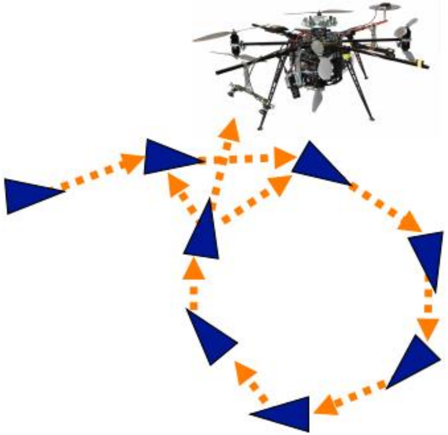
COMS W4733: Computational Aspects of Robotics

Lecture 22: Graph SLAM



Instructor: Tony Dear

Slide materials from P. Allen, M. Bennewitz, W. Burgard, D. Fox, M. Kaess, L. Spinello, C. Stachniss, D. Tipaldi, et al.

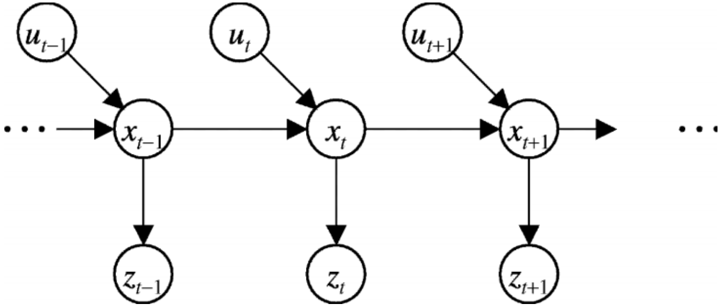
State Estimation

- Belief distribution $B(\boldsymbol{x}_t) = p(\boldsymbol{x}_t \mid \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t})$

Robot state at time t Actions from 1 to t Observations from 1 to t

- Transition model $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$

- Observation model $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t)$



Bayes Filter Algorithm

Algorithm **Bayes_filter**($B(\boldsymbol{x}), \boldsymbol{d}$):

1. $\eta = 0$
2. **if** $\boldsymbol{d}$ is an *action* data item $\boldsymbol{u}$ **then**
3. **for all** $\boldsymbol{x}$ **do**
4. $\overline{B}(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{x}', \boldsymbol{u}) B(\boldsymbol{x}') d\boldsymbol{x}'$
5. **if** $\boldsymbol{d}$ is a *perceptual* data item $\boldsymbol{z}$ **then**
6. **for all** $\boldsymbol{x}$ **do**
7. $\overline{B}(\boldsymbol{x}) = p(\boldsymbol{z}|\boldsymbol{x}) B(\boldsymbol{x})$
8. $\eta = \eta + \overline{B}(\boldsymbol{x})$
9. **for all** $\boldsymbol{x}$ **do**
10. $\overline{B}(\boldsymbol{x}) = \eta^{-1} \overline{B}(\boldsymbol{x})$
11. **return** $\overline{B}(\boldsymbol{x})$

Prediction:

$$B'(\boldsymbol{x}_t) = \int \underbrace{p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}_{\text{Transition model}} B(\boldsymbol{x}_{t-1}) \, d\boldsymbol{x}_{t-1}$$

Observation:

$$B(\boldsymbol{x}_t) = \eta^{-1} \underbrace{p(\boldsymbol{z}_t|\boldsymbol{x}_t)}_{\text{Observation model}} B'(\boldsymbol{x}_t)$$

Kalman Filter

- Start with current belief distribution:

$$Bel(\boldsymbol{x}_{k-1}) \sim N(\widehat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1})$$

- Predict according to transition model:

$$\widehat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \widehat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k \boldsymbol{u}_k$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k$$

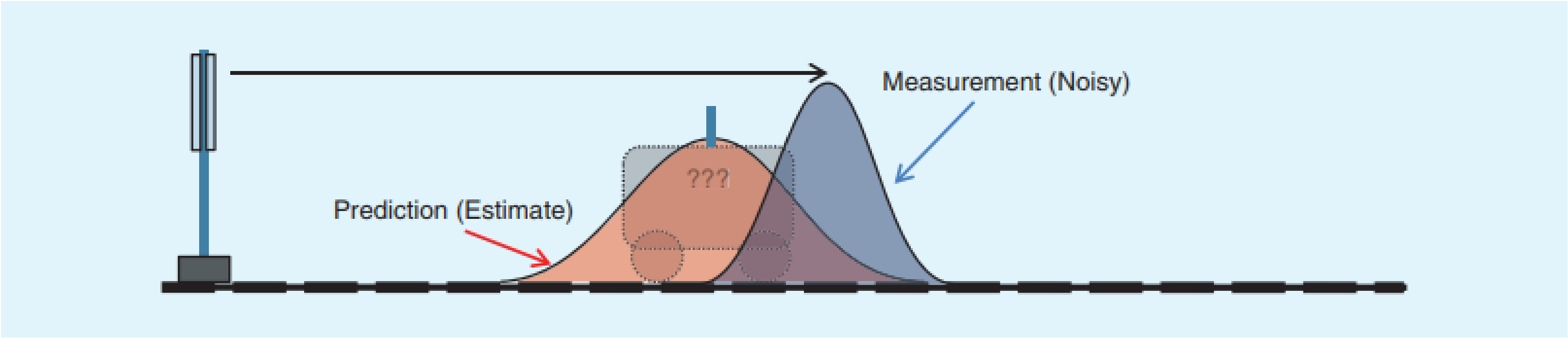
- Update according to observation model and measurement:

