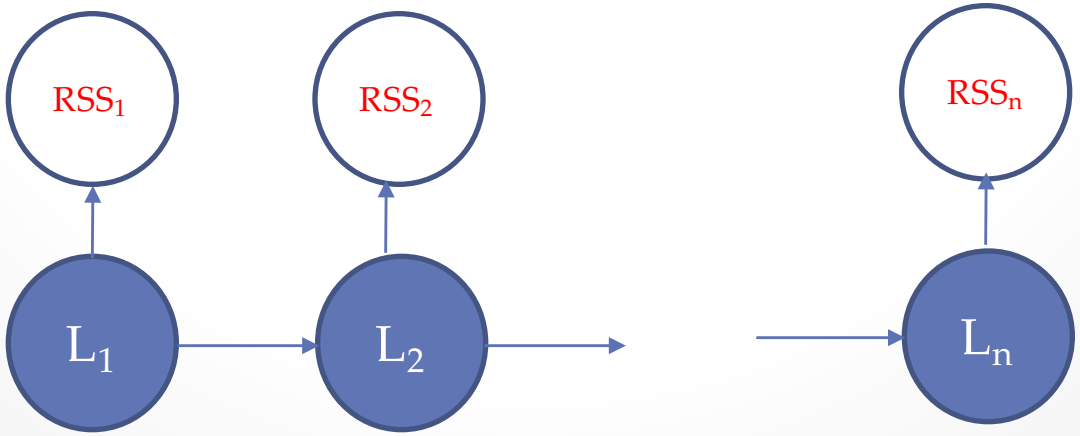
CAS 765 Fall'15
Mobile Computing and Wireless Networking

Rong Zheng

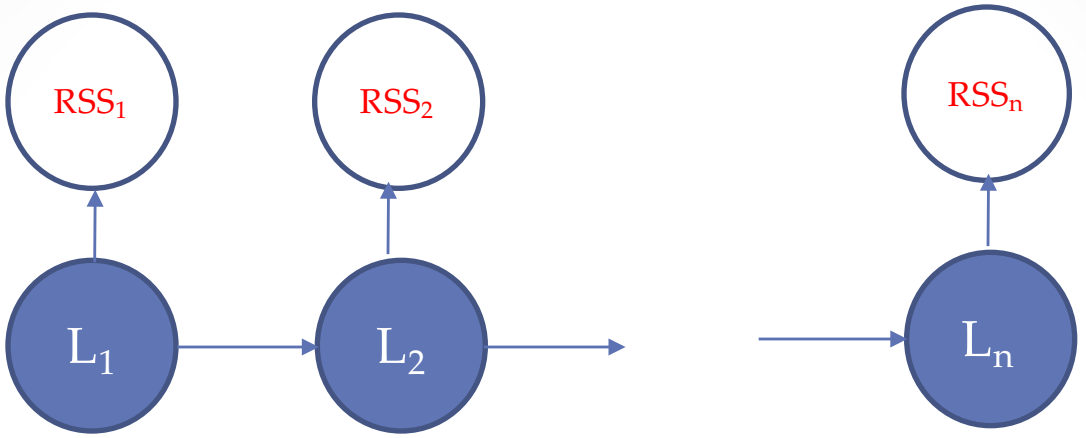
Bayesian Filtering and SLAM

Why Bayesian Filters?

- In discussion till this point, we consider a stationary system
- In practice, it is of interest to estimate the states of a time-varying system indirectly observed through noisy measurements
- Example: WiFi-based localization
  - States → Locations
  - Observations → WiFi RSSI vectors



Example Cont'd



1. We can view states in isolation and infer locations from RSS readings
2. Alternatively, we can consider the joint distribution of all possible states and observations and make inference accordingly – Computationally prohibitive as the state space grows over time!

Bayesian filters are a computational efficient way to deal with time-varying systems – incremental in nature

