AA 274
Principles of Robotic Autonomy

SLAM II: graph-based SLAM and particle-filter SLAM

Stanford University

Autonomous Systems Lab

Today’s lecture

- Aim
  - Learn about additional SLAM techniques, chiefly graph SLAM and fast SLAM

- Readings
  - SNS: 5.8.7-5.8.10
  - S. Thrun, W. Burgard, and D. Fox. Probabilistic robotics. MIT press, 2005. Sections 11.1, 13.1-13.3, 13.5

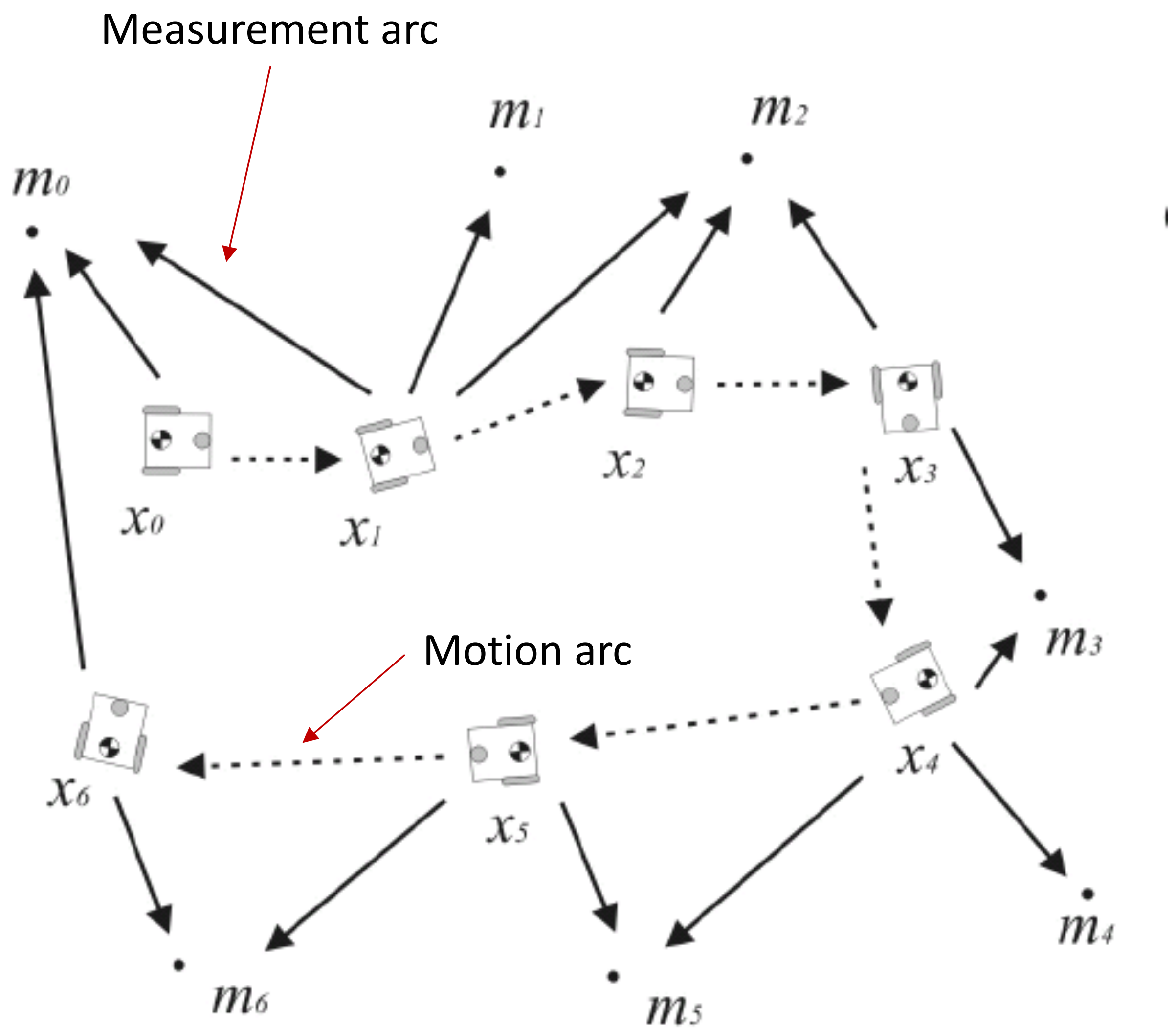
Graph SLAM

- Key idea: interpret the SLAM problem as a sparse graph of nodes and constraints between nodes
- Goal is to solve full-scale SLAM, i.e., estimate