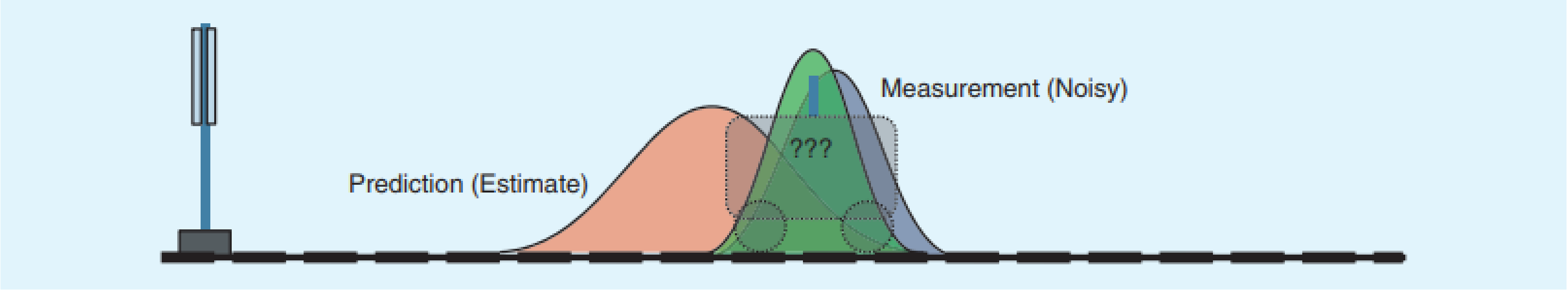
$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \left(\boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k \right)^{-1}$$

$$\widehat{\boldsymbol{x}}_{k|k} = \widehat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k \widehat{\boldsymbol{x}}_{k|k-1})$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{P}_{k|k-1}$$

Example: Kalman Filter





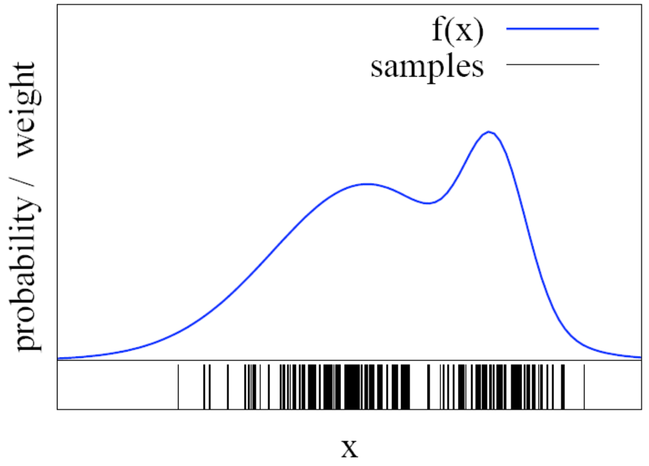
Particle Filter

Algorithm **Particle_filter**($X_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$):

1. $\overline{X}_t = \emptyset, X_t = \emptyset$
2. **for** each particle x_{t-1}^j in X_{t-1} **do**
3. sample x_t^j from $p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^j, \boldsymbol{u}_t)$
4. compute weight $w_t^j = p(\boldsymbol{z}_t | \boldsymbol{x}_t^j)$
5. insert (x_t^j, w_t^j) into $\overline{X}_t$
6. **for** all j **do**
7. sample $i \in \{1,2,\ldots,J\}$ with prob $\frac{w_t^j}{\sum w_t^j}$
8. insert x_t^i from $\overline{X}_t$ into X_t
9. **return** X_t

$$B'(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) B(\boldsymbol{x}_{t-1})\, d\boldsymbol{x}_{t-1}$$

$$B(\boldsymbol{x}_t) = \eta^{-1} p(\boldsymbol{z}_t | \boldsymbol{x}_t) B'(\boldsymbol{x}_t)$$

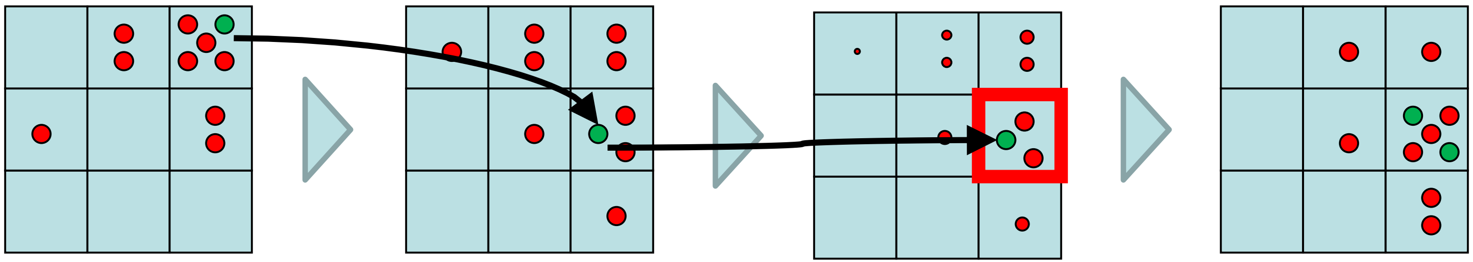


Example: Particle Filter

sample $p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^j, \boldsymbol{u}_t)$