Review of Bayes' Theorem

- Also called Bayes' rule

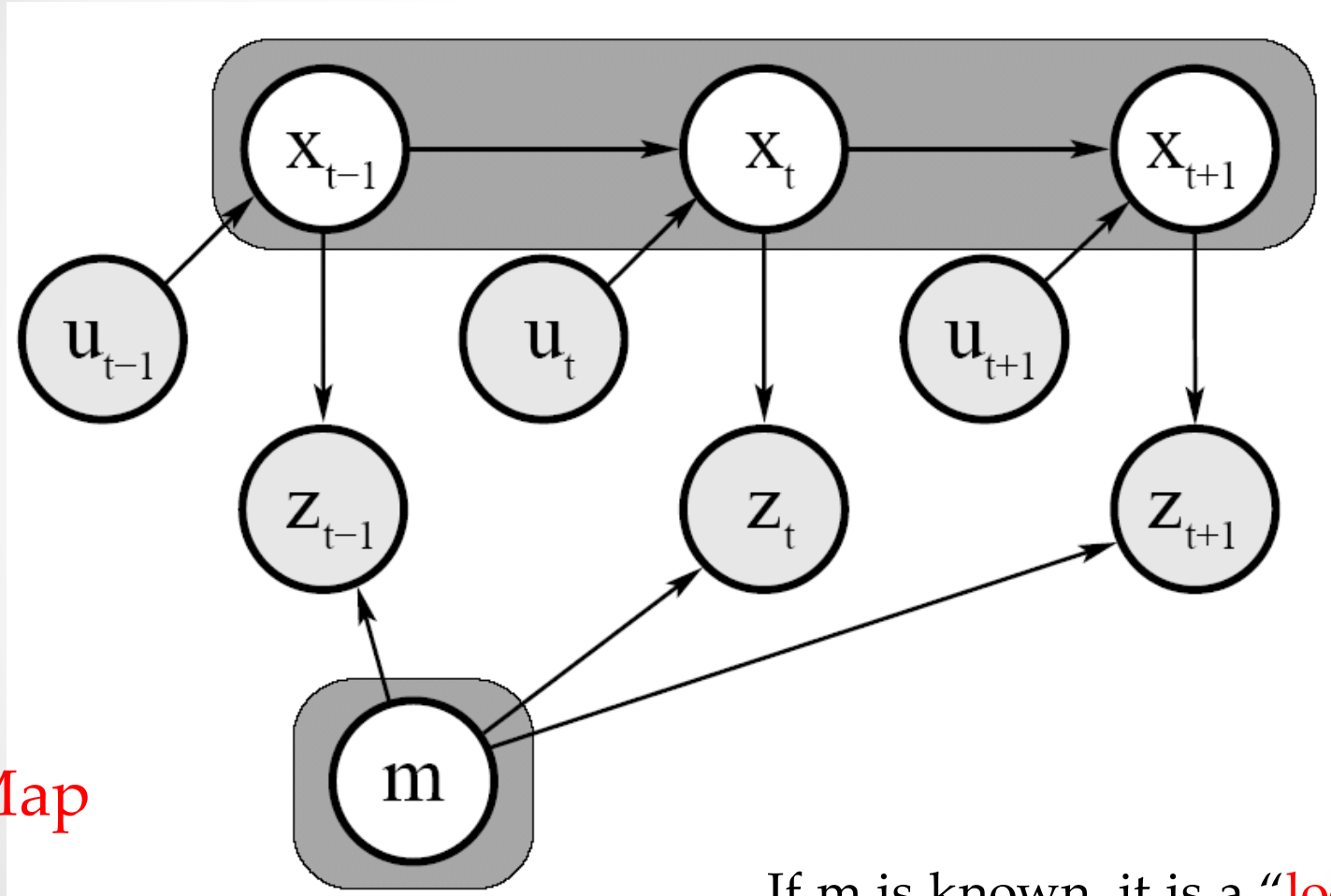
$$P(A,B) = P(A|B)P(B) \text{ or}$$

$$P(A|B) = \frac{P\,(A,B)}{P(B)}$$

- Chain rules

$$P(x_1, x_2, \dots, x_n)$$
$$= P(x_1|x_2, \dots, x_n)P(x_2|x_3, \dots, x_n) \dots P(x_{n-1}|x_n)$$

System Model



States, e.g., locations

Control, e.g., angular and translation motion

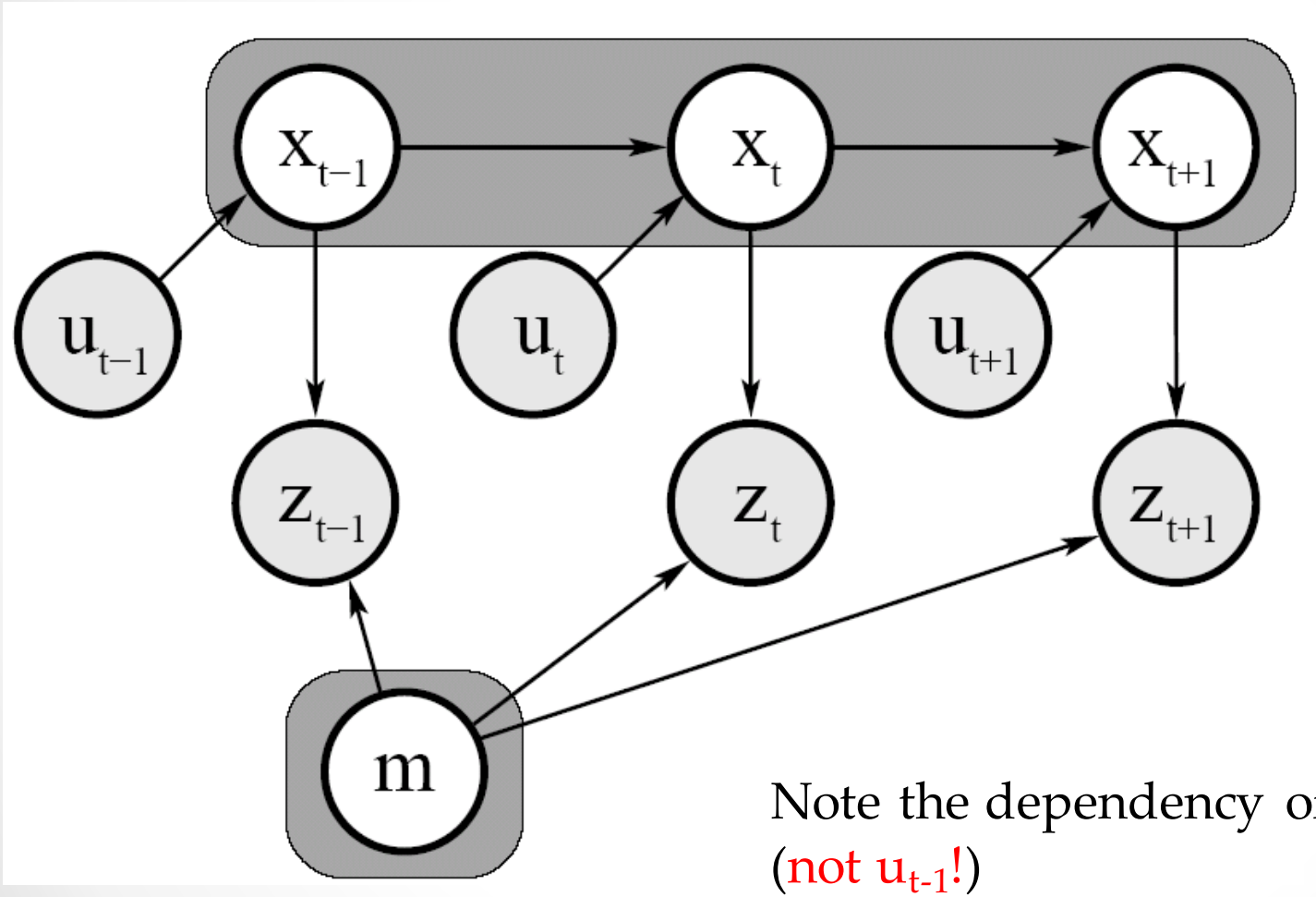
Observations, e.g., WiFi RSS, Magnetic fields

Map

If m is known, it is a "localization" problem
If m is unknown, it is a Simultaneous Localization and Mapping (SLAM) probem