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

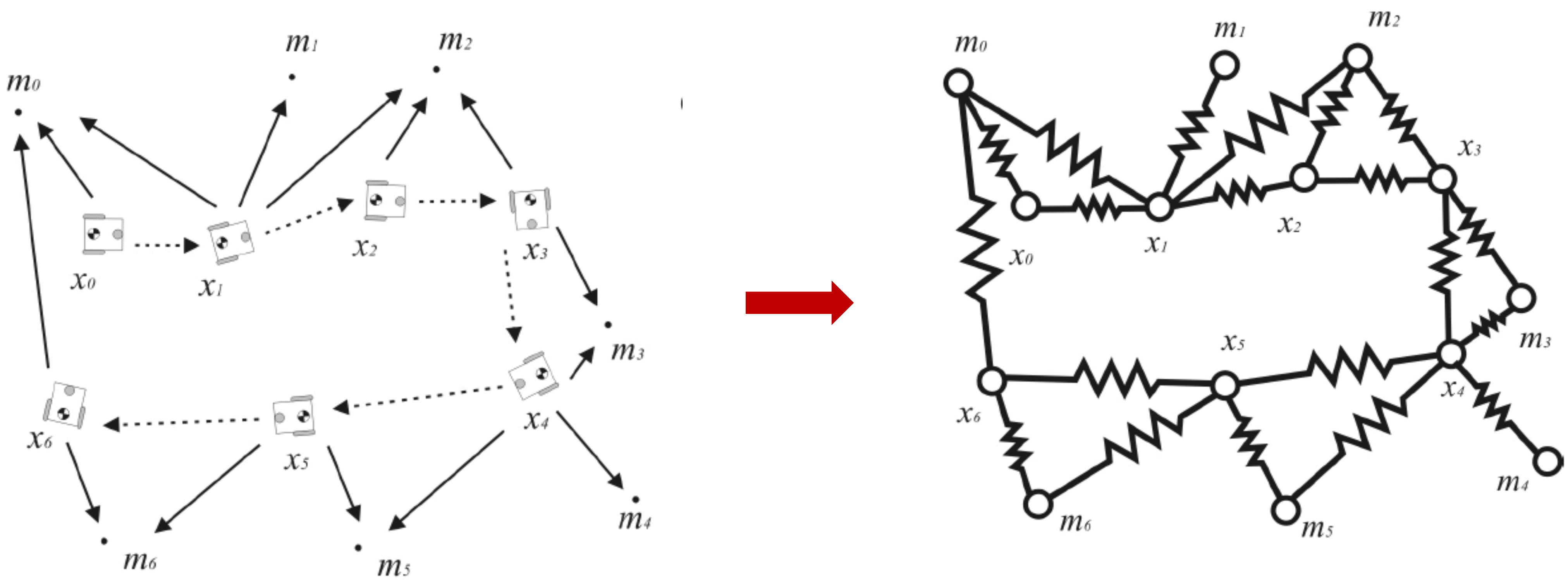
Graph SLAM

- Nodes of the graph are the robot locations and the features in the map
- Constraints are relative positions (1) between consecutive robot poses and (2) among robot and feature locations
- Each edge corresponds to a non-linear constraint, related to the likelihood of the measurement and motion models
- Growing the graph is cheap!



Graph SLAM

- Constraints should be thought as *soft constraints* -> graph should be thought as an elastic net
- SLAM solution is found by computing state of minimal energy of the net



Particle filter SLAM

- Key idea: use particles to approximate the belief, and particle filter to simultaneously estimate the robot path and the map
- Goal is to solve full-scale SLAM, i.e., estimate

$$p(x_{1:t}, m, c_t \mid z_{1:t}, u_{1:t})$$

- Challenge: naïve implementation of particle filter to SLAM is intractable, due to the excessively large number of particles required
- Key insight: knowledge of the robot's true path renders features conditionally independent -> mapping problem can be *factored* into separate problems, one for each feature in the map