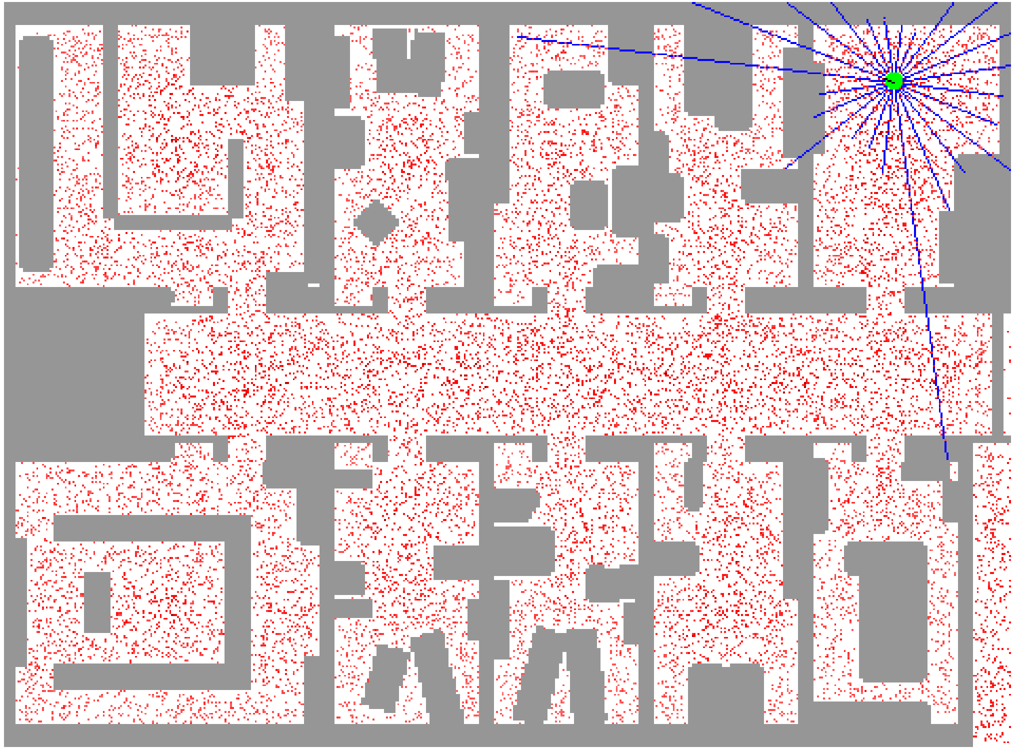
weight $w_t^j = p(\boldsymbol{z}_t | \boldsymbol{x}_t^j)$

Resample (renormalize):



Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,2)	(2,2) w=.4	(3,2)

Example: Particle Filter

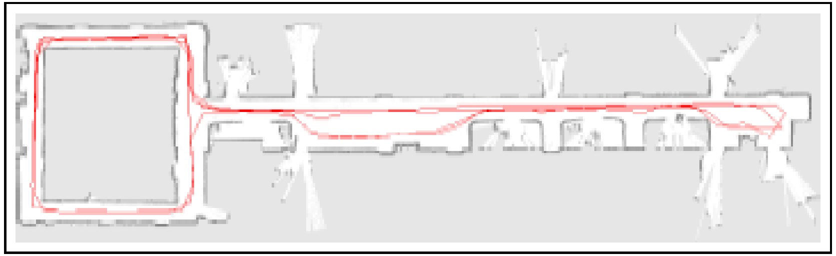


Example: Particle Filter

Navigation Tasks

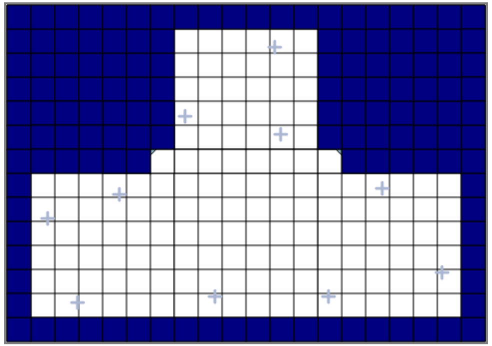
- **Localization**: Given a map, infer the robot's location
- Variations: Known/unknown initial location, false location (kidnapped robot)

- **Mapping**: Given location(s), infer a *map* of the environment
  - Manually driven or fully autonomous?
  - How to define exploration of the area?
  - Termination condition?