Some contents are adapted from Cyrill Stachniss' slides on robotic mapping

System Model

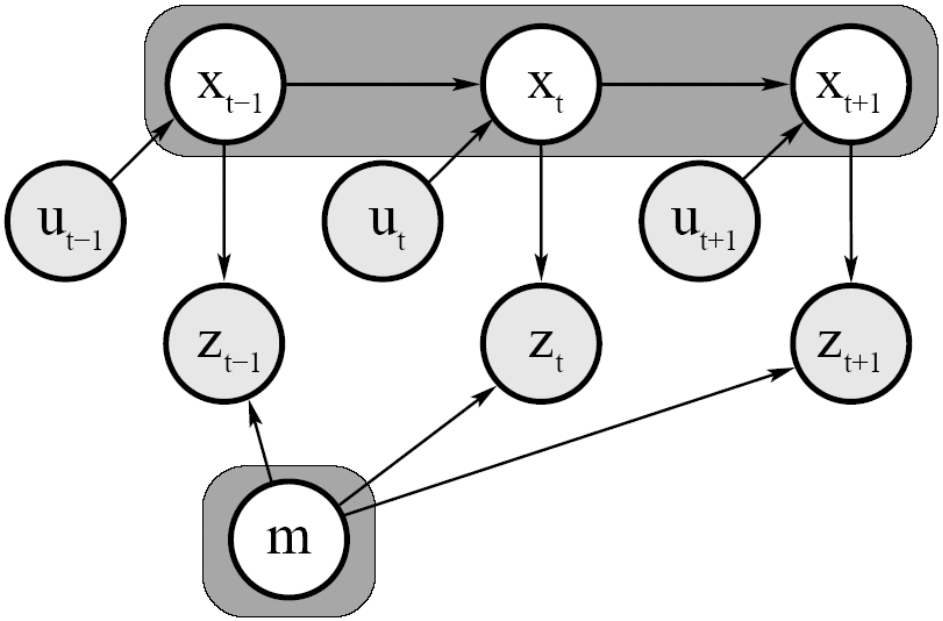


Unknown

Observed

Note the dependency of X_t on X_{t-1} and u_t (not u_{t-1}!)

System Model



- Given x_{t-1} and u_t, x_t does not depend on the past states, controls and observations (Markovian)

$$p(x_t|x_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

  - Called **motion model**

- Given m and x_t , z_t does depend on past states, observations and controls

$$p(z_t|x_{1:t}, u_{1:t}, m) = p(z_t|x_t, m)$$

  - Called **observation model**

- Observation and motion models are application-dependent

Bayes Filters

Prediction step:

$$\underbrace{p(x_t|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)} = \int p(x_t|x_{t-1}, u_t) \cdot p(x_{t-1}|z_{1:t-1}, u_{1:t}, m) dx_{t-1}$$

$$\stackrel{?}{=} \int p(x_t|x_{t-1}, u_t) \cdot \underbrace{p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)}_{bel(x_{t-1})} dx_{t-1}$$

Update step:

$$\underbrace{p(x_t|z_{1:t}, u_{1:t}, m)}_{bel(x_t)} = \eta \cdot \underbrace{p(z_t|x_t, m)}_{Observation\ model} \cdot \underbrace{p(x_t|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)}$$

An Alternative Bayes Filter

Prediction step:

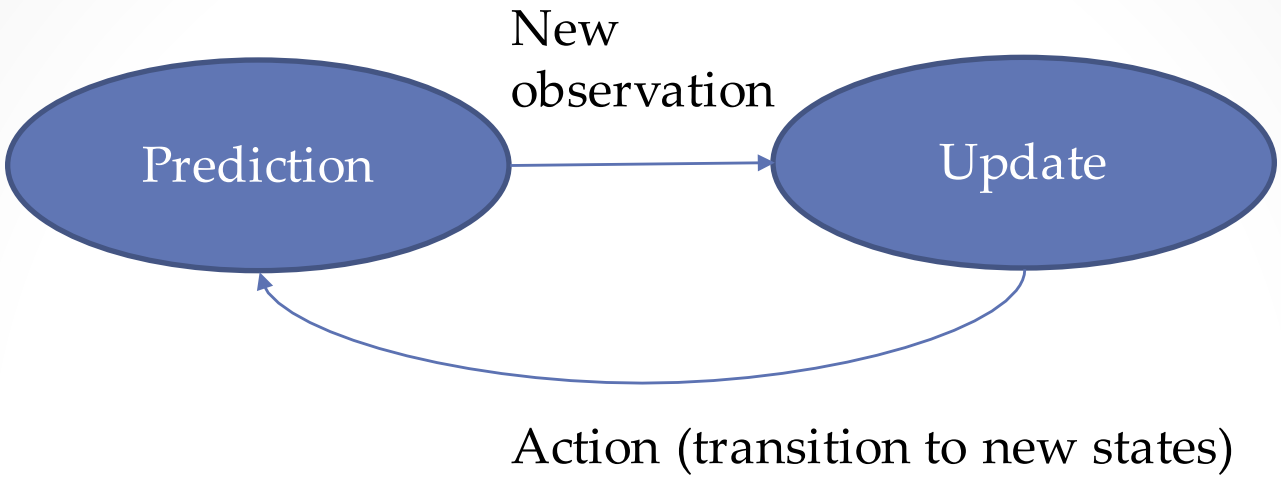
$$\underbrace{p(x_{1:t}|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)} = p(x_t|x_{t-1}, u_t) \cdot p(x_{1:t-1}|z_{1:t-1}, u_{1:t}, m)$$

$$= p(x_t|x_{t-1}, u_t) \cdot \underbrace{p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)}_{bel(x_{t-1})}$$

Update step:

$$\underbrace{p(x_{1:t}|z_{1:t}, u_{1:t}, m)}_{bel(x_t)} = \eta \cdot \underbrace{p(z_t|x_t, m)}_{Observation\ model} \cdot \underbrace{p(x_{1:t}|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)}$$