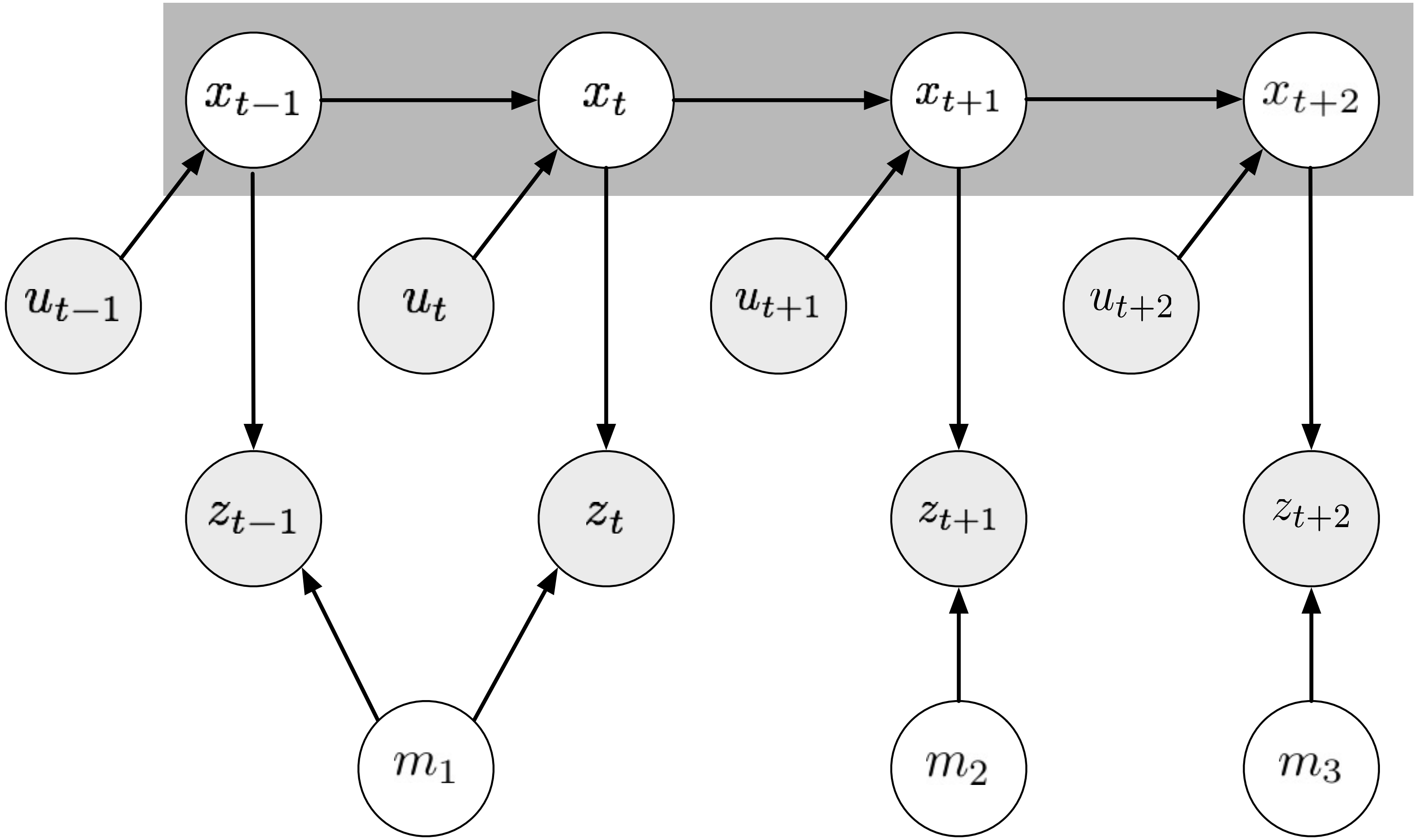
Factoring the posterior

- The key mathematical insight behind particle filter SLAM is the factorization of the posterior

$$\underbrace{p(y_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})}_{\text{SLAM posterior}} = \underbrace{p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})}_{\text{Path posterior}} \prod_{n=1}^{N} \underbrace{p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t})}_{\text{Feature posterior}}$$

Factoring the posterior

- Intuition



Factoring the posterior

- Proof follows from Bayes' rule and induction
- Step #1:

$$
\begin{aligned}
p(y_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) &= p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})\, p(m \mid x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \\
&= p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})\, p(m \mid x_{1:t}, z_{1:t}, c_{1:t})
\end{aligned}
$$

Factoring the posterior

- Step 2.a: assume $c_t \neq n$

$$p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t}) = p(m_n \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})$$

