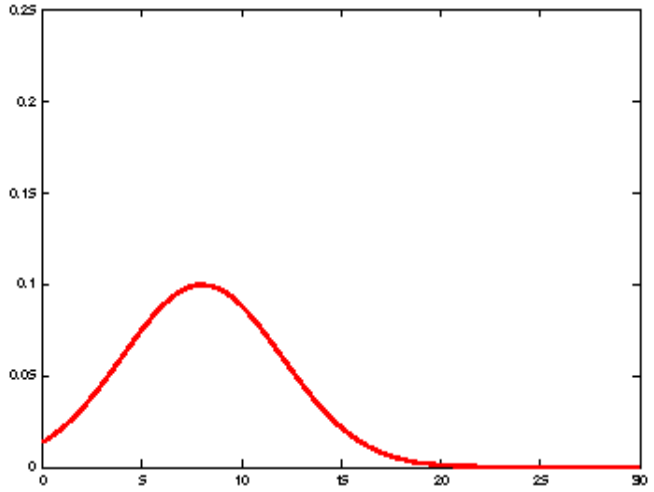


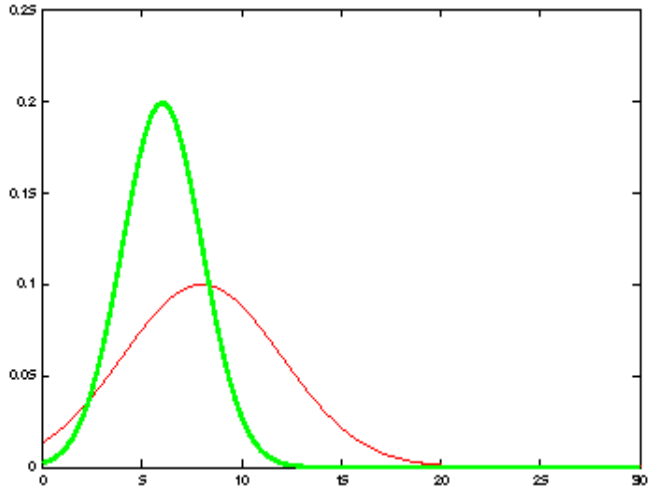


Intuition Behind Correction Step

Previous belief

New Measurement

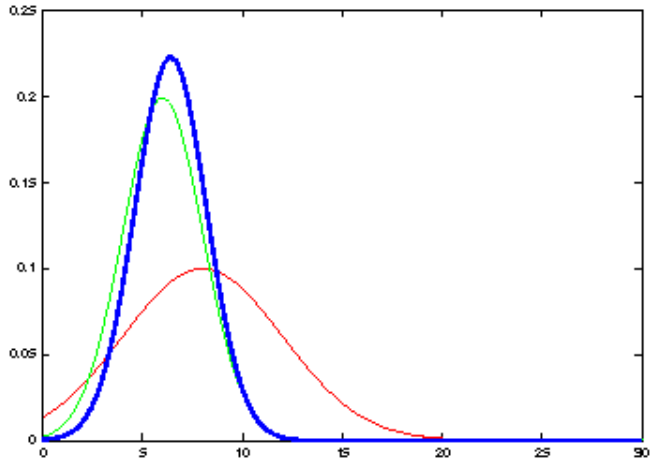




$$p(x_{t+1}|u_{0:t+1}, z_{0:t+1}) = \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$
$$K_{t+1} = \Sigma_{t+1|0:t}C^T(C\Sigma_{t+1|0:t}C^T + R)^{-1}$$

For the sake of simplicity, let's say C = I

$$K_{t+1} = \frac{\Sigma_{t+1|0:t}}{\Sigma_{t+1|0:t} + R}$$



Corrects belief based on measurement

→ Average between mean and measurement based on K

→ Scale down uncertainty based on K

Unpacking the Kalman Gain

Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$

$$K_{t+1} = \Sigma_{t+1|0:t}C^T(C\Sigma_{t+1|0:t}C^T + R)^{-1}$$

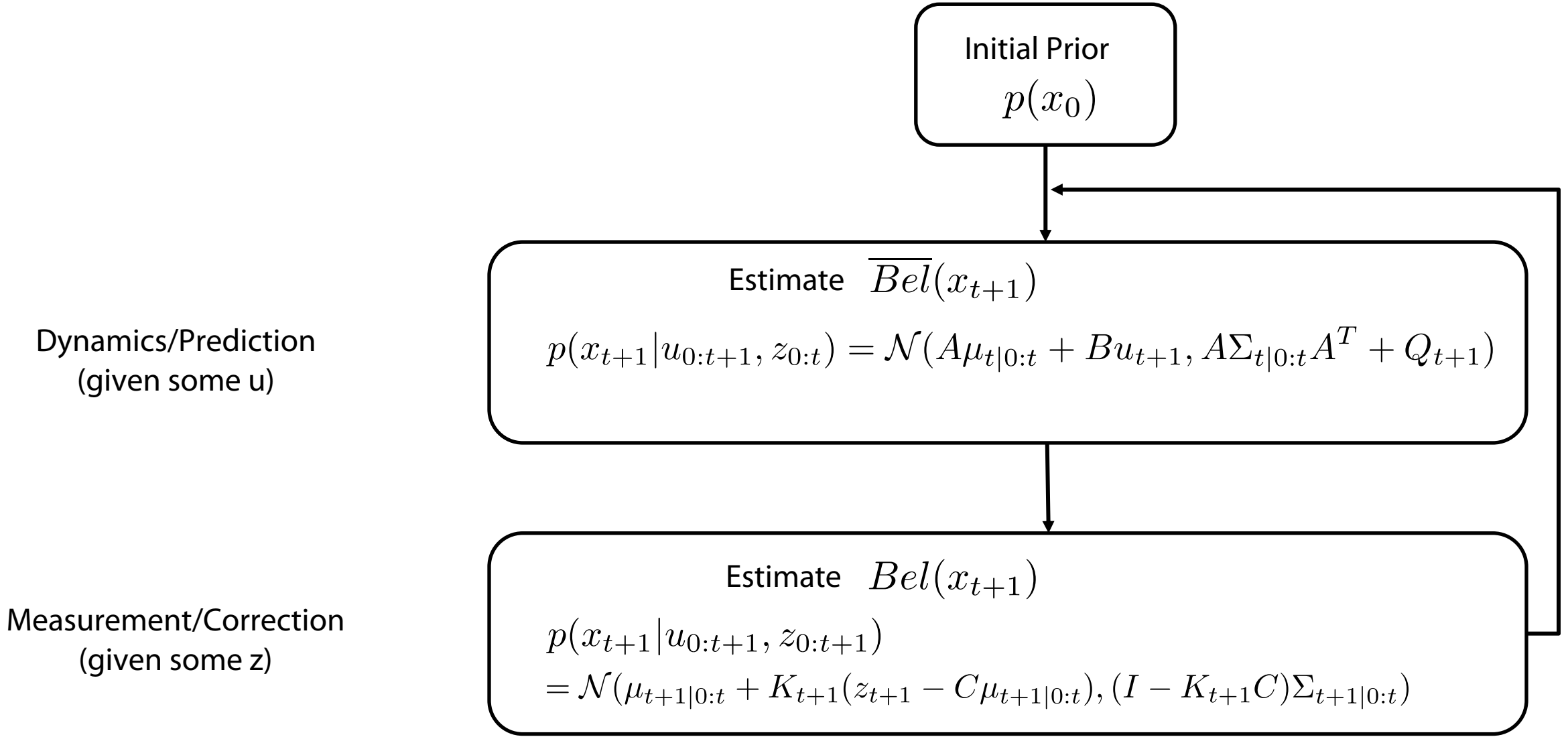
Case 1: Very noisy sensor, R>>Σ

For the sake of simplicity, let's say C = I

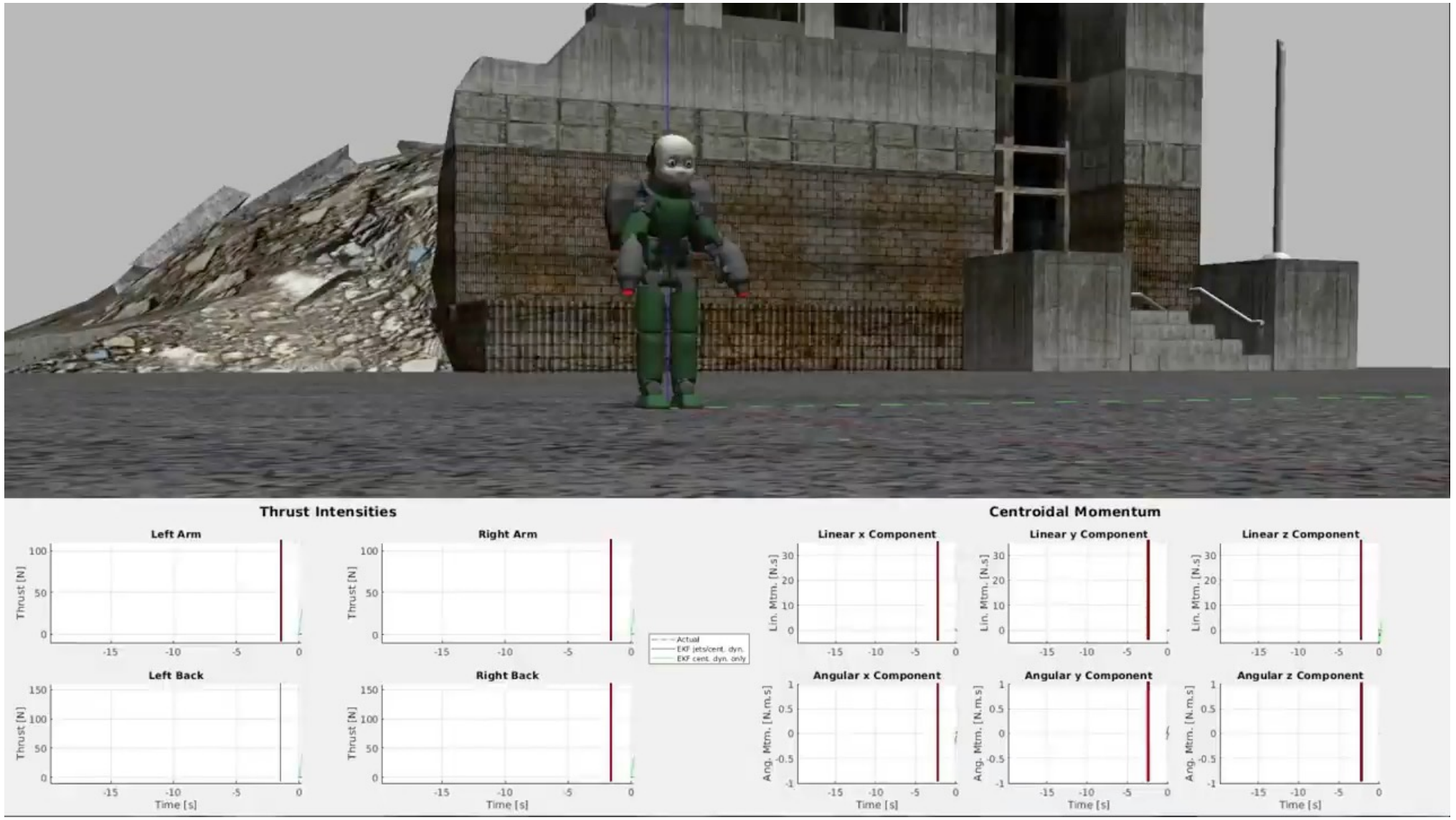
$$K_{t+1} = \frac{\Sigma_{t+1|0:t}}{\Sigma_{t+1|0:t} + R}$$

Case 2: Deterministic sensor, R = 0

Kalman Filter Algorithm



Kalman Filter in Action



Kalman Filter Summary

- Highly efficient: Polynomial in measurement dimensionality ***k*** and state dimensionality ***n***:

  $O(k^{2.376} + n^2)$

Matrix Inversion (Correction)

$$K_{t+1} = \Sigma_{t+1|0:t} C^T (C \Sigma_{t+1|0:t} C^T + R_{t+1})^{-1}$$

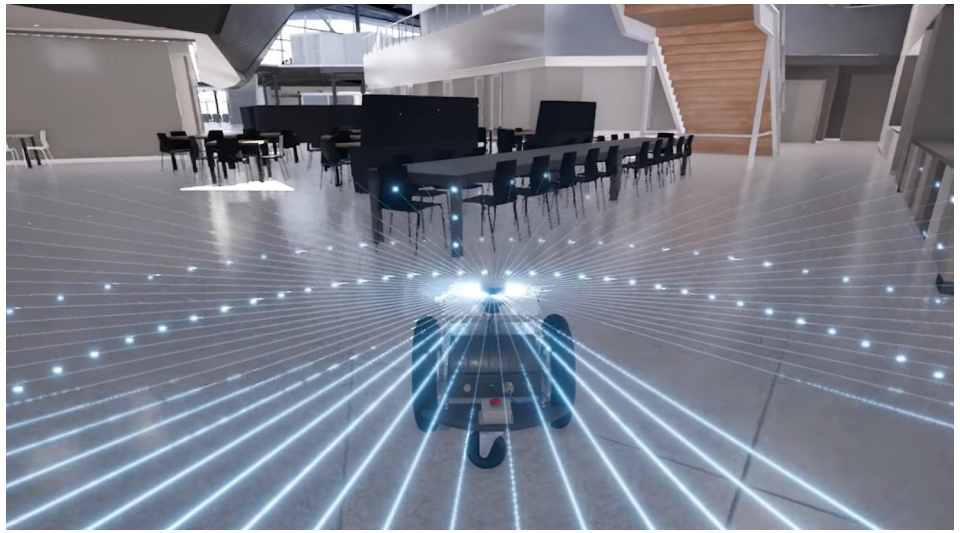
Matrix Multiplication (Prediction)