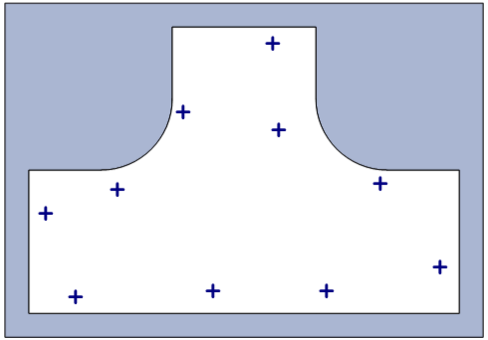


Map Representations

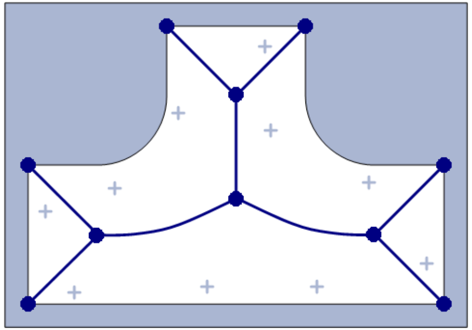
Grid-based:
Collection of discretized obstacle / free-space pixels



Feature-based:
Collection of landmark locations and correlated uncertainties



Topological:
Collection of nodes and interconnections



Map Representations

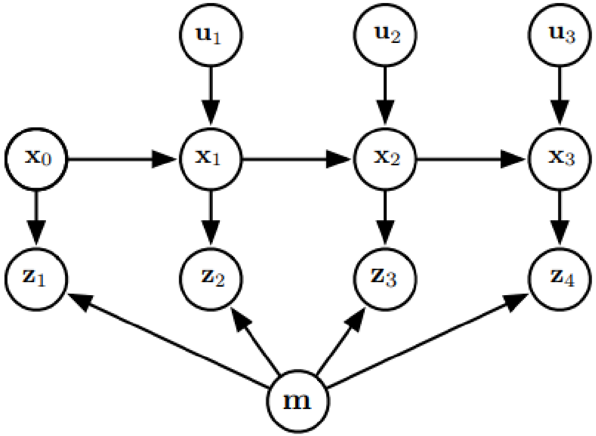
	Grid-Based	**Feature-Based**	**Topological**
Resolution vs. Scale	Discrete localization	Arbitrary localization	Localize to nodes
Computational Complexity	Grid size *and* resolution	Landmark covariance (N^2)	Minimal complexity
Obstacle Avoidance	Discretized obstacles	Only structured obstacles	GVG defines the safest path
Exploration Strategies	Frontier-based exploration	No inherent exploration	Graph exploration

SLAM

- **Simultaneous localization and mapping (SLAM)** entails learning a map and locating the robot at the same time!
- Clearly harder than either problem separately

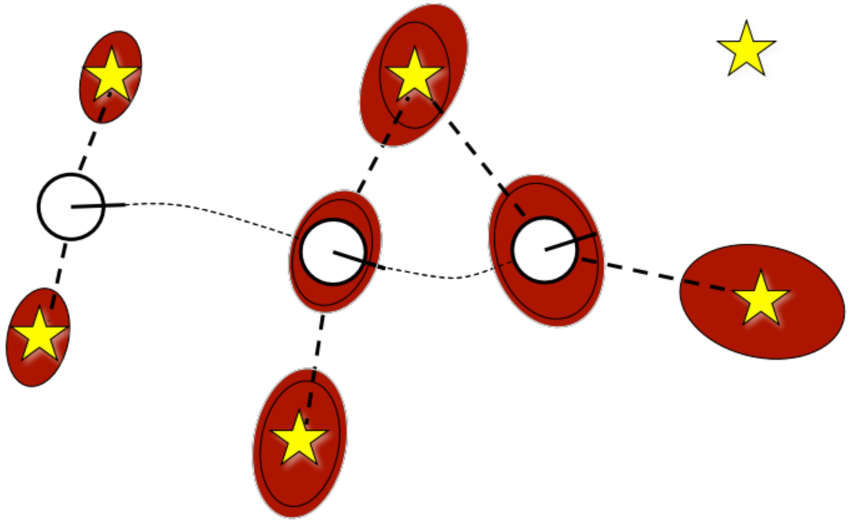
- Main approach: Add the map (e.g., in the form of feature locations) to the state and update using Bayesian filtering

- Map is another hidden state!



Feature-Based SLAM

- At any given time, robot may be able to observe some number of landmarks
- These measurements are always *relative*, not absolute
- Errors in robot pose estimate and map representation are correlated!



Bayes Filter with Landmarks

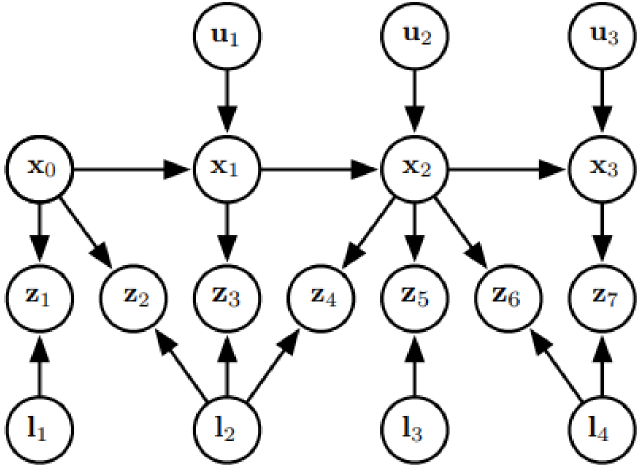
- Belief distribution: $$B(\boldsymbol{x}_t, \boldsymbol{m}) = p(\boldsymbol{x}_t, \boldsymbol{m} | \, \boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t})$$