Bayes Filter Updates



- The question is how to do the updates in a computation-efficient way
- For linear models, Kalman filter is shown to optimal
- For non-linear models
  - EKF – linearize the non-linear model; approximation
  - Other KF variants (not covered)
  - Particle filters

Kalman Filter

- Motion model

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

  - A_t is a n-by-n matrix
  - Bt is a n-by-m matrix
  - ϵ_t models the randomness of state transitions ~ N(0, R_t)
- Observation model
  - Ct is a k-by-n matrix
  - δ_t models the observation noise ~ N(0, Q_t)

$$z_t = C_t x_t + \delta_t$$

- Initial state $p(x_0) \sim N(\mu_0, \Sigma_0)$

Under the Model

Prediction step:

$$\underbrace{p(x_t|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)} = \int p(x_t|x_{t-1}, u_t) \cdot p(x_{t-1}|z_{1:t-1}, u_{1:t}, m) dx_{t-1}$$

$$= \int p(x_t|x_{t-1}, u_t) \cdot \underbrace{p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)}_{bel(x_{t-1})} dx_{t-1}$$

$$\overline{bel}(x_t) \sim N\,(\underbrace{B_t u_t + A_t \mu_{t-1}}_{\overline{\mu}_t}, \underbrace{R_t + A_t \Sigma_{t-1} A_t^T}_{\overline{\Sigma}_t})$$

Under the Model

Update step:

$$\underbrace{p(x_t|z_{1:t}, u_{1:t}, m)}_{bel(x_t)} = \eta \cdot \underbrace{p(z_t|x_t, m)}_{Observation\ model} \cdot \underbrace{p(x_t|z_{1:t-1}, u_{1:t}, m)}_{\overline{bel}(x_t)}$$

$$bel(x_t) \sim N(\underbrace{\overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)}_{\mu_t}, \underbrace{(I - K_tC_t)\overline{\Sigma}_t)}_{\Sigma_t}$$

Where K_t is called the Kalman gain given by,

$$K_t = \Sigma_t C_t^T Q_t^{-1}$$

KF Algorithm

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

2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$

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

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

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

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

7: return μ_t, Σ_t

Prediction

Update

Then what?

- Maximum likelihood estimator (MLE) for the states → $x_t = \mu_t$

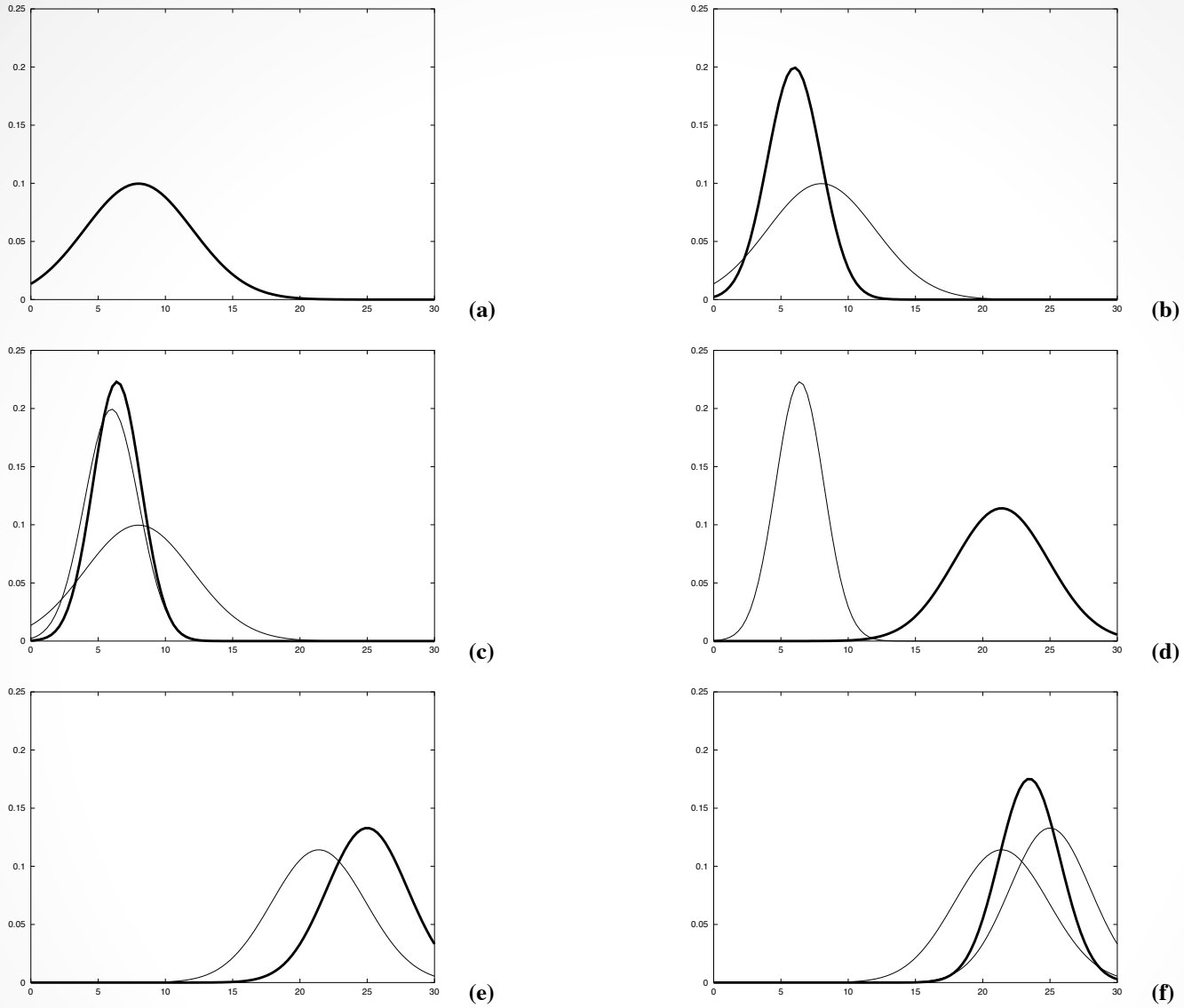


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

Extended Kalman Filter

- If the motion and observation models are non-linear,

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

Then, using Taylor expansion to linearize them at u_t, μ_{t-1}, $\bar{\mu}_t$, z_t gives