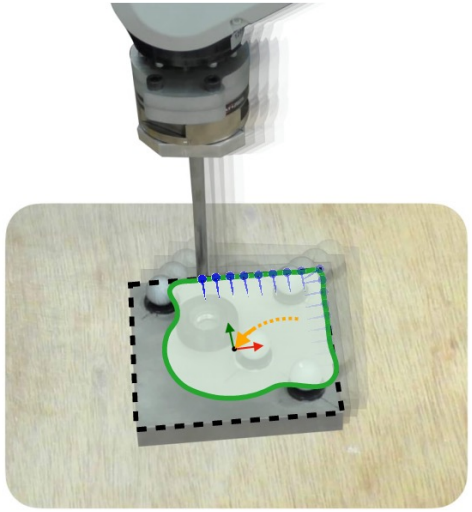
$$p(x_{t+1}|z_{0:t}, u_{0:t+1}) \sim \mathcal{N}(A\mu_{t|0:t} + Bu_t, A\Sigma_{t|0:t}A^T + Q_t)$$

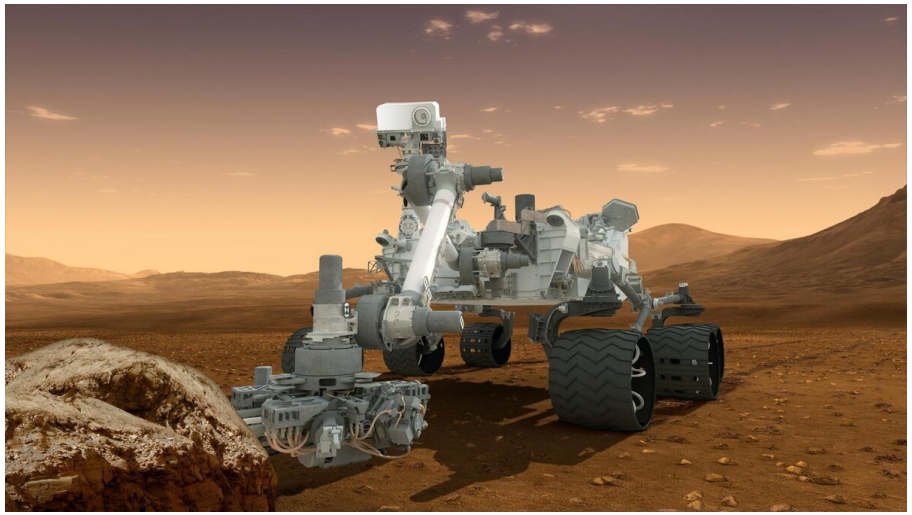
- Optimal for linear Gaussian systems!
- Most robotics systems are nonlinear!

Why should we care?

Still a very widely used technique for estimation/localization/mapping in real problems







Lecture Outline

Kalman Filtering

Extended Kalman Filtering

SLAM as a filtering problem

Can I reuse Kalman Filter math for non-linear systems?

Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

$$x_{t+1} = g(x_t, u_t) + \epsilon_t$$

$$z_t = h(x_t) + \delta_t$$

Non-linear system

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

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

Additive Gaussian noise

More reasonable assumption than linear Gaussian. More on non-Gaussian systems next time

How do we deal with non-linearity?

- Differentiable non-linear functions can be expressed via their Taylor expansion

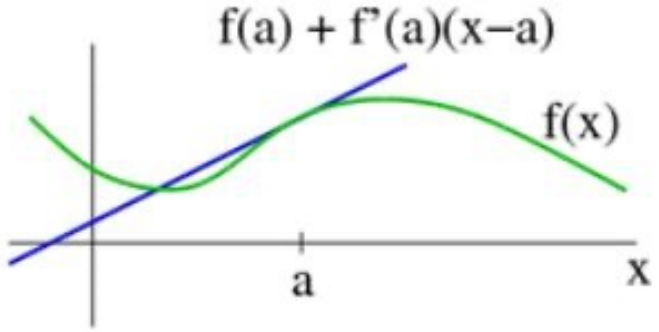
$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots,$$

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a)$$

Dropping higher order terms, when x-a is small enough

Linear function in x

Pretend that your function is linear in this neighborhood
→ Reapprox in a new neighborhood



EKF Linearization: First Order Taylor Series Expansion

- Idea behind EKF: Linearize the dynamics and measurement around current μ_t
- Dynamics Model (linearize around previous belief):

$$x_{t+1} = g(x_t, u_t) + \epsilon_t \approx g(\mu_t, u_t) + \left.\frac{\partial g(x_t, u_t)}{\partial x_t}\right|_{x_t=\mu_t} (x_t - \mu_t) + \epsilon_t$$

$$= g(\mu_t, u_t) + G(x_t - \mu_t) + \epsilon_t$$

- Measurement Model (linearize around post dynamics belief):

$$z_t = h(x_t) + \delta_t \approx h(\bar{\mu}_t) + \left.\frac{\partial h(x_t)}{\partial x_t}\right|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t) + \delta_t \approx h(\bar{\mu}_t) + H(x_t - \bar{\mu}_t) + \delta_t$$

Now everything is linear → back to Kalman filtering!

Modified System under EKF Linearization

- Start by linearizing dynamics model under current belief
- Dynamics Model (linearize around previous belief):

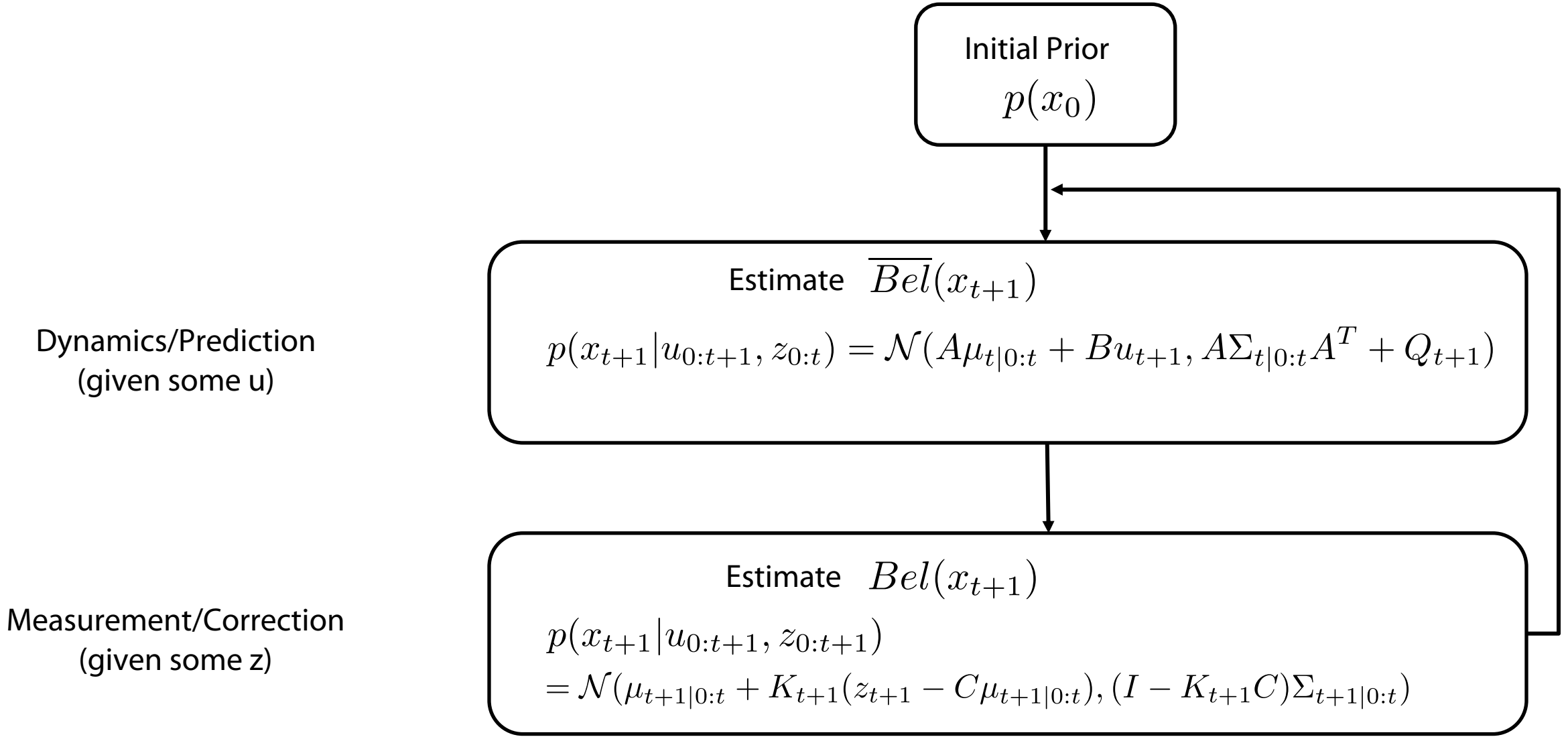
$$x_{t+1} = g(x_t, u_t) + \epsilon_t \qquad \approx g(\mu_t, u_t) + \left.\frac{\partial g(x_t, u_t)}{\partial x_t}\right|_{x_t=\mu_t} (x_t - \mu_t) + \epsilon_t$$

- Perform dynamics update
- Linearize measurement around post dynamics belief

$$z_t = h(x_t) + \delta_t \approx h(\bar{\mu}_t) + \left.\frac{\partial h(x_t)}{\partial x_t}\right|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t) + \delta_t \approx h(\bar{\mu}_t) + H(x_t - \bar{\mu}_t) + \delta_t$$

- Perform measurement update
- Repeat

Original Kalman Filter Algorithm



EKF Algorithm – linearize non-linear functions

Initial Prior

$$p(x_0)$$

Linearize dynamics

$$x_{t+1} = g(x_t, u_t) + \epsilon_t \approx g(\mu_t, u_t) + \left.\frac{\partial g(x_t, u_t)}{\partial x_t}\right|_{x_t=\mu_t} (x_t - \mu_t) + \epsilon_t$$

Dynamics/Prediction
(given some u)

Estimate $\overline{Bel}(x_t)$

$$p(x_{t+1}|z_{0:t}, u_{0:t}) \sim \mathcal{N}(g(\mu_t, u_t), G\Sigma_{t|0:t}G^T + Q_t)$$

Linearize measurement

$$z_t = h(x_t) + \delta_t \approx h(\bar{\mu}_t) + \left.\frac{\partial h(x_t)}{\partial x_t}\right|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t) + \delta_t$$

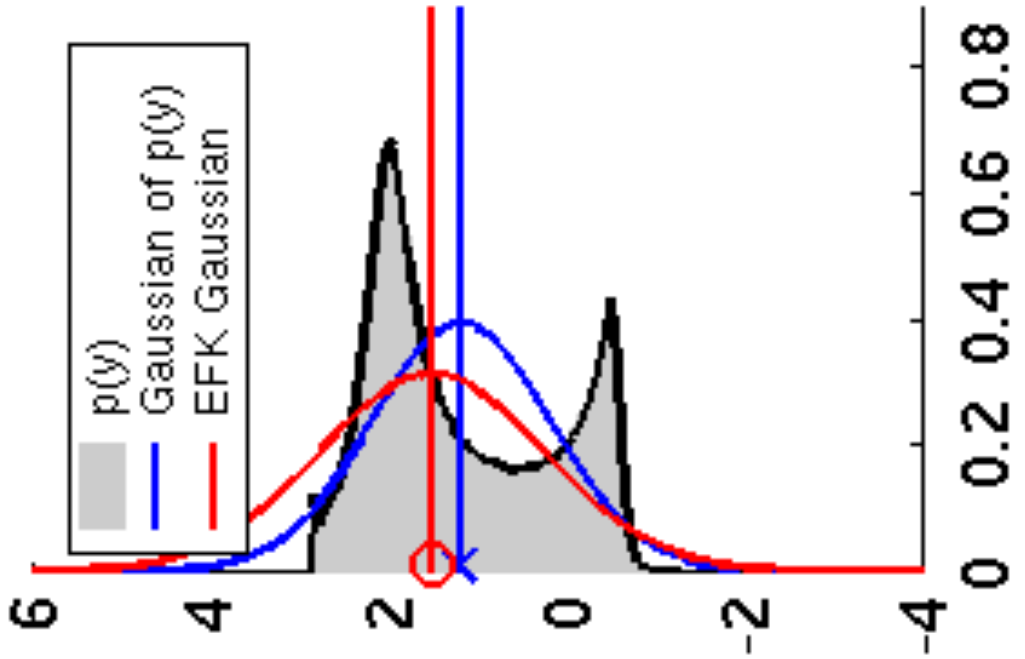
Measurement/Correction
(given some z)

Estimate $Bel(x_t)$

$$p(x_{t+1}|z_{0:t+1}, u_{0:t}) = \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - h(\bar{\mu}_t), (I - K_{t+1}H)\Sigma_{t+1|0:t})$$

Why might we still want to use particle filters?

- **Non-linear functions**
- **Non-Gaussian functions** ⟵ EKFs still require Gaussian Distributions



Ok so what have we learned

Bayesian Filtering!

Key Idea: Apply Markov to get a recursive update!