- Prediction:

$$B'(\boldsymbol{x}_t, \boldsymbol{m}) = \int \underbrace{p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}_{\text{Transition model}} B(\boldsymbol{x}_{t-1}, \boldsymbol{m}) \, d\boldsymbol{x}_{t-1}$$

- Observation:

$$B(\boldsymbol{x}_t, \boldsymbol{m}) = \eta^{-1} \underbrace{p(\boldsymbol{z}_t | \boldsymbol{x}_t, \boldsymbol{m})}_{\text{Observation model}} B'(\boldsymbol{x}_t, \boldsymbol{m})$$



Graph SLAM

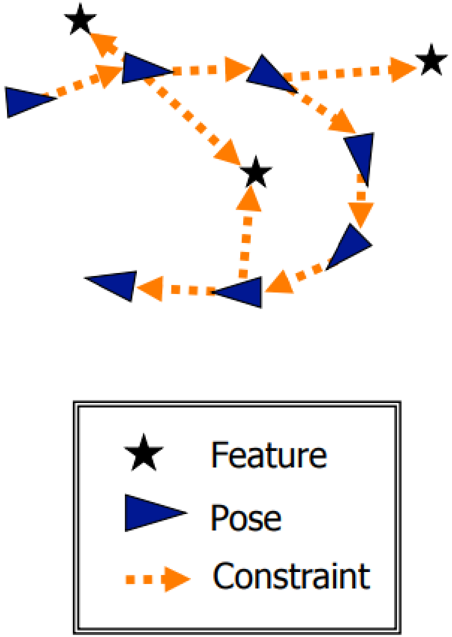
- We need to be able to correlate different observations of the same landmarks
- Idea: Use a graph to capture structure and *constraints* between observations

- Nodes: Poses of the robot and landmark locations
- Edges: Spatial constraints between poses due to successive transitions or between poses and landmarks due to observations

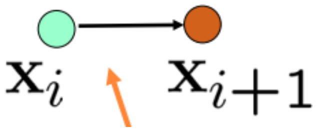
- Once graph is constructed, use optimization to minimize the total *error* as the difference between predicted and actual observations

Graph Construction

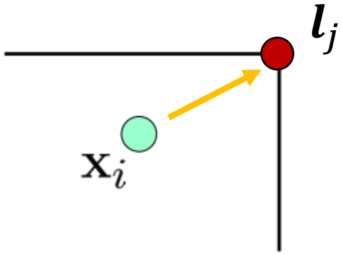
- n nodes $\boldsymbol{x}_i$, one for each robot pose at time t_i



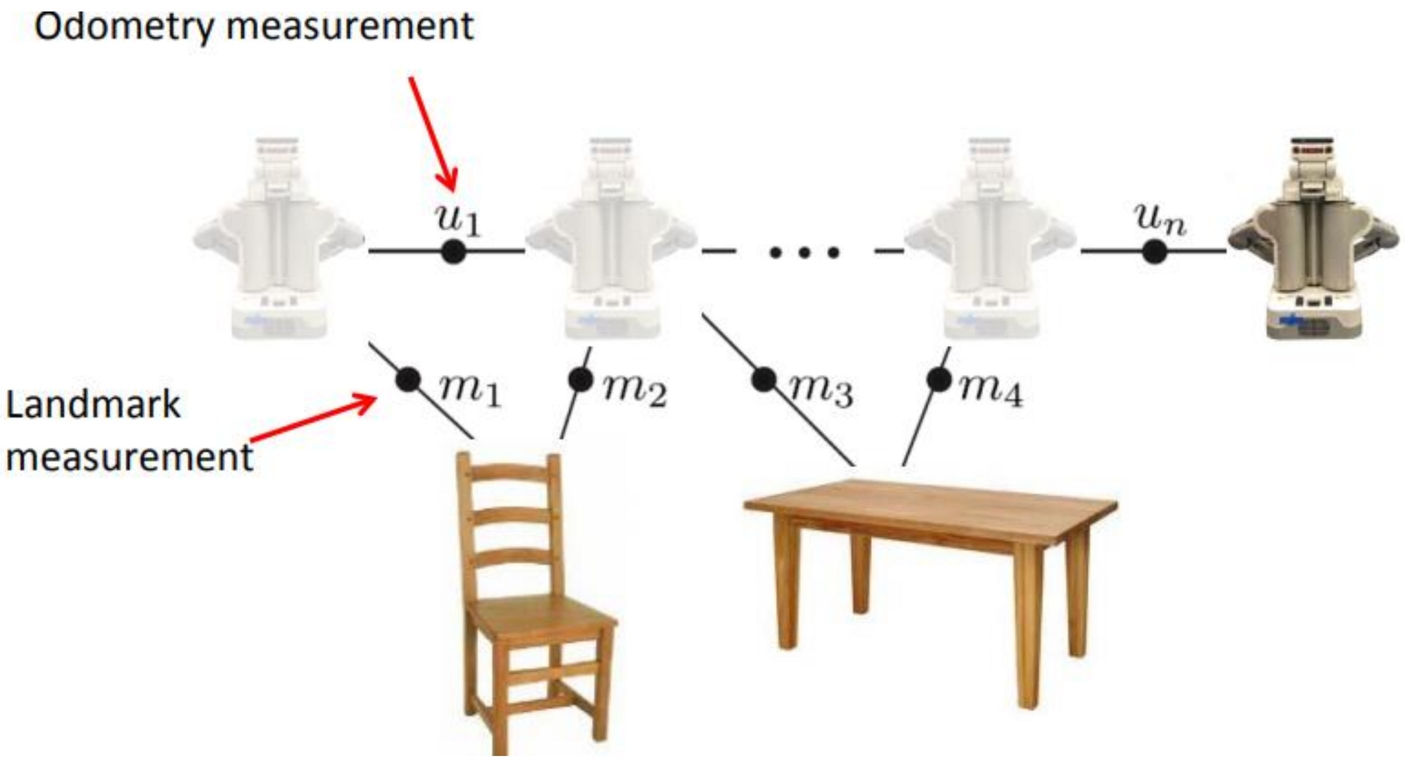
- *Odometry* edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$