- Step 2.b: assume $c_t = n$

$$\begin{aligned} p(m_{c_t} \mid x_{1:t}, z_{1:t}, c_{1:t}) &= \frac{p(z_t \mid m_{c_t}, x_{1:t}, z_{1:t-1}, c_{1:t})\, p(m_{c_t} \mid x_{1:t}, z_{1:t-1}, c_{1:t})}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \\ &= \frac{p(z_t \mid m_{c_t}, x_t, c_t)\, p(m_{c_t} \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \end{aligned}$$

Factoring the posterior

- Step 3 (induction): assume at time $t-1$ (induction hypothesis)

$$p(m \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1}) = \prod_{n=1}^{N} p(m_n \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})$$

Factoring the posterior

- Then at time t

$$
\begin{aligned}
p(m \mid x_{1:t}, z_{1:t}, c_{1:t}) &= \frac{p(z_t \mid m, x_{1:t}, z_{1:t-1}, c_{1:t})\, p(m \mid x_{1:t}, z_{1:t-1}, c_{1:t})}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \\
&= \frac{p(z_t \mid m, x_t, c_t)\, p(m \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \\
&= \frac{p(z_t \mid m, x_t, c_t)}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \prod_{n=1}^{N} p(m_n \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1}) \\
&= \frac{p(z_t \mid m, x_t, c_t)}{p(z_t \mid x_{1:t}, z_{1:t-1}, c_{1:t})} \underbrace{p(m_{c_t} \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})}_{\text{Steb 2.b}} \prod_{n \neq c_t} \underbrace{p(m_n \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})}_{\text{Steb 2.a}} \\
&= p(m_{c_t} \mid x_{1:t}, z_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t}) = \prod_{n=1}^{N} p(m \mid x_{1:t}, z_{1:t}, c_{1:t})
\end{aligned}
$$

Fast SLAM with known correspondences

- Key idea: exploit factorization result to decompose problem into sub-problems
  - Path posterior is estimated using particle filter
  - Map features are estimated via EKF conditioned on the robot path (one EKF for each feature)

- Accordingly, particles in Fast SLAM are represented as

$$Y_t^{[k]} = \left\langle x_t^{[k]}, \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \ldots, \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \right\rangle$$

Fast SLAM with known correspondences

- Each particle possesses its own set of EKFs!
- In total there are *NM* EKFs
- Filtering involves generating a new particle set Y_t from Y_{t-1} by incorporating a new control u_t and a new measurement z_t with associated correspondence variable c_t
- Update entails three steps
  1. Extend path posterior
  2. Update observed feature estimate
  3. Resample

Step 1: Extending path posterior

- For each particle $Y_t^{[k]}$, sample pose x_t according to motion posterior

$$x_t^k \sim p(x_t \mid x_{t-1}^k, u_t)$$

- Sample $x_t^{[k]}$ is then concatenated with previous poses $x_{1:t-1}^{[k]}$

Step 2: updating observed feature estimate

- This step entails updating the posterior over the feature estimates
- If $c_t \neq n$

$$\left\langle \mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]} \right\rangle = \left\langle \mu_{n,t-1}^{[k]}, \Sigma_{n,t-1}^{[k]} \right\rangle$$

- If $c_t = n$

$$p(m_{c_t} \mid x_{1:t}, z_{1:t}, c_{1:t}) = \eta \, p(z_t \mid m_{c_t}, x_t, c_t) \, p(m_{c_t} \mid x_{1:t-1}, z_{1:t-1}, c_{1:t-1})$$

$$\sim \mathcal{N}(\mu_{n,t-1}^{[k]}, \Sigma_{n,t-1}^{[k]})$$

Step 2: updating observed feature estimate

- To ensure that the new estimate is Gaussian as well, measurement model is linearized as usual

$$h(m_{c_t}, x_t^{[k]}) \approx h(\mu_{c_t,t-1}^{[k]}, x_t^{[k]}) + \underbrace{h'(\mu_{c_t,t-1}^{[k]}, x_t^{[k]})}_{:=H_t^{[k]}}(m_{c_t} - \mu_{c_t,t-1}^{[k]})$$