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{:=G_t}(x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \end{aligned}$$

and

$$h(x_t) \approx h(\overline{\mu}_t) + h'(\overline{\mu}_t)(x_t - \overline{\mu}_t)$$

EKF Algorithm

1: **Algorithm Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
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: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$
7: return μ_t, Σ_t

- Performance EKF depends on 1) the degree of non-linearity, and 2) the uncertainty
- Other linearization-based solutions exist: unscented KF and moment matching KF

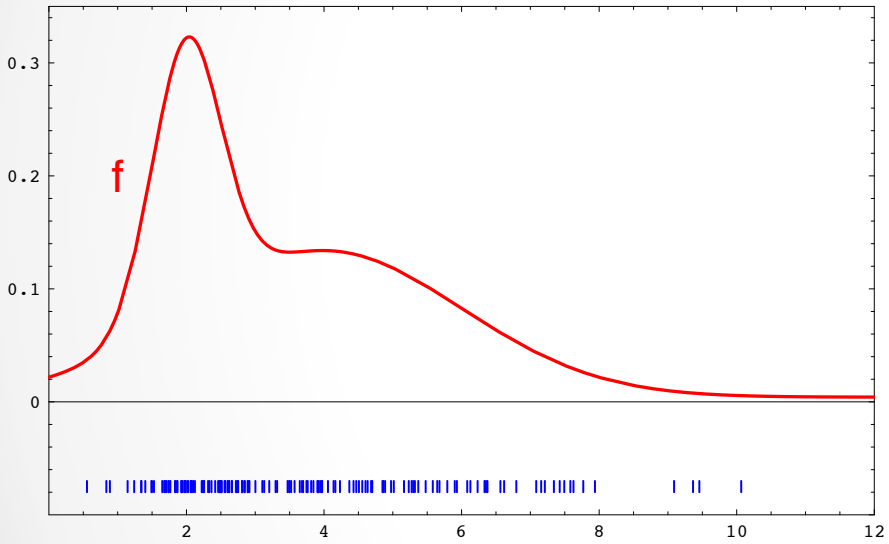
Particle Filter

- A non-parametric solution to Bayes filter – no linearity is assumed
- Key idea: use a particle mass to represent state distribution
  - Each particle corresponds to one possible state
  - The density of the particles, or the weight on the particles reflect the likelihood of the states
  - Update the particles using the motion model
  - Resample the particles from the observations

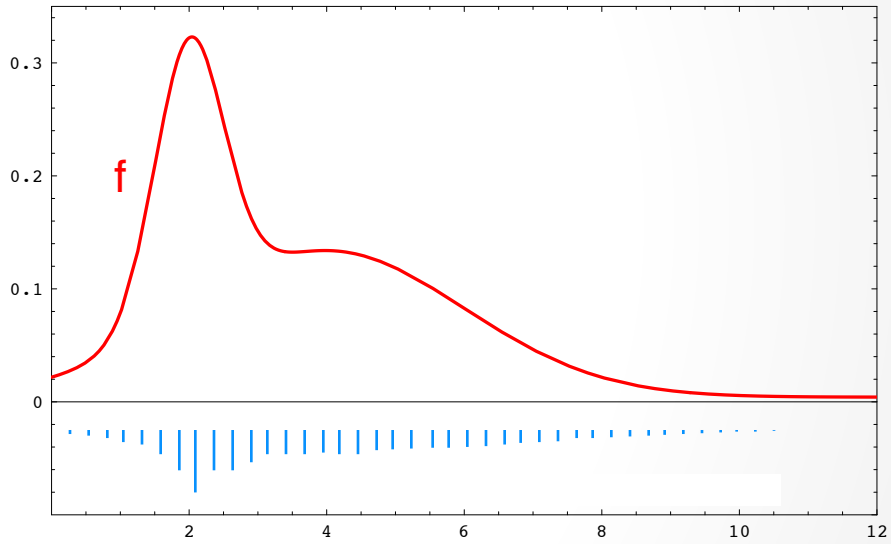
https://www.youtube.com/watch?v=aUkBa1zMKv4

A Little Digression

- How to generate samples according to a probability distribution?
  - What exactly do we mean by that?



Non-uniform sampling unity weights



Uniform sampling with weights

- For any area A,

$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} I(x^{[m]} \in A) w^{[m]} \rightarrow \int_A f(x)dx$$

More Digression

- We can also approximate the probability density with Dirac functions

$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} \delta_{x^{[m]}}(x) w^{[m]} \rightarrow f(x)$$