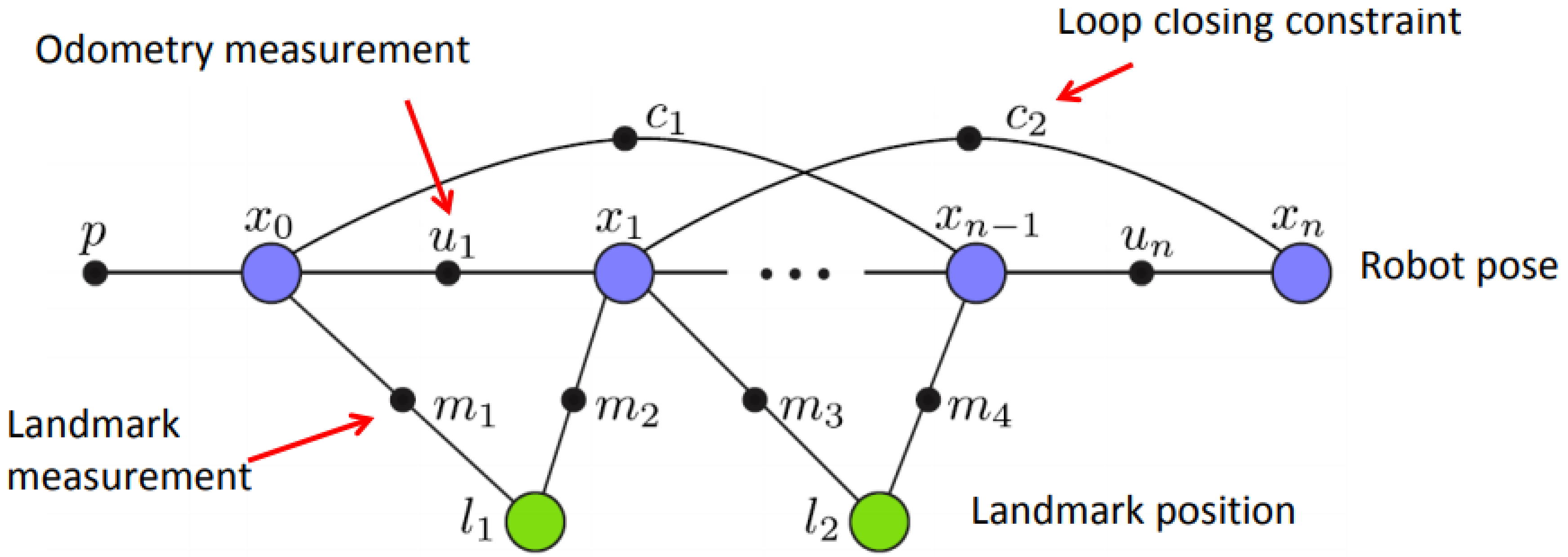
- *Observation* edge if observed $\boldsymbol{l}_j$ from $\boldsymbol{x}_i$



Graph Construction



Graph Construction



Linear Optimization

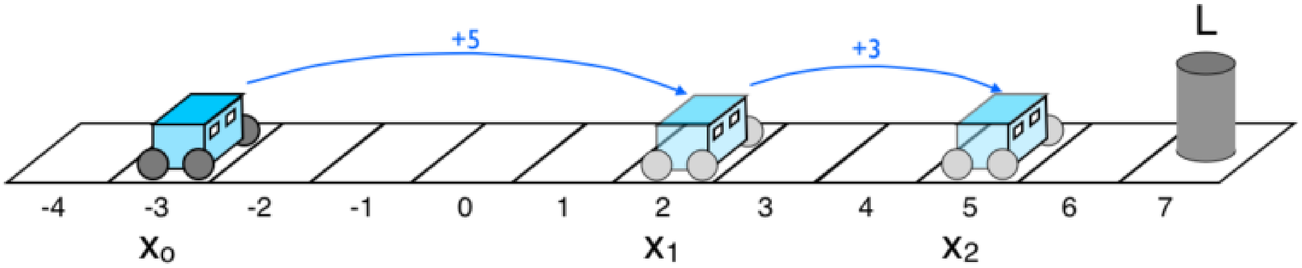
- If our models are linear (or can be linearized), then we can solve for best-fitting poses over time and landmark locations using least-squares