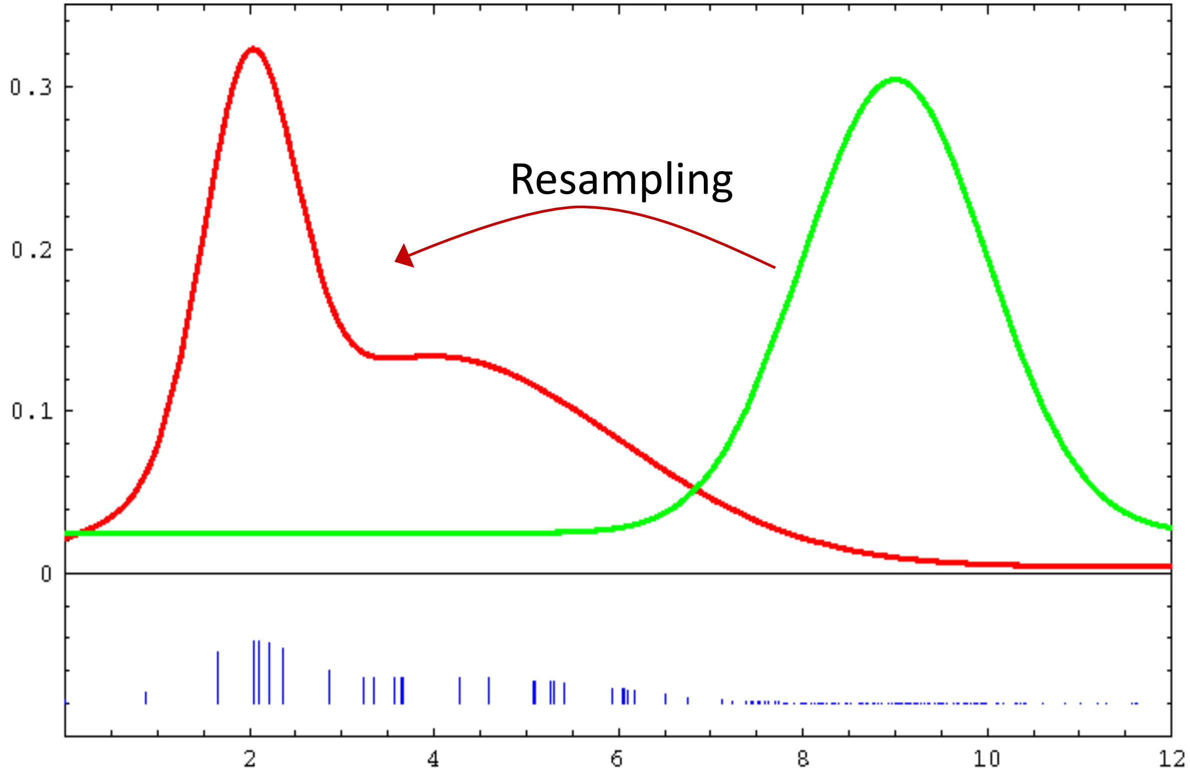
- Mean and covariance are then obtained as per standard EKF

$$K_t^{[k]} = \Sigma_{c_t,t-1}^{[k]}[H_t^{[k]}]^T \, (H_t^{[k]}\Sigma_{c_t,t-1}^{[k]}[H_t^{[k]}]^T + Q_t)^{-1}$$
$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + K_t^{[k]}(z_t - \hat{z}_t^{[k]})$$
$$\Sigma_{c_t,t}^{[k]} = (I - K_t^{[k]} H_t^{[k]})\Sigma_{c_t,t-1}^{[k]}$$

Step 3: resampling

- Step 1 generates pose x_t only in accordance with the most recent control u_t, paying no attention to the measurement z_t

- Goal: resample particles to correct for this mismatch



Step 3: resampling

- How do we find the weights?
- Path particles at this stage are distributed according to

$$p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1}) = p(x_t \mid x_{t-1}^{k}, u_t)\, p(x_{1:t-1}^{[k]} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$$

Sampling distribution

Distribution of path particles in $Y_{t-1}^{[k]}$

- The target distribution takes into account z_t, along with c_t

$$p(x_{1:t}^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t})$$

Step 3: resampling

- Importance factor is then given by

$$
\begin{aligned}
w_t^{[k]} &= \frac{p(x_{1:t}^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\
&= \frac{\eta\, p(z_t \mid x_{1:t}^{[k]}, z_{1:t-1}, u_{1:t}, c_{1:t}) p(x_{1:t}^{[k]} \mid , z_{1:t-1}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\
&= \frac{\eta\, p(z_t \mid x_t^{[k]}, c_t) p(x_{1:t}^{[k]} \mid , z_{1:t-1}, u_{1:t}, c_{1:t-1})}{p(x_{1:t}^{[k]} \mid z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\
&= \eta\, p(z_t \mid x_t^{[k]}, c_t)
\end{aligned}
$$