Therefore, for any expectation test

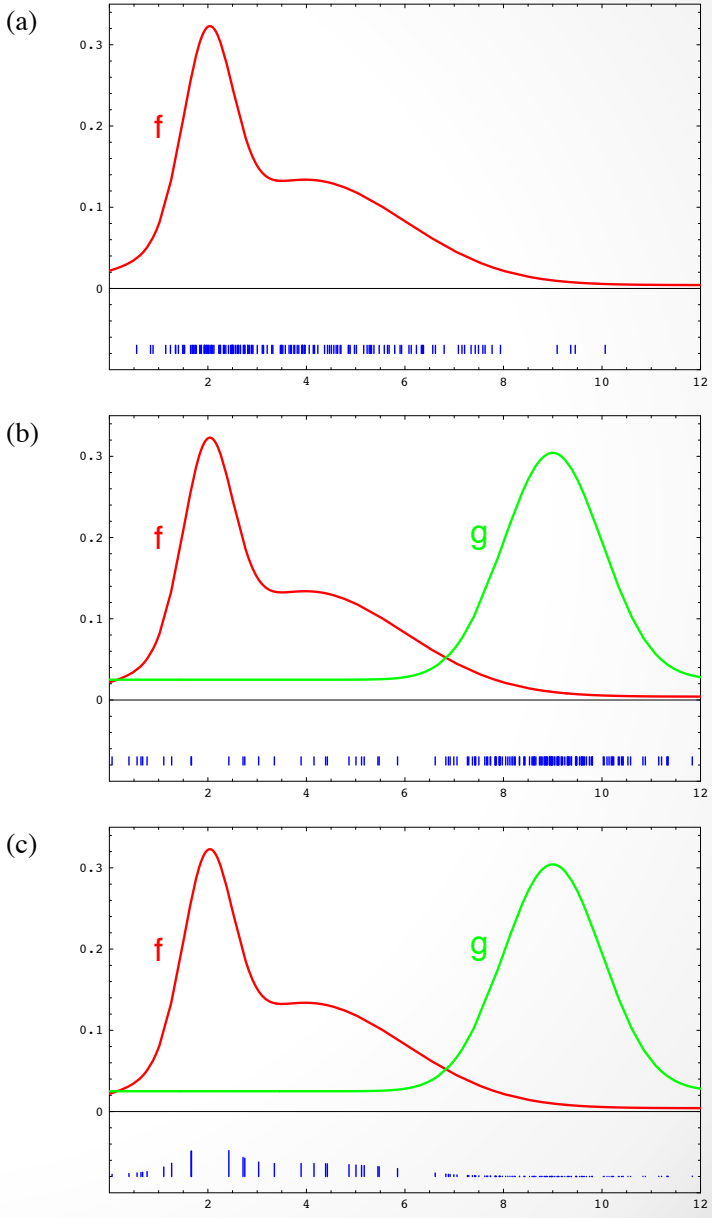
$$I(\phi) := \int \phi(x) f(x) dx$$

We have

$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} \phi(x^{[m]}) w^{[m]} \rightarrow I(\phi)$$

Sampling Methods

- Random number generator + function transformation (inverse of CDF) – good for simple distributions: Gaussian, exponential, etc. – how Matlab does it
- Importance sampling (IS): sample from a simpler distribution (called proposal distribution) and then weight the samples by the ratio between the target distribution and the proposal distribution



Sebastian Thrun, Wolfram Burgard, Dieter Fox, "Probabilistic Robotics"

More Formally…

- Importance sampling method
  - Proposal distribution g(x)
  - Target distribution f(x)
  - M samples/particles from g(x), $x^{[m]}$, m = 1, 2,…, M
  - The M particles are weighted by $f(x^{[m]})/g(x^{[m]})$
- How to choose proposal distributions?
  - Easy to sample from
  - g(x) > 0 whenever f(x) > 0
- M should be sufficiently large (typically >= 1000)

PF Algorithm

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Predication
(sample from proposal distribution)

Update
(weighted by the ratio)

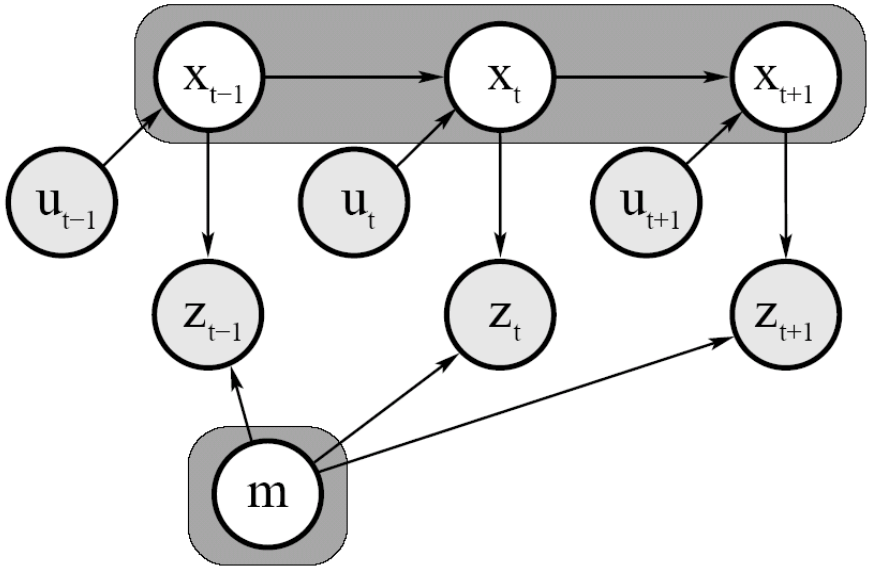
Resampling
(back to "density" based)

- Resampling needed when there are lots of samples with low weights and only a few samples at higher weights
  - Resample in each step, or
  - Resample when the disparity of weights is too large – less computation overhead

About the Exercise

- For simplicity, no observation is included in this exercise
  - Only need to consider the motion model, and
  - Wall constraints – people cannot go through the walls

SLAM



- Now, we consider the problem where maps are not known
- Bayes filter
  - Motion model