Step 0. Start with the belief at time step t-1

$$bel(x_{t-1})$$

Step 1: Prediction - push belief through dynamics given action

$$\overline{bel}(x_t) = \sum P(x_t|u_t, x_{t-1})bel(x_{t-1})$$

Step 2: Correction - apply Bayes rule given measurement

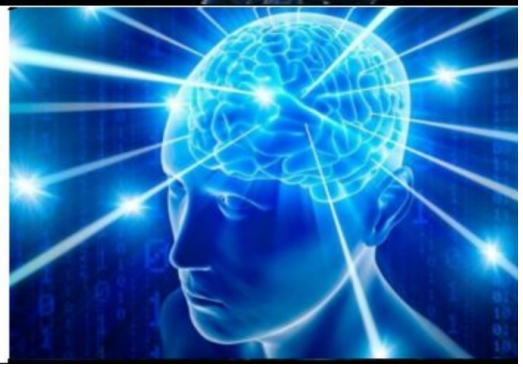
$$bel(x_t) = \eta P(z_t|x_t)\overline{bel}(x_t)$$

Motion and Measurement Model

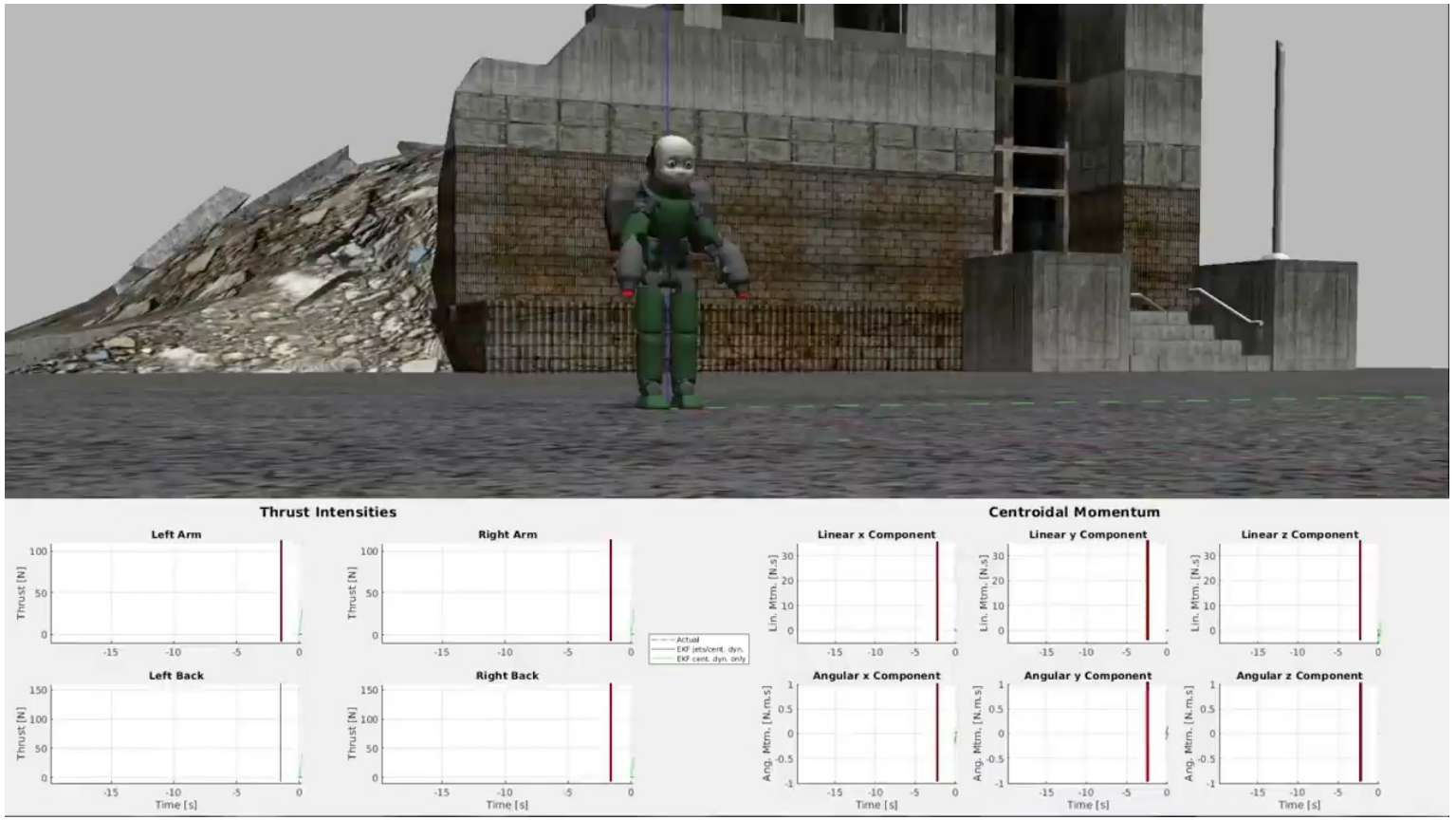
Linear Gaussian
– Kalman Filter

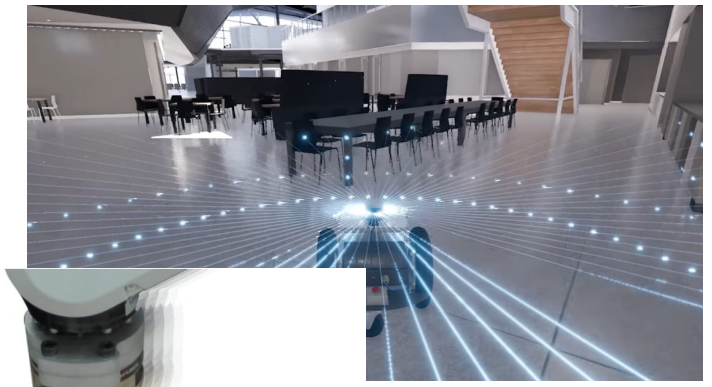
Nonlinear Gaussian
– Extended Kalman Filter

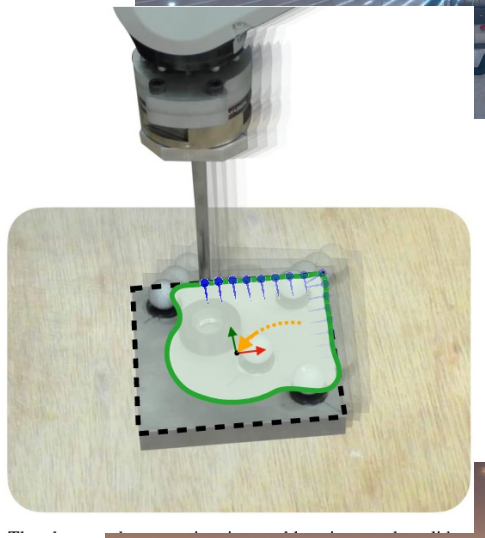
Nonlinear non-gaussian
– Particle Filter

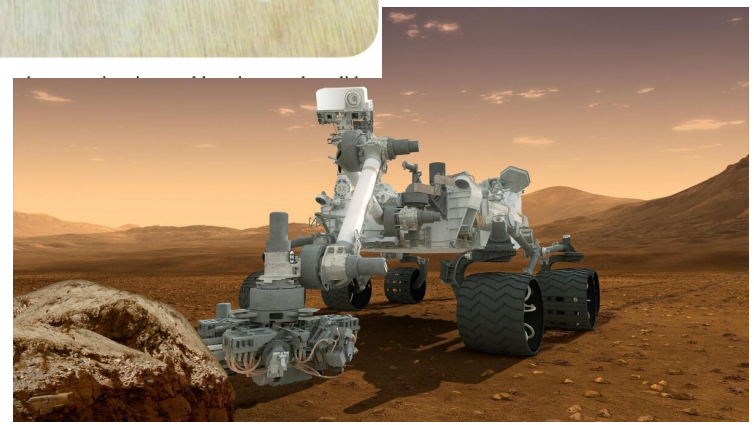


Why is this useful - Localization









Why is this useful - Localization

Lecture Outline

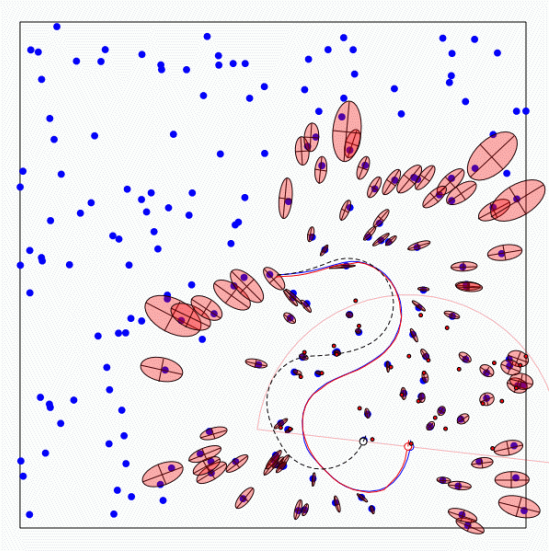
Kalman Filtering

Extended Kalman Filtering

SLAM as a filtering problem

Why is this useful - SLAM

- So far, the maps have been assumed to be known → often untrue → SLAM problem
- A robot is exploring an unknown, static environment.



Given:

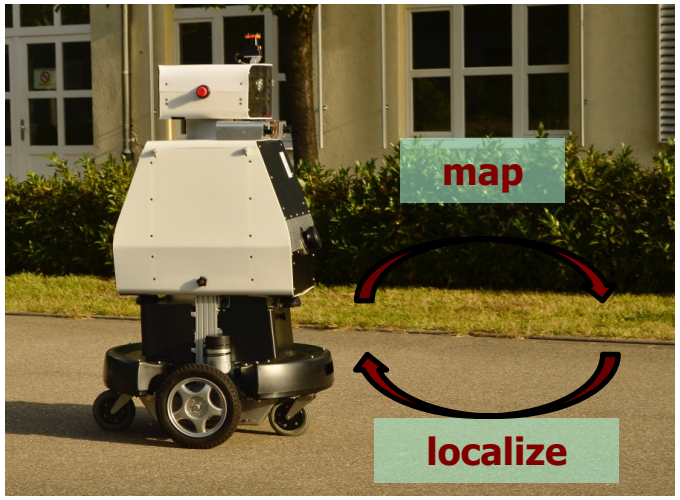
The robot's controls (u)

Observations of nearby features (z)

Estimate:

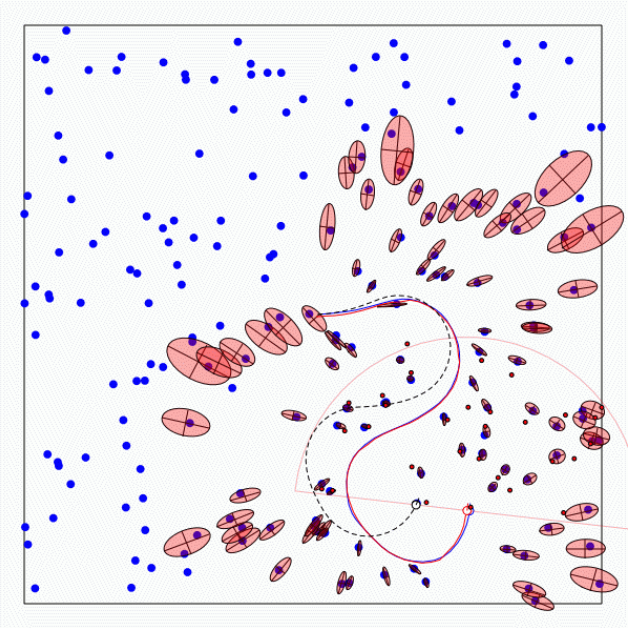
Map of features (x)

Path of the robot (x)



The SLAM Problem

A robot is exploring an unknown, static environment.



Given:

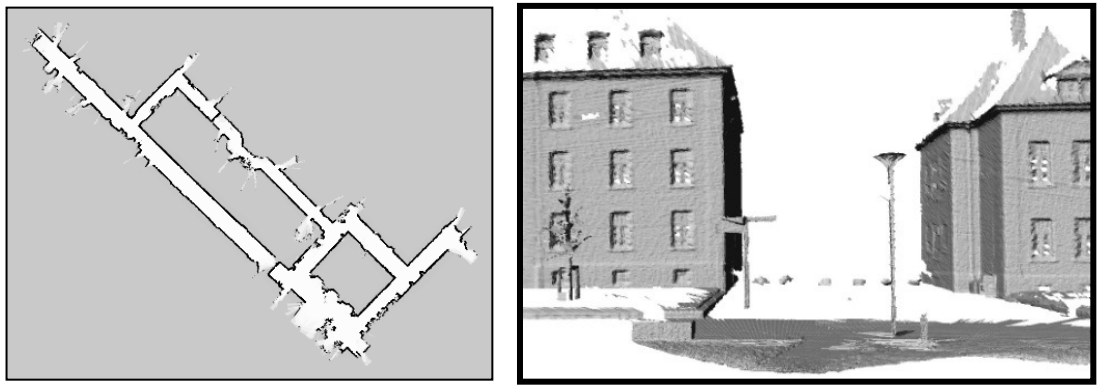
- The robot's controls
- Observations of nearby features

Estimate:

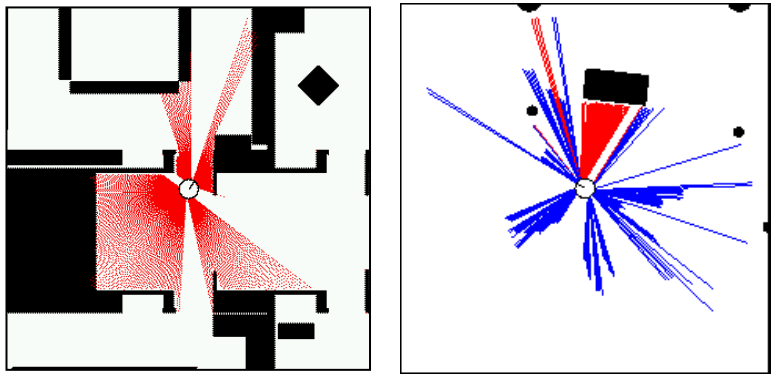
- Map of features
- Path of the robot

Why is SLAM difficult?