Step 3: resampling

- To derive an (approximate) close-form expression for $w_t^{[k]}$, one can then apply the total probability law along with a linearization of the measurement model to obtain

$$w_t^{[k]} = \eta \det(2\pi Q_t^{[k]})^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_t^{[k]})[Q_t^{[k]}]^{-1}(z_t - \hat{z}_t^{[k]})\right\}$$

$$Q_t^{[k]} = [H_t^{[k]}]^T \Sigma_{n,t-1}^{[k]} H_t^{[k]} + Q_t$$

Fast Slam algorithm

- Key fact: only the most recent pose is used in the process of generating a new particle at time t!
- One can show that the complexity of an entire update requires $O(M \log N)$

Data: Y_{t-1}, u_t, z_t, c_t
Result: Y_t
for $k = 1$ **to** M **do**
  $x_t^k \sim p(x_t \mid x_{t-1}^k, u_t)$;
  $j = c_t$;
  if *feature j never seen before* **then**
    initialize feature
  else
    $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$;
    calculate Jacobian H;
    $Q = H\Sigma_{j,t-1}^{[k]} H_t^T + Q_t$;
    $K = \Sigma_{j,t-1}^{[k]} H^T Q^{-1}$;
    $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$;
    $\Sigma_{j,t}^{[k]} = (I - KH)\Sigma_{j,t-1}^{[k]}$;
    $w^{[k]} = \det(2\pi Q)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z})Q_t^{-1}(z_t - \hat{z})\right\}$;
  end
  for *all other features $n \neq j$* **do**
    $\left\langle \mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]} \right\rangle = \left\langle \mu_{n,t-1}^{[k]}, \Sigma_{n,t-1}^{[k]} \right\rangle$;
  end
  $Y_t = \emptyset$;
end
for $i = 1$ **to** M **do**
  Draw k with probability $\propto w^{[k]}$;
  Add $\left\langle x_t^{[k]}, \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \ldots, \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \right\rangle$ to Y_t;
end
Return Y_t

Fast SLAM with unknown correspondences

- Key advantage of particle filters: each particle can rely on its own, local data association decisions!
- Key idea: per-particle data association generalizes the per-filter data association to individual particles
- Each particle maintains *a local set* of data association variables, $\hat{c}_t^{[k]}$
- Data association is solved, as usual, via maximum likelihood estimation

$$\hat{c}_t^{[k]} = \arg\max_{c_t} p(z_t \mid c_t, \hat{c}_{1:t-1}^{[k]}, x_{1:t}^{[k]}, z_{1:t-1}, u_{1:t})$$

Computed, as usual, via total probability law + linearization

Summary: Gaussian filtering (EKF, UKF)

- Key ideas:
  - Represent a belief with a Gaussian distribution
  - Assume all uncertainty sources are Gaussian
- Pros:
  - Runs online
  - Well understood
  - Works well when uncertainty is low
- Cons:
  - Unimodal estimate
  - States must be well approximated by a Gaussian
  - Works poorly when uncertainty is high

Summary: graph-theoretical approaches

- Key ideas:
  - Interpret the SLAM problem as an inference problem on a graph
  - Assume all uncertainty sources are Gaussian
- Pros:
  - Best possible (most likely) estimate given the data and models
  - Exploitation of matrix sparsity leads to efficient solutions
- Cons
  - Can be computationally demanding
  - Difficult to provide online estimates for a controller

Summary: particle filter approaches

- Key ideas:
  - Approximate belief with particles
  - Use particle filters to perform inference
- Pros:
  - Can handle “any” noise distribution
  - Relatively easy to implement
  - Naturally represents multimodal beliefs
  - Robust to data association errors
- Cons:
  - Does not scale well to large dimensional problems
  - Might require many particles for good convergence
  - Might have issues with loop closure

Final considerations

- A recent overview of SLAM (with strong focus on graph SLAM): C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age." IEEE Transactions on Robotics 32, no. 6 (2016): 1309-1332.
- Popular open-source software packages
  - https://www.openslam.org/: contains a comprehensive list of SLAM software
  - http://www.robots.ox.ac.uk/~gk/PTAM/: visual SLAM
  - https://developers.google.com/tango/developer-overview: project Tango
  - http://www.rawseeds.org/home/: collection of benchmarked datasets
- Trends: from the classical age, to the algorithmic-analysis age, to the robust perception age

Next time