$$p(x_t|x_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

  - Observation model

$$p(z_t|x_{1:t}, u_{1:t}, m) = p(z_t|x_t, m)$$

- Goal: jointly estimate states and map
  - Online SLAM

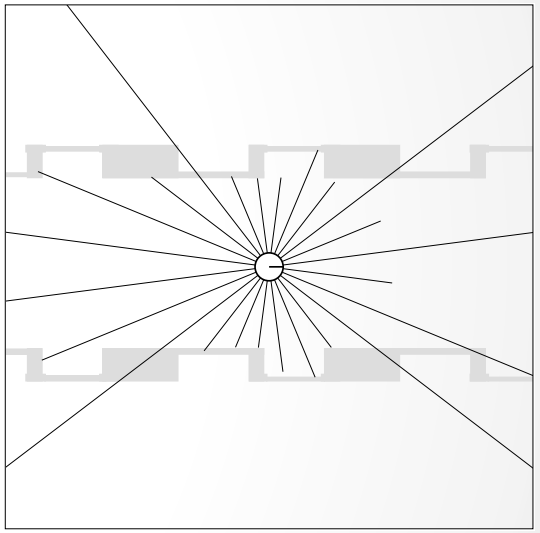
$$p(x_t, m|z_{1:t}, u_{1:t})$$

  - Full SLAM

$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

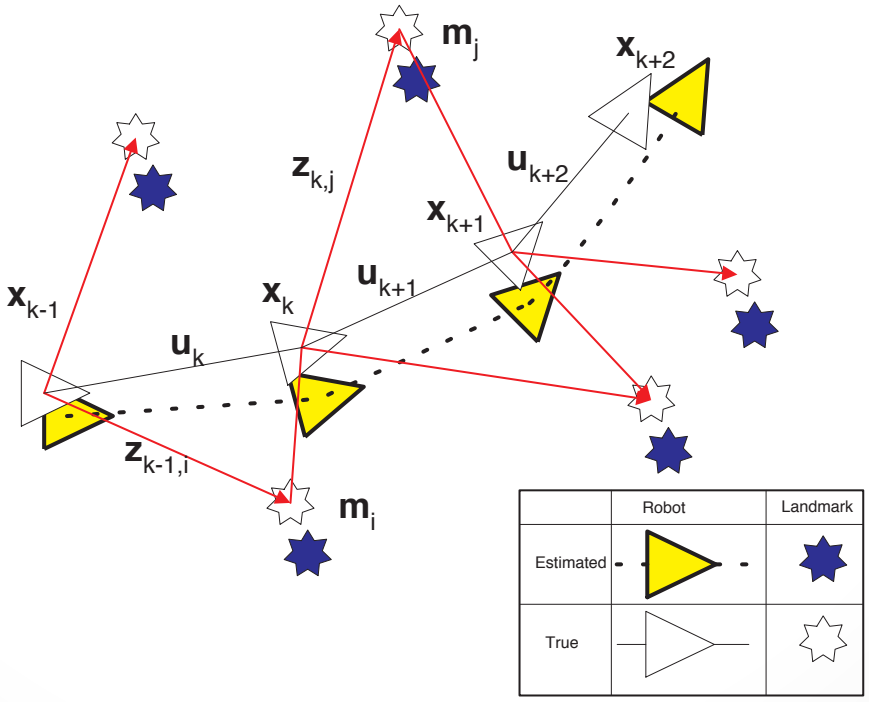
What is "Map"?

- In robotics context, map typically refers to a collection of objects
  - Obstacles, beacons …
- Two types:
  - Feature based: m_n
  - Location based: $m_{x,y}$
- Occupancy grid map is an example of location-based maps: binary variable for each grid point for presence or absence
- Landmark map is an example of feature-based maps: a collection of locations for the landmarks, e.g., WiFi APs



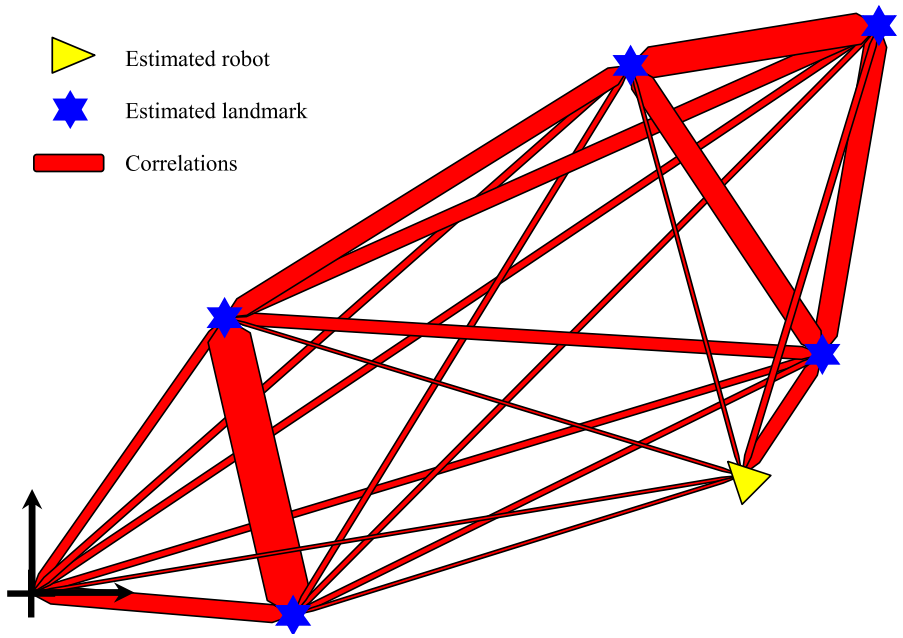
How is SLAM Even Possible?

- Suppose you are blind-folded
  - You can count your steps, turns
  - You hear sounds from unknown landmarks, which allow you to estimate their "relative" locations to you



How is SLAM Even Possible?

- Correlations among landmarks occur
  - When multiple landmarks can be observed at the same location
  - When a landmark can be observed at multiple locations



Spring network analogy. The landmarks are connected by springs describing correlations between landmarks. As the vehicle moves back and forth through the environment, spring stiffness or correlations increase (red links become thicker). As landmarks are observed and estimated locations are corrected, and these changes are propagated through the spring network. Note, the robot itself is correlated to the map.

Basic Online SLAM

- Prediction step

$$\underbrace{p(x_t, m | z_{1:t-1}, u_{1:t})}_{\overline{bel}(x_t,m)} = \int p(x_t | x_{t-1}, u_t) \cdot p(x_{t-1}, m | z_{1:t-1}, u_{1:t}) dx_{t-1}$$

$$= \int \underbrace{p(x_t | x_{t-1}, u_t)}_{Motion\ model} \cdot \underbrace{p(x_{t-1}, m | z_{1:t-1}, u_{1:t-1})}_{bel(x_{t-1},m)} dx_{t-1}$$

- Update step

$$\underbrace{p(x_t, m | z_{1:t}, u_{1:t})}_{bel(x_t,m)} = \eta \cdot \underbrace{p(z_t | x_t, m)}_{Observation\ model} \cdot \underbrace{p(x_t, m | z_{1:t-1}, u_{1:t})}_{\overline{bel}(x_t,m)}$$