- Localization assumed map was perfectly known in the sensor/motion
- Mapping assumes position is fully known
- Doing both jointly is hard!

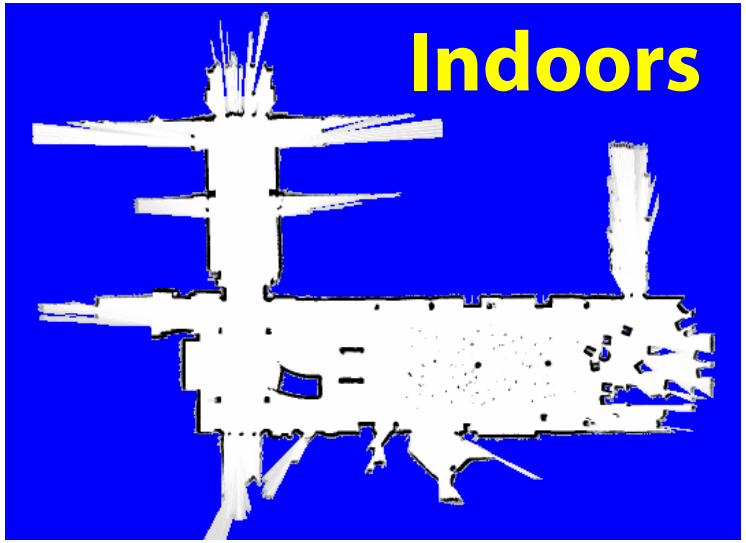


Mapping



Localization

SLAM Applications



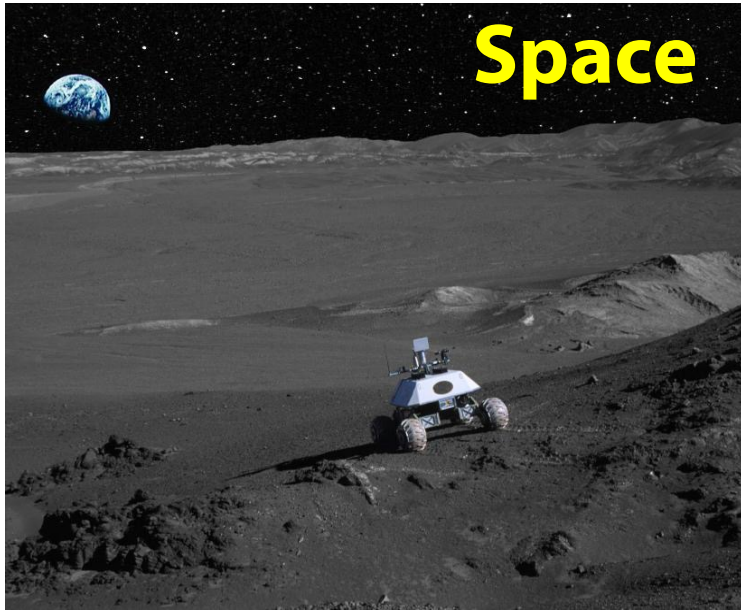
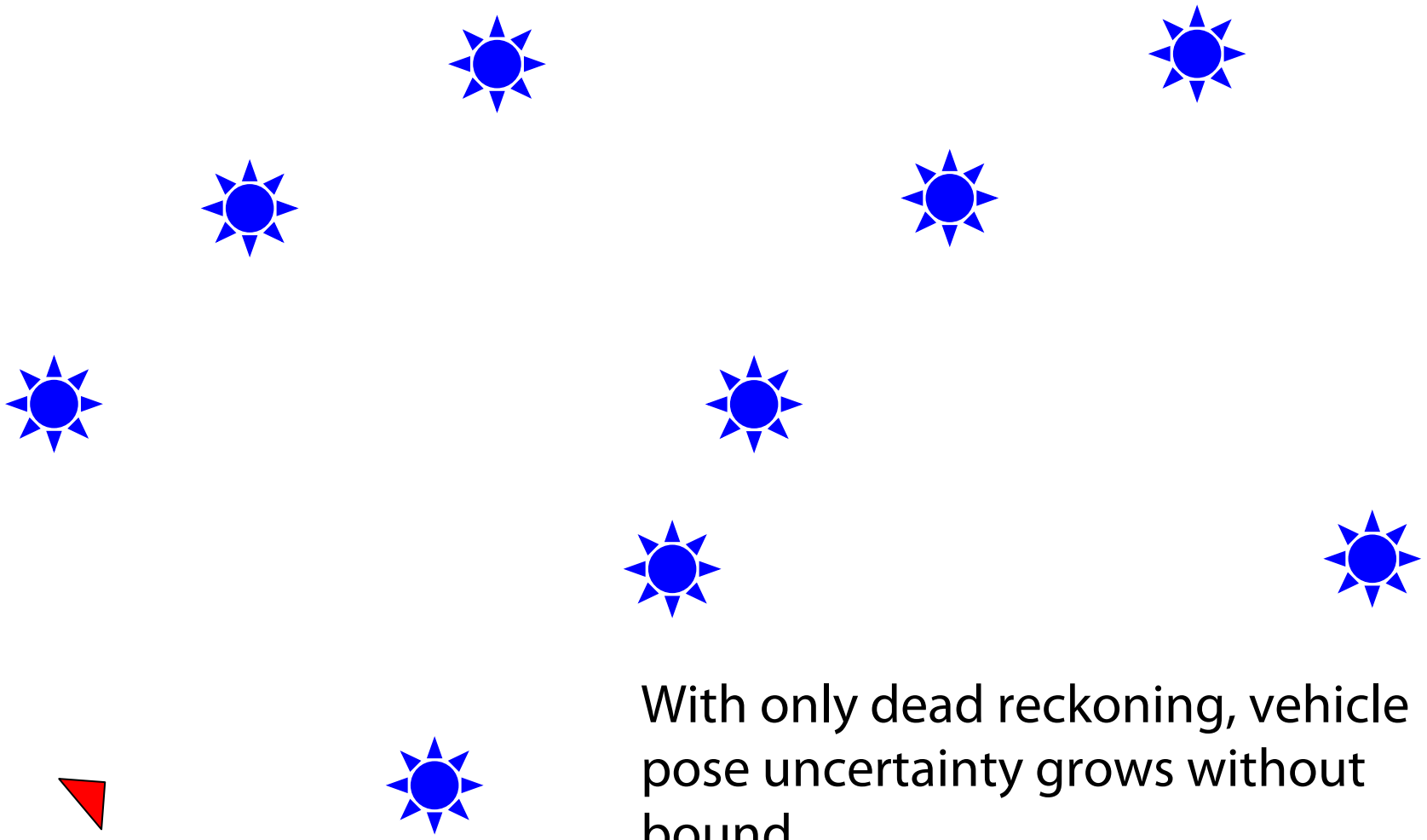