- Ex: 1-D state, 3 timesteps, 1 landmark

$$\boldsymbol{AX} = \boldsymbol{b}$$

Can solve for $\boldsymbol{X}$ by "pseudo-inverting" $\boldsymbol{A}$

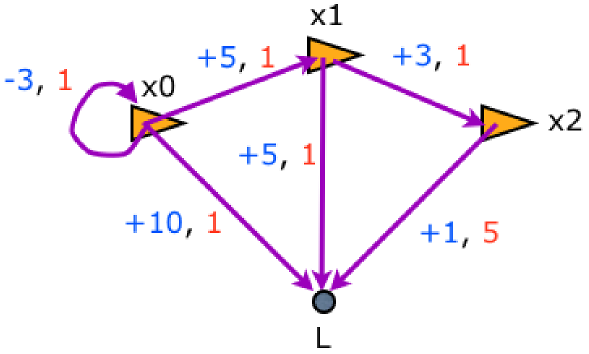
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ L \end{bmatrix} = \begin{bmatrix} -3 \\ -5 \\ -3 \\ -10 \\ -5 \\ -2 \end{bmatrix}$$



Adding Uncertainty

- Recall form of left pseudo-inverse: $\boldsymbol{X} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b}$
- How to deal with uncertainty from transitions or observations?
- They will *weight* our constraints—the greater the uncertainty (variance), the less that we should trust the particular constraint
- $\boldsymbol{W}$ is block-diagonal, containing 1/variance terms:

$$\boldsymbol{X} = \left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{b}$$

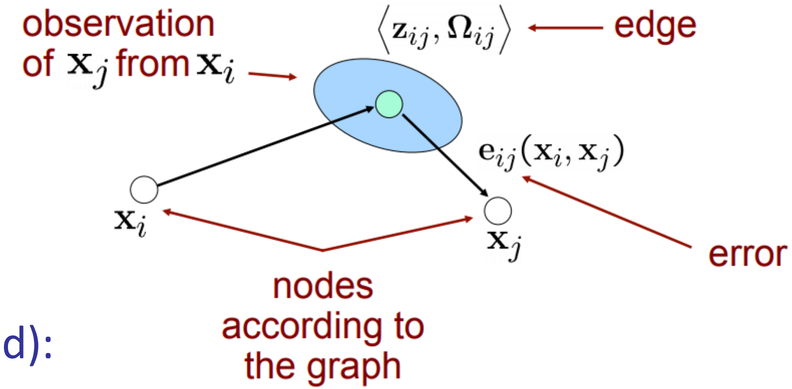


$$\boldsymbol{W} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Full Graph SLAM Problem

- In the most general case, each constraint represents an *error*
- Implicit assumption that this is zero-mean, normally distributed

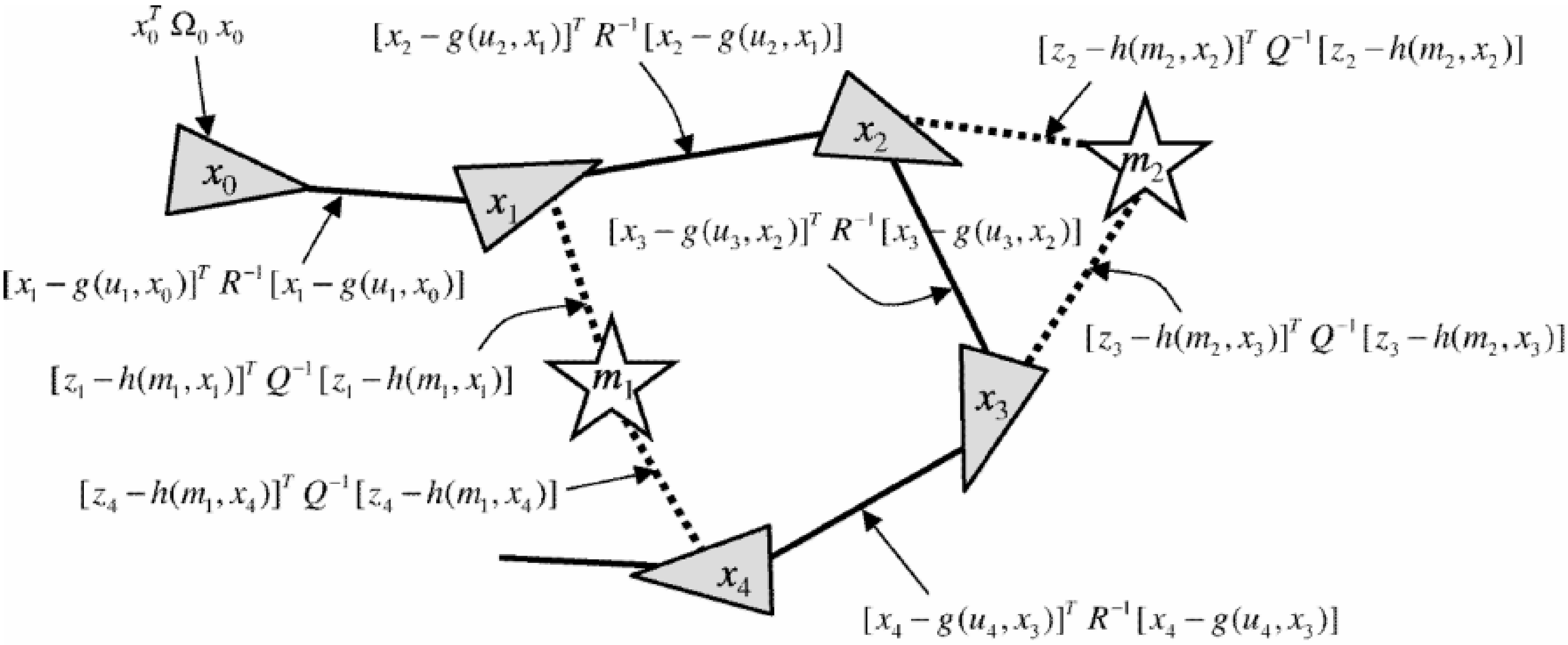
- Odometry constraints weighted by $\boldsymbol{Q}_t^{-1}$
- Observation constraints weighted by $\boldsymbol{R}_t^{-1}$
- May also have covariance for initial state $\boldsymbol{x}_0$