Basic Full SLAM

- Prediction step

$$\underbrace{p(x_{1:t}, m | z_{1:t-1}, u_{1:t})}_{\overline{bel}(x_t, m)} = p(x_t | x_{t-1}, u_t) \cdot p(x_{1:t-1}, m | z_{1:t-1}, u_{1:t})$$

$$= \underbrace{p(x_t | x_{t-1}, u_t)}_{Motion\ \ model} \cdot \underbrace{p(x_{1:t-1}, m | z_{1:t-1}, u_{1:t-1})}_{bel(x_{1:t-1}, m)}$$

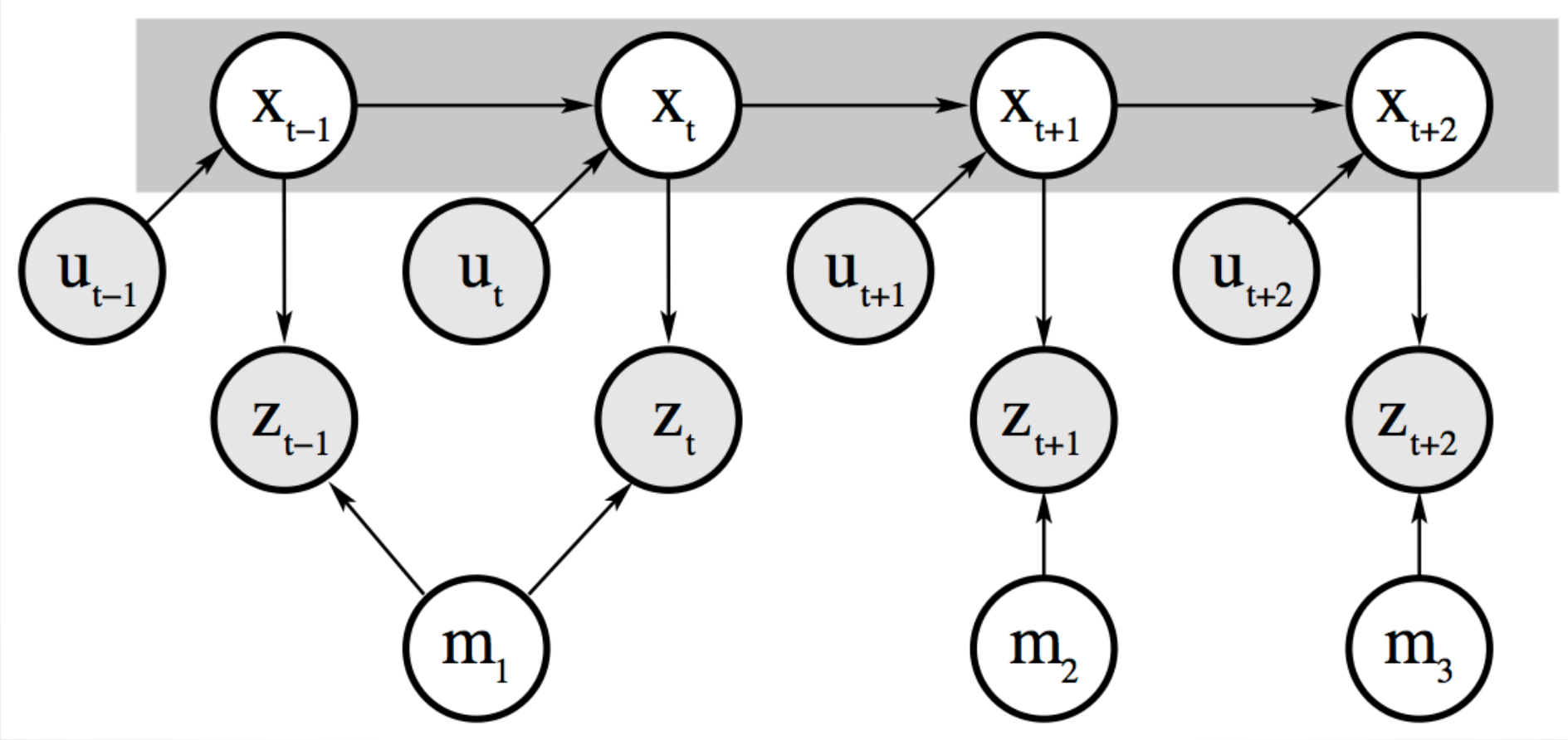
- Update step

$$\underbrace{p(x_{1:t}, m | z_{1:t}, u_{1:t})}_{bel(x_{1:t}, m)} = \eta \cdot \underbrace{p(z_t | x_{1:t}, m)}_{Observation\ \ model} \cdot \underbrace{p(x_{1:t}, m | z_{1:t-1}, u_{1:t})}_{\overline{bel}(x_{1:t}, m)}$$

Basic SLAM

- Typically non-linear observation models
- Can be solved approximately via
  - EKF: in the update step, only need to update a subset of m that can be observed
  - Particle filter: the key idea is that each particle corresponds <one possible state, its own map>
- Need further reduction in computation complexity when the dimension of m is large
  - Sparse EIF SLAM, fast SLAM

Fast SLAM



Fast SLAM

- Recall that the # of particles in PF needs to be large enough to represent the state space
- In SLAM, even with feature-based, the possible map realization is still large; with occupancy map, 2^L possible maps where L is the # of grid points – curse of dimensionality!
- Key insight: if the states x's are known, then mapping is a simpler problem

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

$$p(m|x_{1:t}, z_{1:t}) = \prod_{l=1}^{L} p(m_l|x_{1:t}, z_{1:t})$$

This is the mapping problem!

  - Treat each path as a particle using the full SLAM model
  - Due to independence, we can update each landmark separately – EKF or one particle cloud per landmark → L instead of 2^L complexity

Comments

- These equations/algorithms would not be very meaningful until you try them out in your own applications
- Computation overhead is a big problem with FP
- Many variants exist not covered in the lecture
- SLAM codes available from www.openslam.org

Further Readings

- Sebastian Thrun, Wolfram Burgard, Dieter Fox, "Probabilistic Robotics", the MIT Process
- Cyrill Stachniss' slides & Youtube video on robotic mapping
- Durrant-Whyte, et al. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms & Part II State of the Art
- A good visual for PF https://www.youtube.com/watch?v=aUkBa1zMKv4