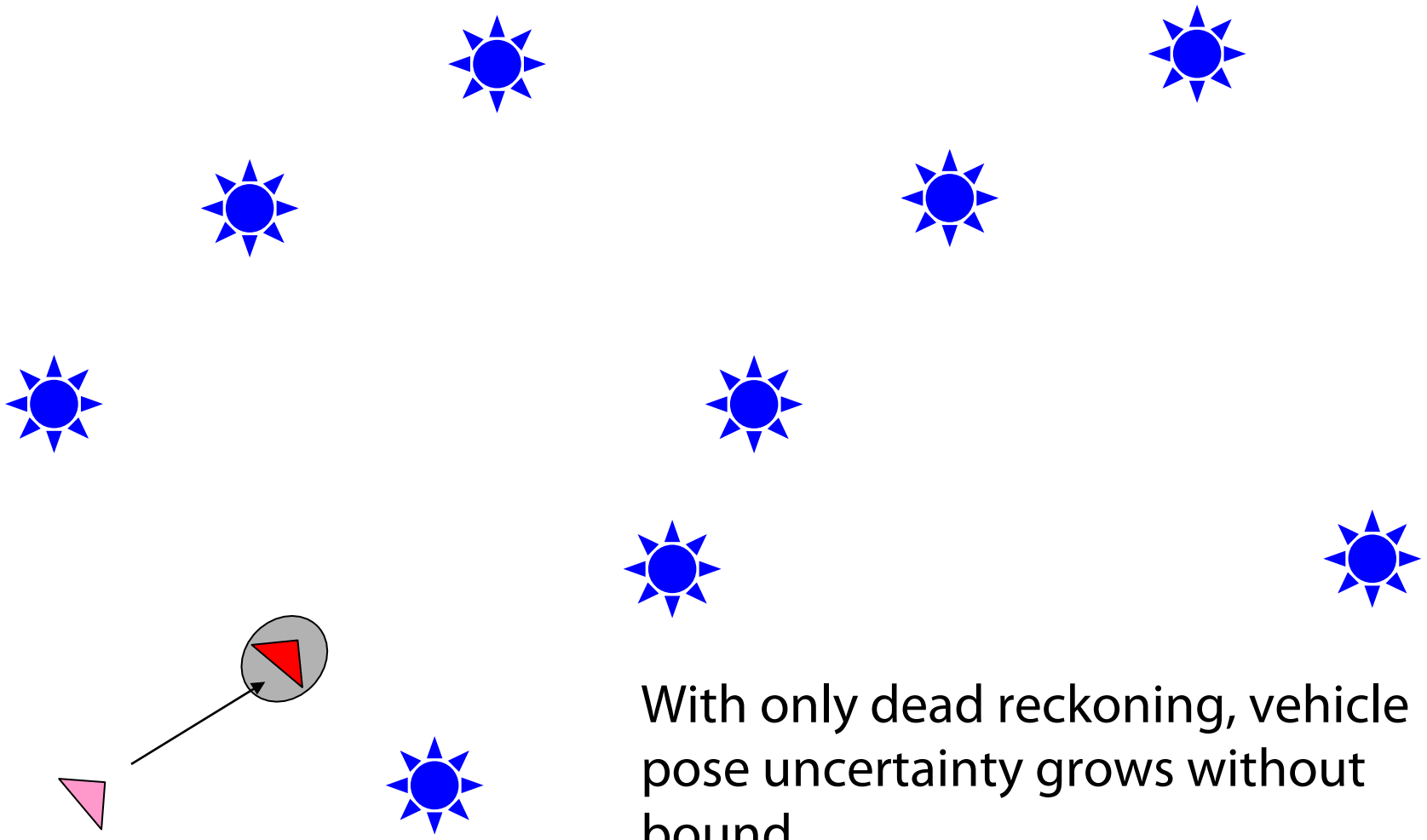
Illustration of SLAM without Landmarks



With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

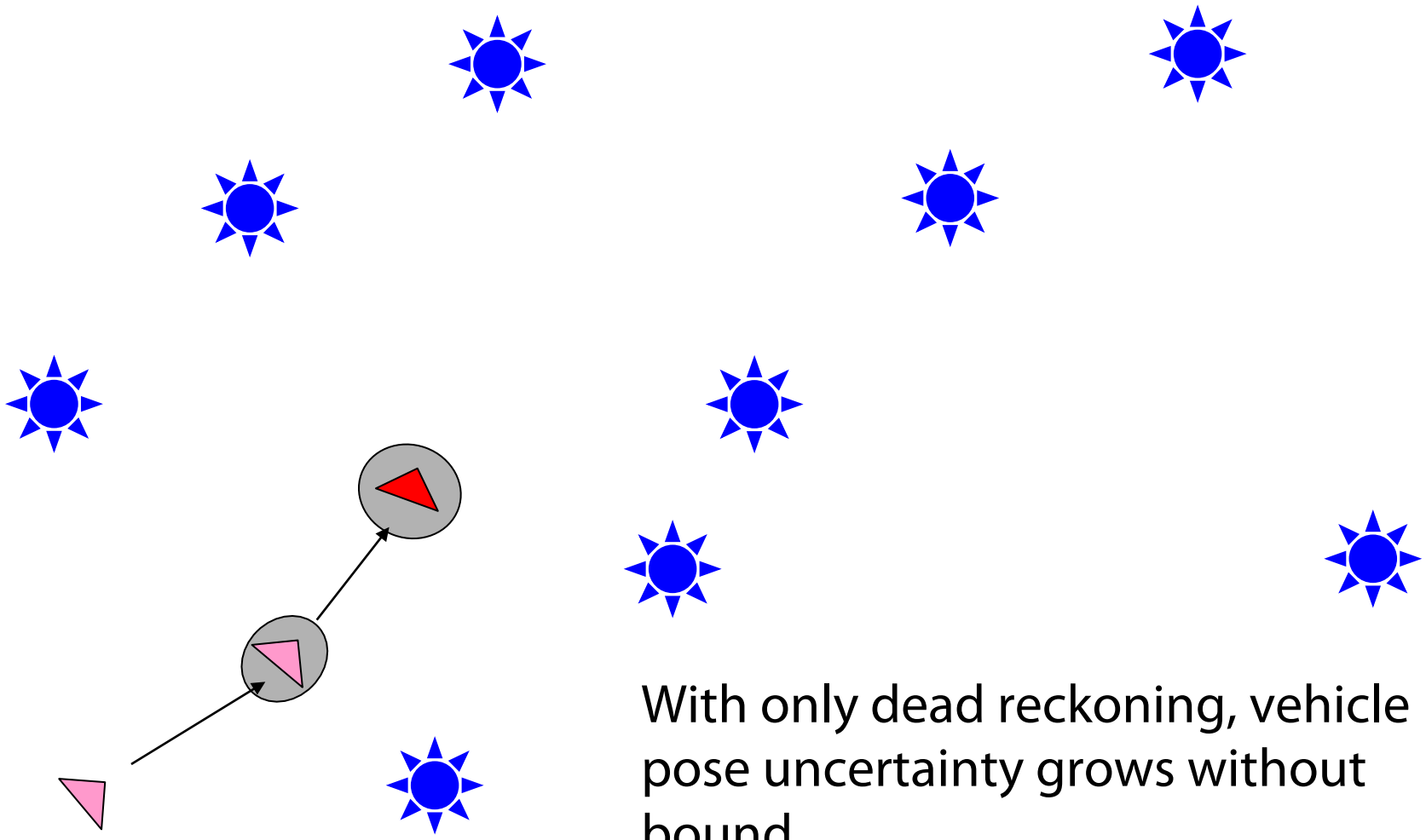
Illustration of SLAM without Landmarks



With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

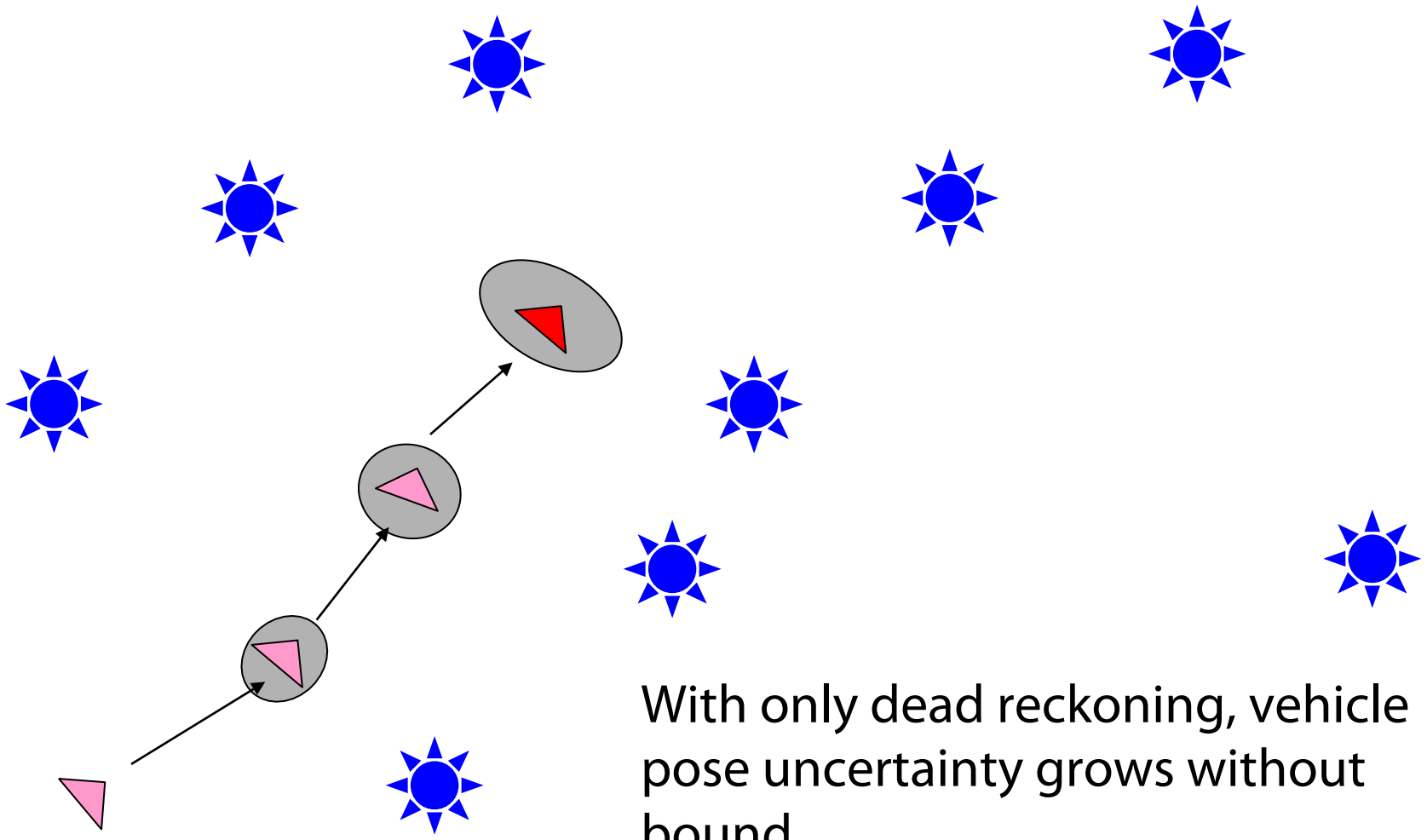
Illustration of SLAM without Landmarks



With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

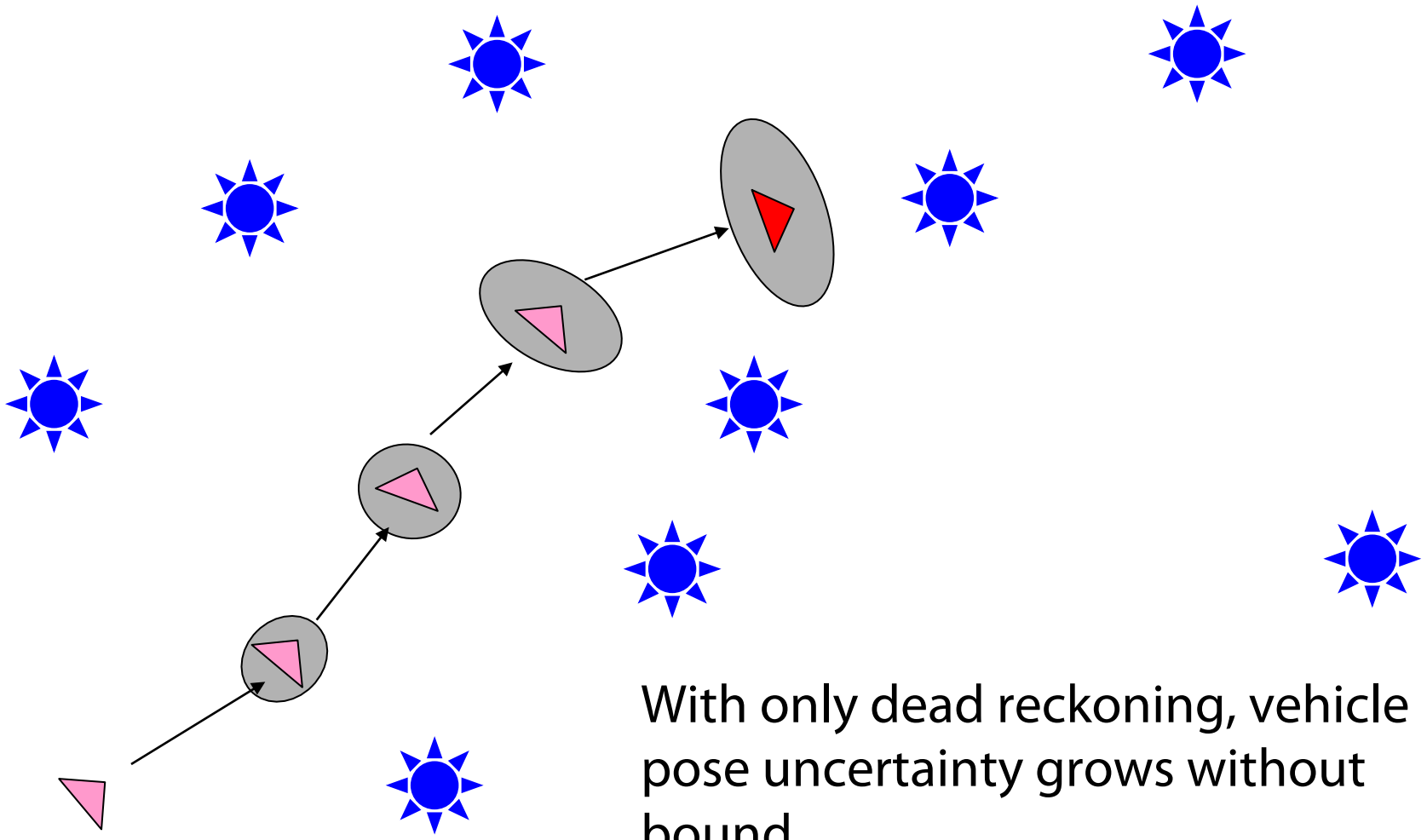