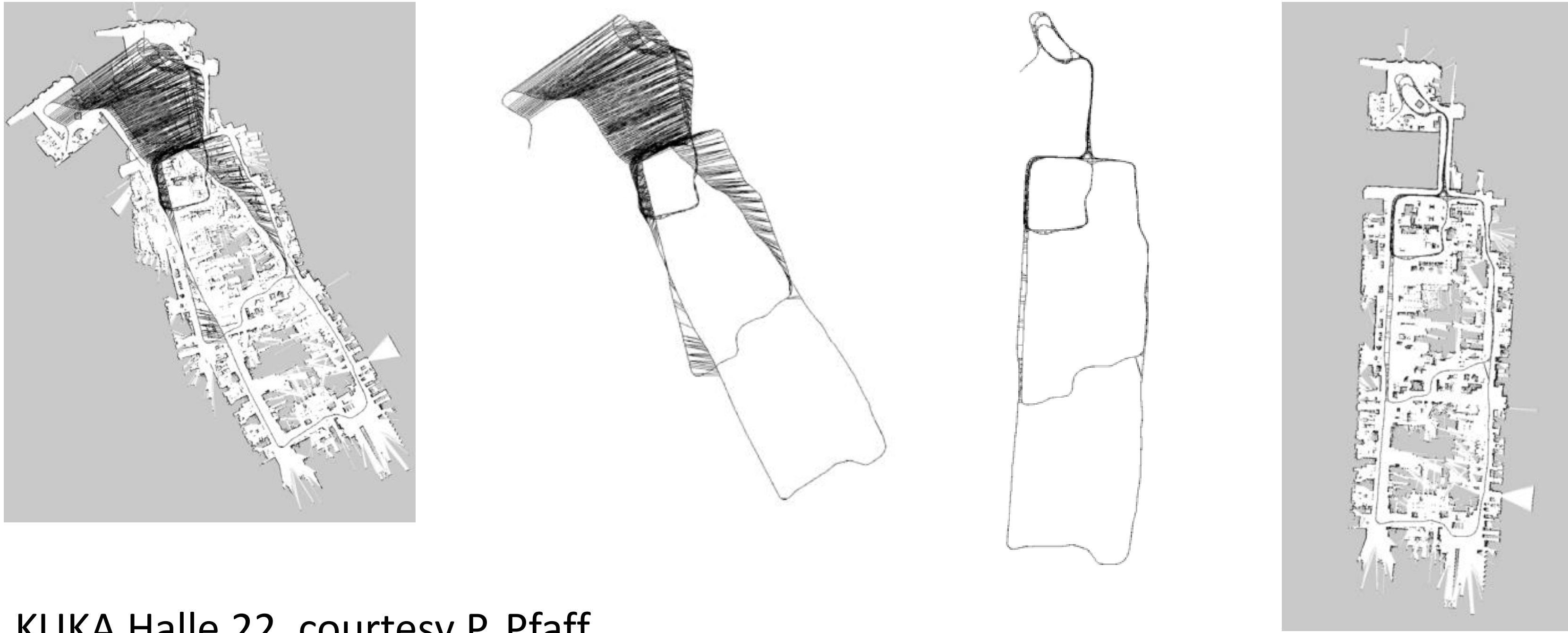
- Loss function to be minimized (may be linearized):

$$J = x_0^T \Omega_0 x_0 + \sum_t \left(x_t - f(x_{t-1}, u_t)\right)^T R_t^{-1} \left(x_t - f(x_{t-1}, u_t)\right) + \sum_{t,i} \left(z_t^i - h(x_t, m_i)\right)^T Q_t^{-1} \left(z_t^i - h(x_t, m_i)\right)$$

Full Graph SLAM Problem



Example: Graph SLAM



KUKA Halle 22, courtesy P. Pfaff

Summary

- Graph SLAM addresses full problem by optimizing (sparse) link graph between poses and landmarks
- Inference is baked into costs on edges
- Maximum likelihood estimate of both poses and map

- Still haven't really addressed the data association problem
- Long trajectories require more storage; method better suited for offline use

- Next time: Back to probabilistic methods (Kalman and particle filters!)