Illustration of SLAM without Landmarks



With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

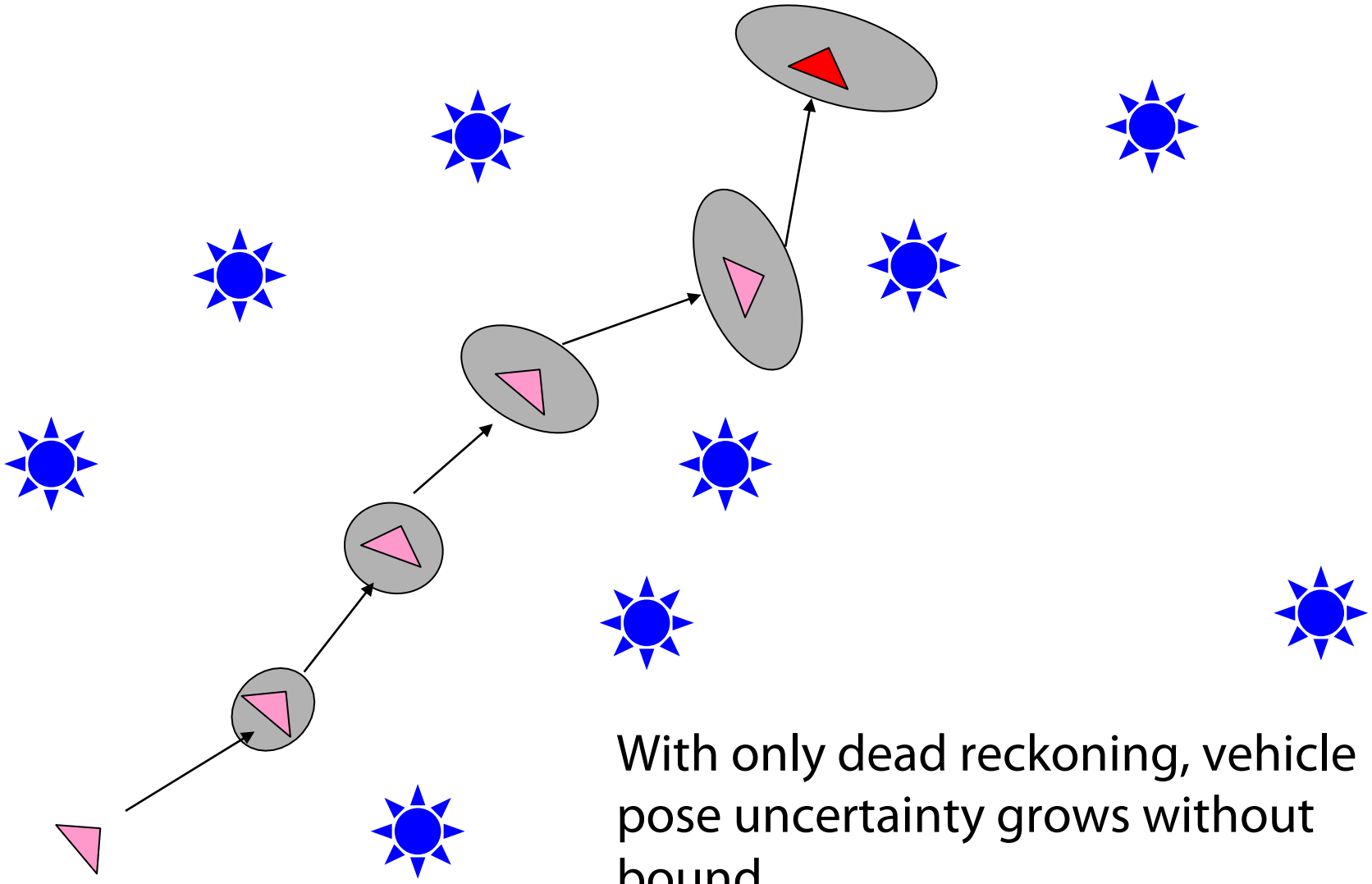
Illustration of SLAM without Landmarks



With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

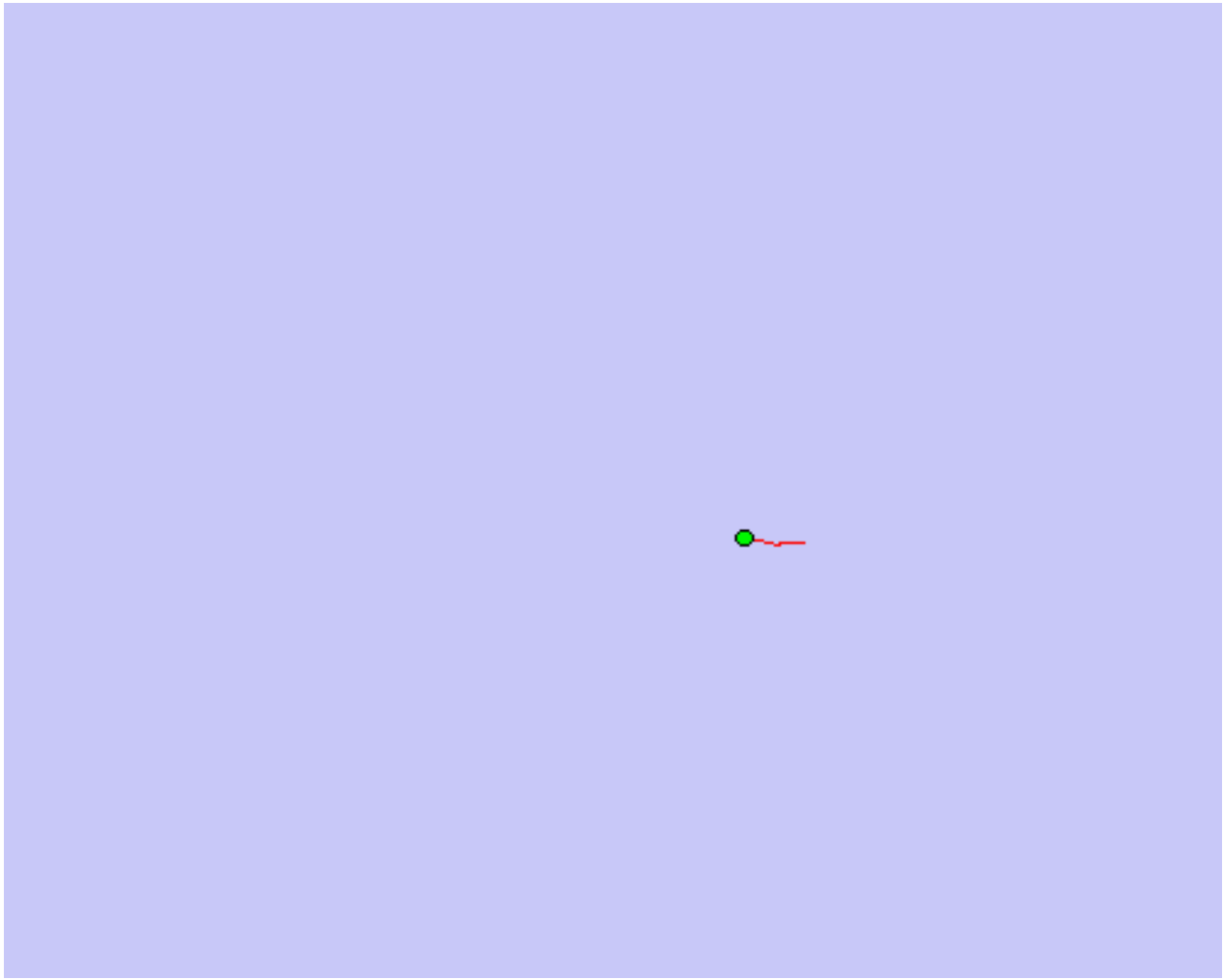
Illustration of SLAM without Landmarks



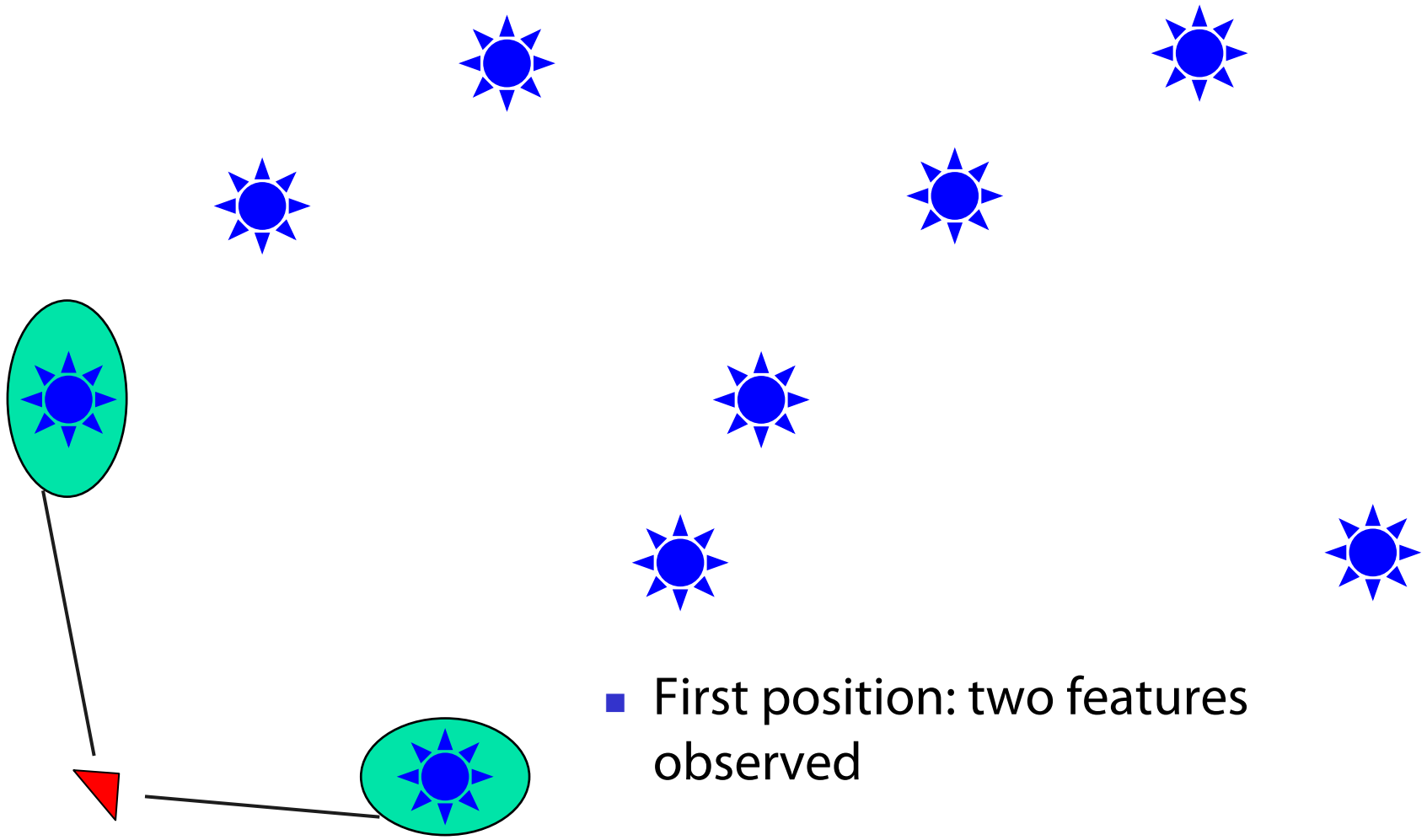
With only dead reckoning, vehicle pose uncertainty grows without bound

Courtesy J. Leonard

Mapping with Raw Odometry



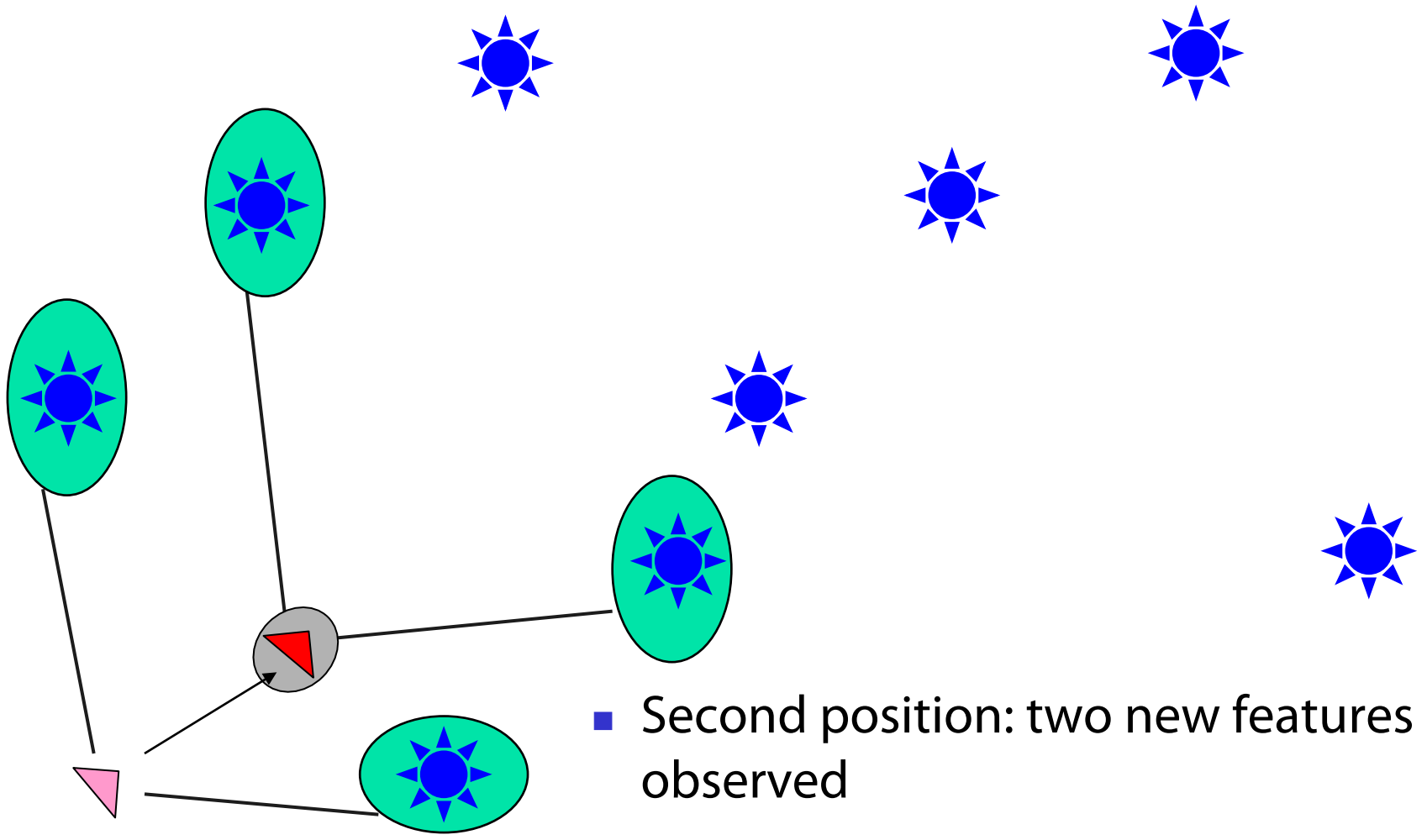
Repeat, with Measurements of Landmarks



- First position: two features observed

Courtesy J. Leonard

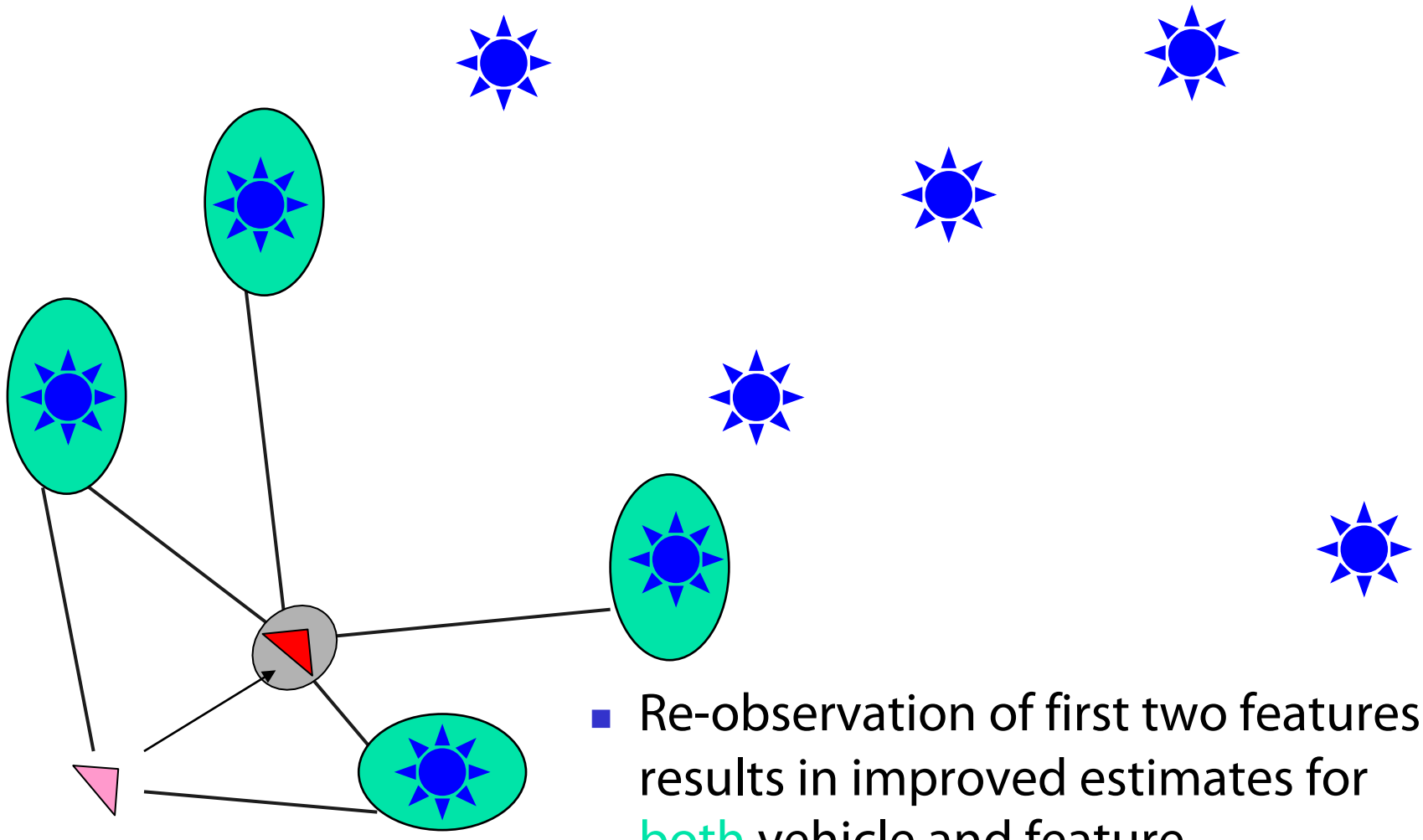
Illustration of SLAM with Landmarks



- Second position: two new features observed

Courtesy J. Leonard

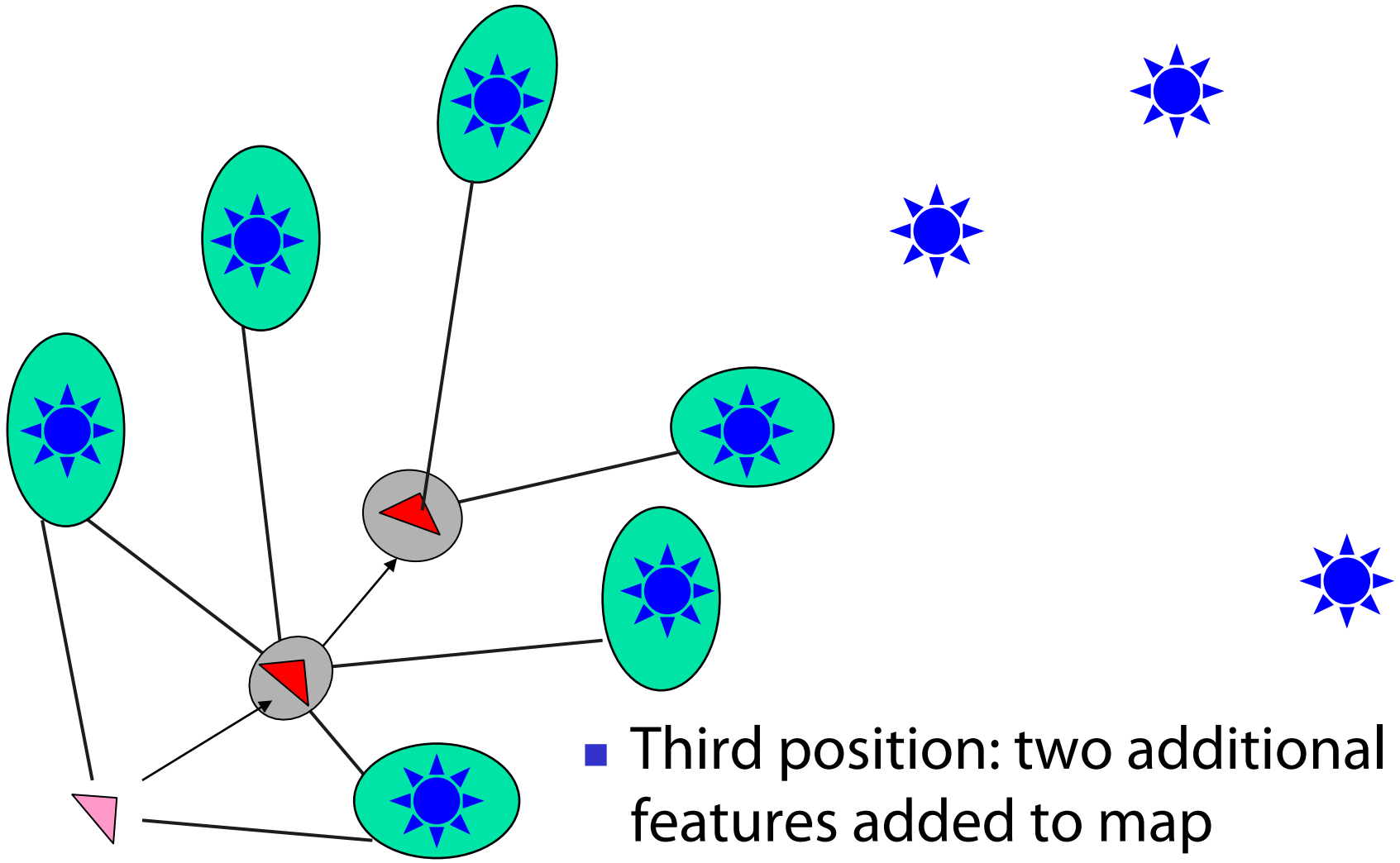
Illustration of SLAM with Landmarks



- Re-observation of first two features results in improved estimates for both vehicle and feature

Courtesy J. Leonard

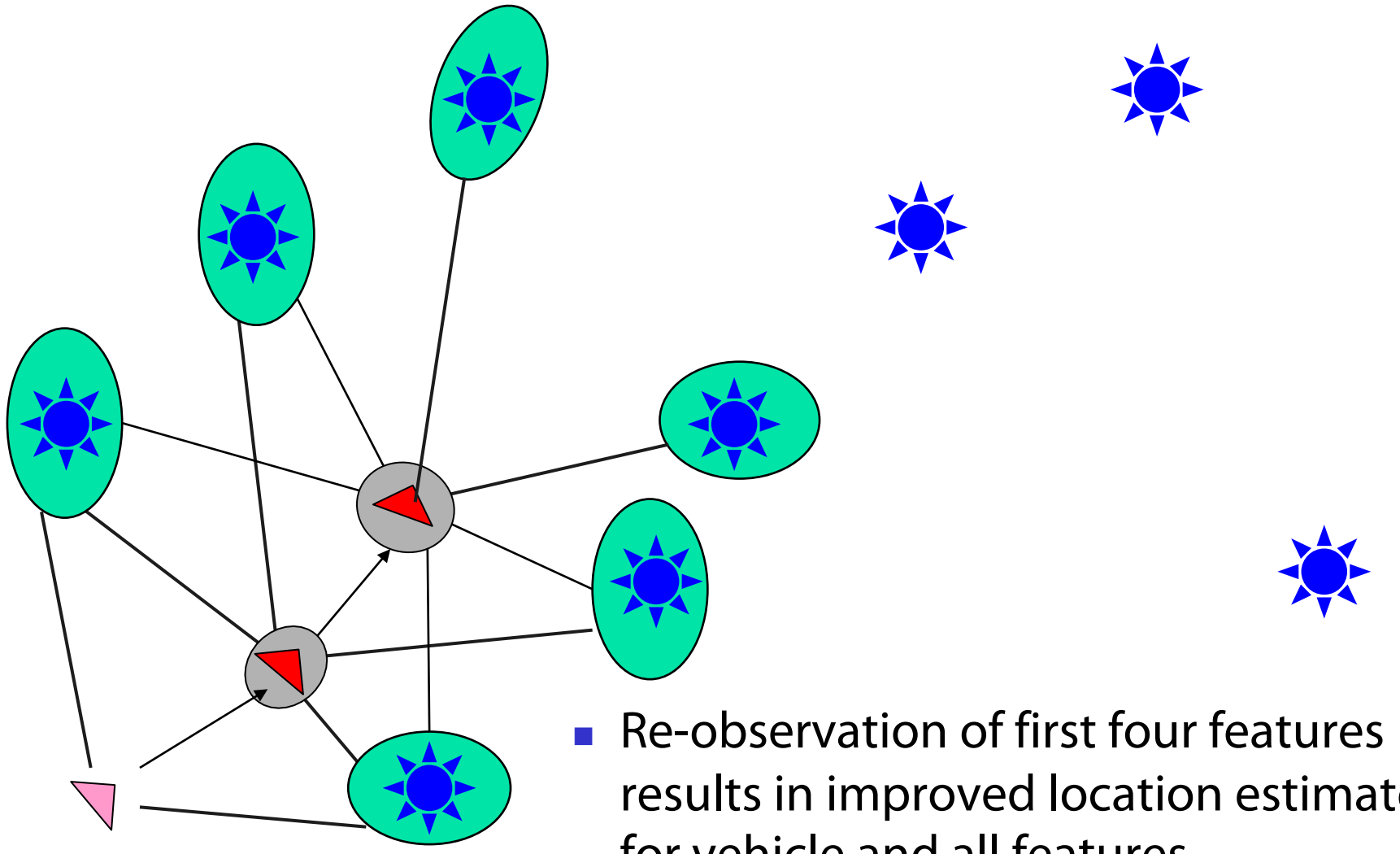
Illustration of SLAM with Landmarks



- Third position: two additional features added to map

Courtesy J. Leonard

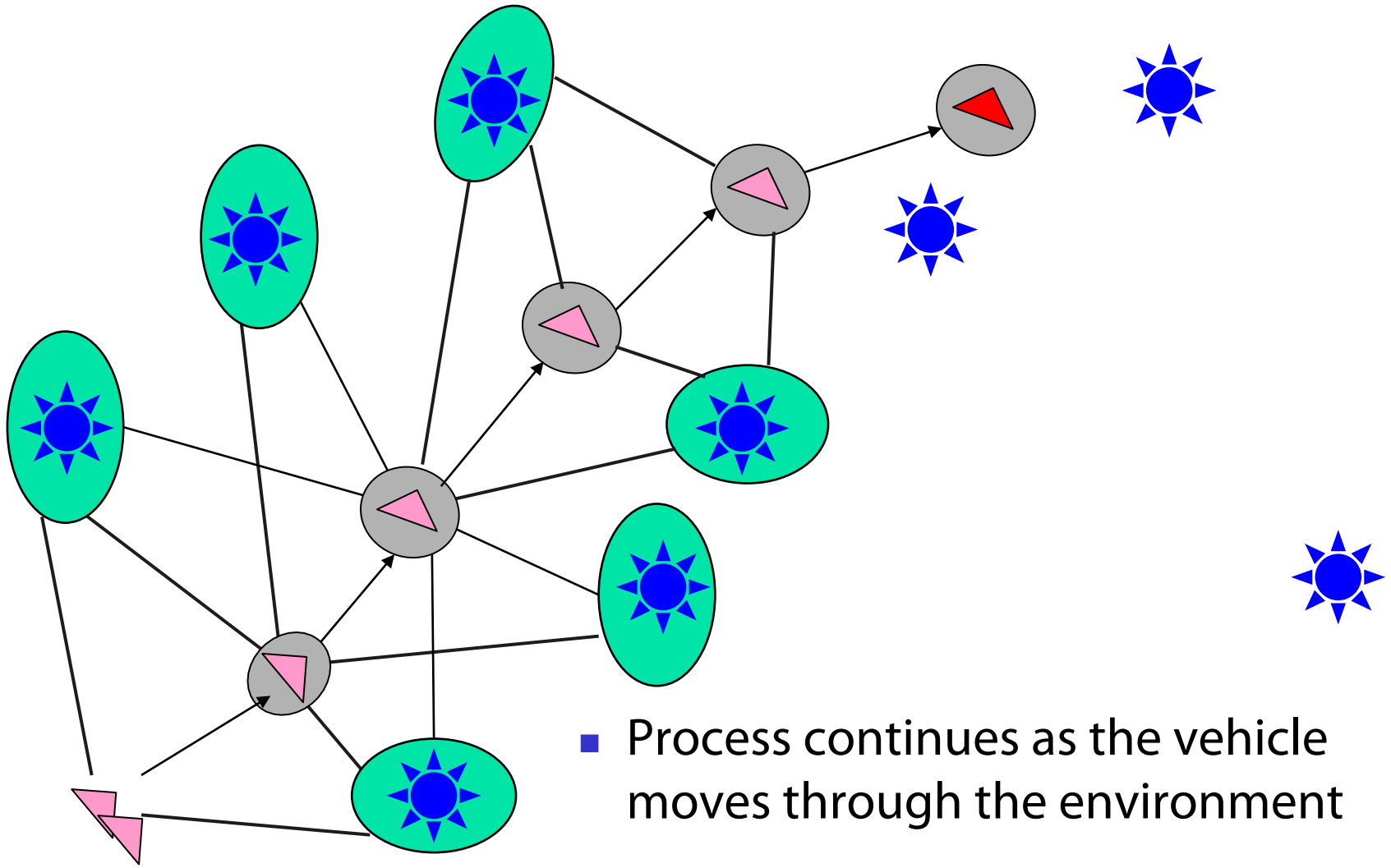
Illustration of SLAM with Landmarks



- Re-observation of first four features results in improved location estimates for vehicle and all features

Courtesy J. Leonard

Illustration of SLAM with Landmarks

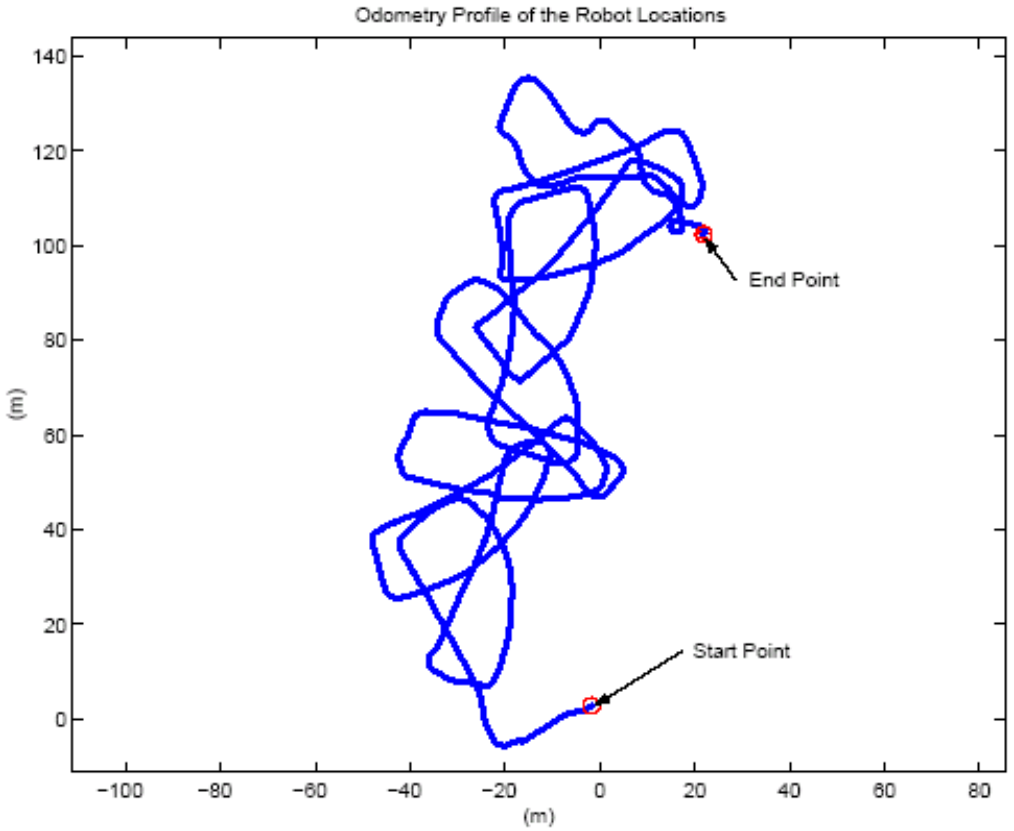


- Process continues as the vehicle moves through the environment

Courtesy J. Leonard

SLAM Using Landmarks

MIT Indoor Track



Courtesy J. Leonard

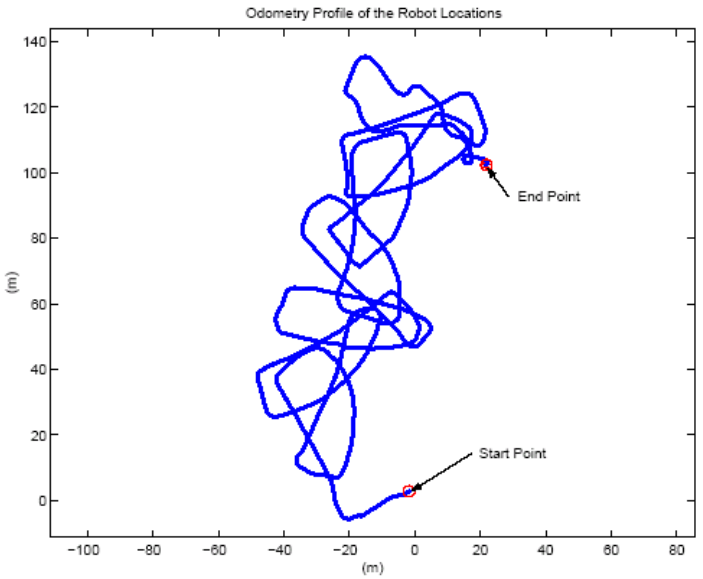
Test Environment (Point Landmarks)

Courtesy J. Leonard

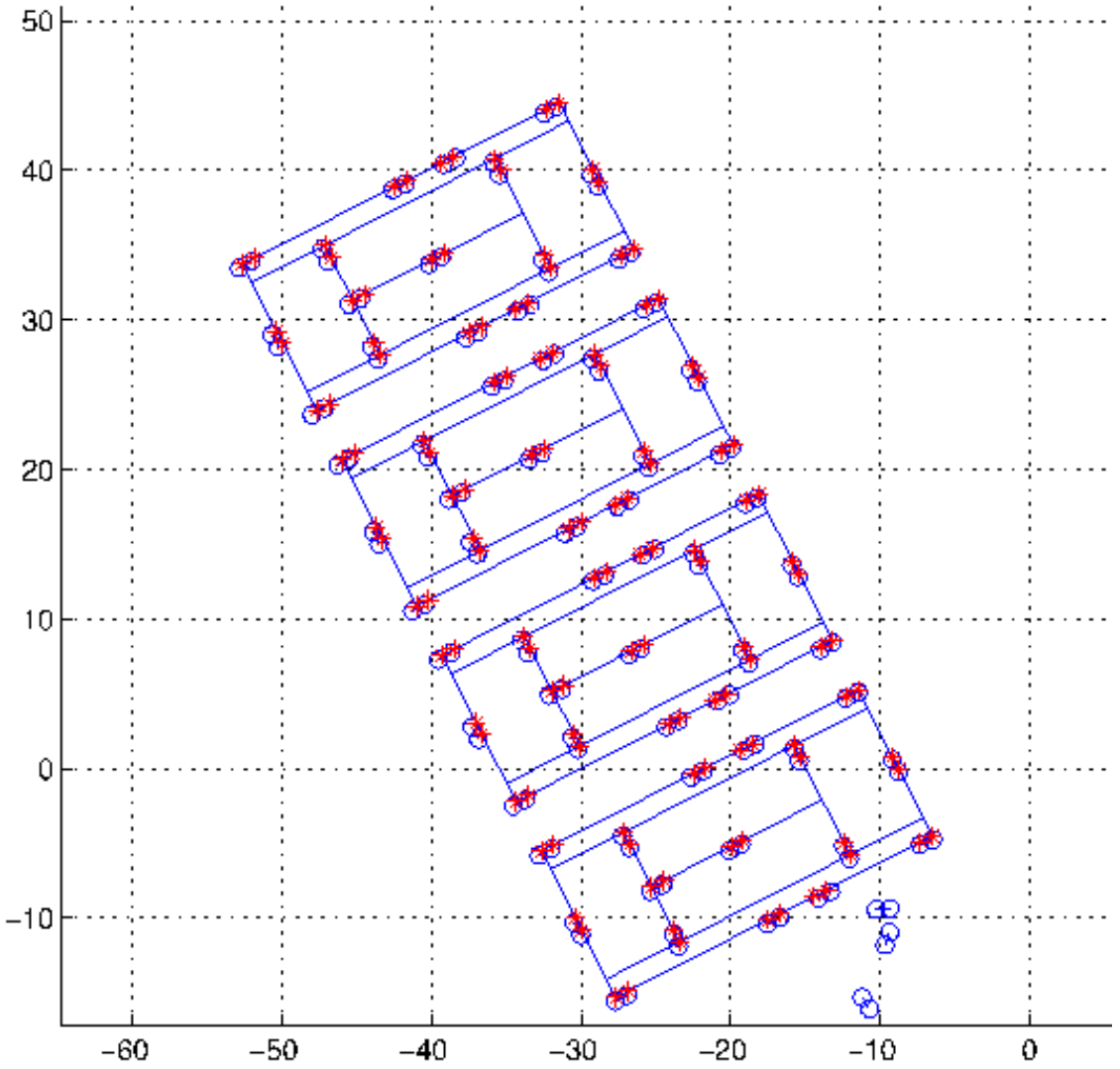
View from Vehicle

Courtesy J. Leonard

Comparison with Ground Truth



Odometry

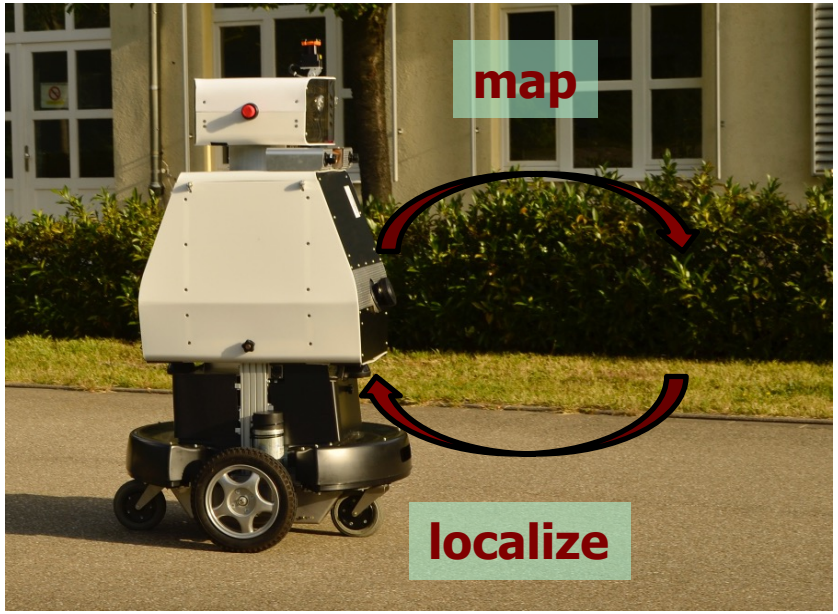


SLAM result

Courtesy J. Leonard

Simultaneous Localization and Mapping (SLAM)

- Building a map and locating the robot in the map at the same time
- Chicken-and-egg problem



Courtesy: Cyrill Stachniss

Definition of the SLAM Problem

Given

- The robot's controls

$$u_{1:T} = \{u_1, u_2, u_3, \ldots, u_T\}$$

- Observations

$$z_{1:T} = \{z_1, z_2, z_3, \ldots, z_T\}$$

Wanted

- Map of the environment m

- Path of the robot $x_{0:T} = \{x_0, x_1, x_2, \ldots, x_T\}$

Courtesy: Cyrill Stachniss

Bayes Filter

- Recursive filter with prediction and correction step

- Prediction

$$\overline{Bel}(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

- Correction

$$Bel(x_t) = \eta p(z_t|x_t) \overline{Bel}(x_t)$$

- EKF Slam sets x to be (position of robot, position of landmarks)

EKF SLAM

- Application of the EKF to SLAM
- Estimate robot's pose and locations of landmarks in the environment
- Assumption: known correspondences
- State space (for the 2D plane) is

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

Courtesy: Cyrill Stachniss

EKF SLAM: State Representation

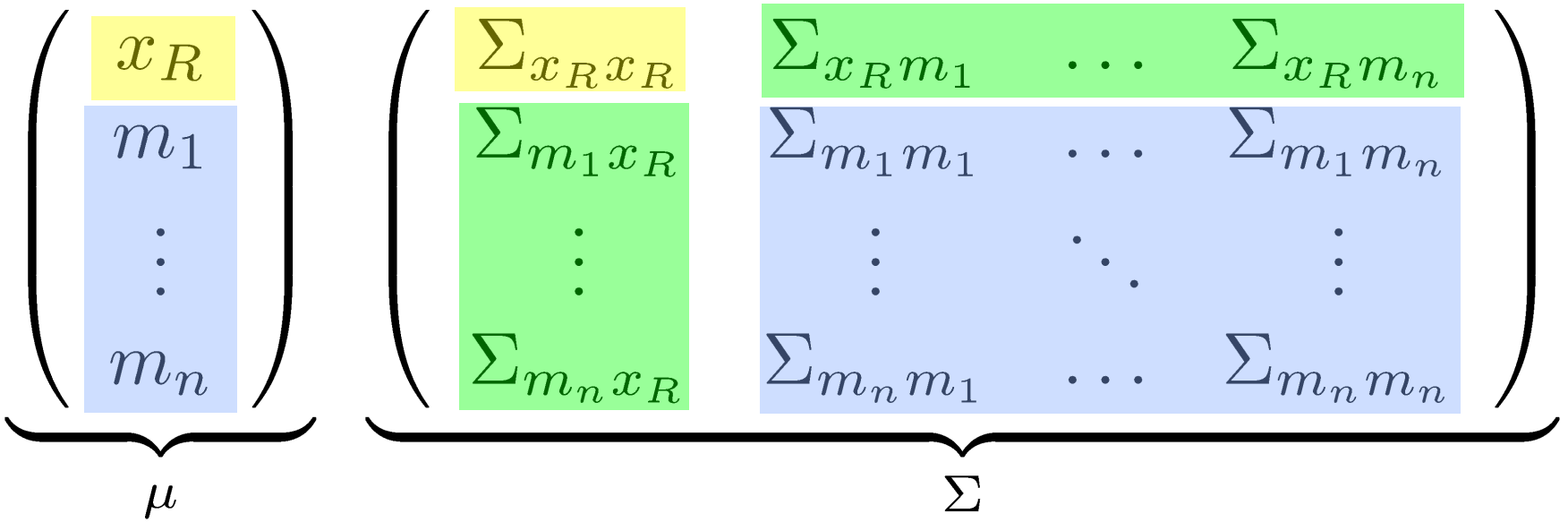
- Map with n landmarks: (3+2n)-dimensional Gaussian
- Belief is represented by

$$\underbrace{\begin{pmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{pmatrix}}_{\Sigma}$$

Courtesy: Cyrill Stachniss

EKF SLAM: State Representation

- More compactly



Courtesy: Cyrill Stachniss

EKF SLAM: State Representation

- Even more compactly (note: $x_R \rightarrow x$)

$$\underbrace{\begin{pmatrix} x \\ m \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix}}_{\Sigma}$$

Courtesy: Cyrill Stachniss

EKF SLAM: Filter Cycle

1. State prediction
2. Measurement prediction
3. Measurement + Data Association
4. Update

Courtesy: Cyrill Stachniss

Why is this useful - SLAM

Class Outline