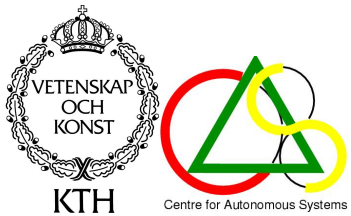
EL2320 - Simultaneous Localization and Mapping SLAM, Rao-Blackwellized Particle Filters (FastSLAM)

John Folkesson

Rao-Blackwellized Particle Filters (FastSLAM)
(Chap 10 and 13.0-13.5 in Thrun)



Simultaneous Localization and Mapping SLAM

- Chickens and eggs.
- Online SLAM $p(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Full Slam $p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$
- Feature based SLAM: the map contains location, (shape, orientation...) of features (walls, poles, points, lines, ...)
- Grid based SLAM: map is a grid of cells with a probability of each cell being occupied (occupancy grid).
- Key frame based SLAM: Map is the raw sensor measurements or some composition of sensor measurments which then are compared to current sensor readings. Camera images (FAB-MAP bag of words), or Laser scan matching (ICP algorithm). Some of these are *topological maps*

EKF SLAM

- Feature based SLAM
- Gaussian noise.
- Adding the map to the pose to form a long state vector and large Covariance Matrix.
- Must remember that the EKF algorithm is $O(n^2)$ wher n is number of state dimensions. This is so as there are n^2 numbers to update in the Covariance. (could be worse but is not for EKF SLAM).

Gaussians

x and **m** are Gaussian vector variables and A, B and C are matricies.

$$p(\mathbf{x},\mathbf{m}) = G\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{m}\end{pmatrix}, \begin{pmatrix}\mu_x\\ \mu_m\end{pmatrix}, \begin{pmatrix}A & C\\ C^T & B\end{pmatrix}\right)$$

$$p(\mathbf{x}) = \int_{-\infty}^{\infty} p(\mathbf{x},\mathbf{m})d\mathbf{m} = G(\mathbf{x},\mu_x,A)$$

$$p(\mathbf{m}|\mathbf{x}) = p(\mathbf{m},\mathbf{x})/p(\mathbf{x}) = G(\mathbf{m},\mu_m + \Gamma(x-\mu_x),\tilde{B})$$

$$\Gamma = C^T A^{-1} \qquad \tilde{B} = B - C^T A^{-1} C)$$

Proofs involve: Matrix algebra and 'completing the square'

$$\alpha x^2 + \beta xy + \gamma = \alpha(x + \frac{\beta}{2\alpha}y)^2 + \gamma - (\frac{\beta}{2\alpha}y)^2$$

only with matricies instead of scalars.

EKF SLAM

$$p(\mathbf{x}, \mathbf{m}) = G\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{m} \end{pmatrix}, \begin{pmatrix} \mu_x \\ \mu_m \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix}\right)$$

$$p(\mathbf{x}, \mathbf{m}) = p(\mathbf{m}|\mathbf{x})p(\mathbf{x})$$

$$p(\mathbf{x}, \mathbf{m}) = G(\mathbf{m}, \mu_m + \Gamma(x - \mu_x), \tilde{B})G(\mathbf{x}, \mu_x, A)$$

EKF SLAM Feature Initialization

- SEE Slides 15 and 16 EL2320.5.pdf
- Important in practice to observe a new feature for several measurements before adding it to the state (can not easily remove it later).
- Thus one has a nursery of immature features that each have there own little EKF linearized assuming that the pose state is correct.
- These then can be added to the map using a single 'pseudo-measurement' of the feature relative to the current robot pose derived from the little EKF state mean and covariance.

EKF SLAM Data Association

- The data association being single modal (one hypothesis must be chosen) makes EKF slam 'brittle', that is not robust.
- typically use ML data association.
- Try to avoid ambiguities:
  - separated features (spatially).
  - signatures, distinguishable features.
  - never get lost (always see some features)
- Fundamental problem is that lots of features give robust estimates but poor computation time.

Rao-Blackwellize

Rao-Blackwellized Particle filters factor the posterior into one part to be represented by a parametric PDF (such as Gaussian) and another part to be represented by a particle filter.

This might be good if the dimension of the full state space is too large for particle samples to cover it.

$$p(\mathbf{x}_{g,1:t}\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{g,1:t}|\mathbf{x}_{p,1:t}\mathbf{z}_{1:t}, \mathbf{u}_{1:t})p(\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

Here $\mathbf{x}_{g,1:t}$ is a symbol to represent the entire path $1:t$ in the [g space, where g designates the part of the state represented by the parametric PDF.

Similarly $\mathbf{x}_{p,1:t}$ is the path in the particle part of the state.
So we need not yet make any Markov assumptions.

Rao-Blackwellize

$$p(\mathbf{x}_{g,1:t}\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = p(\mathbf{x}_{g,1:t}|\mathbf{x}_{p,1:t}\mathbf{z}_{1:t},\mathbf{u}_{1:t})p(\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t},\mathbf{u}_{1:t})$$

In particular we can write the particle filter part as:

$$p(\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t},\mathbf{u}_{1:t}) = \sum_{i=1}^{M} w^i \prod_{j=1}^{t} K(\mathbf{x}_{p,j} - \mathbf{x}^i_{p,j}).$$

Where K is some Kernel function. Then if the Kernels are *narrow* realtive to changes in $p(\mathbf{x}_{g,1:t}|\mathbf{x}_{p,1:t}\mathbf{z}_{1:t}\mathbf{u}_{1:t})$. And, for example, the parametric part is Gaussian:

$$p(\mathbf{x}_{g,1:t}\mathbf{x}_{p,1:t}|\mathbf{z}_{1:t}\mathbf{u}_{1:t}) \approx \sum_{i=1}^{M} w^i G(\mathbf{x}_{g,1:t}, \mu^i_t, \Sigma^i_t) \prod_{j=1}^{t} K(...).$$

Here there is a separate Gaussian estimated fof each particle of the representation. That is given the whole path for the particles.

If the Gaussian part is static then, $\mathbf{x}_{g,1:t} = \mathbf{x}_g$ That is we need not estimate a path for numbers that have no dynamics.

For the SLAM problem the factorization can go even further due to the observation:

The mapping problem for each feature given the robot path is independent of the other features.

This then makes the Gaussian over the m features in the map factor into a product of m simple Gaussians, one for each feature.

FastSLAM

We set $\mathbf{x}_p$ to be the part of the state representing the robot postition and orientation (the robot's *pose*).

We then set $\mathbf{x}_g$ to the positions of the m features, (ie. the map).

Then computing the Gaussian parameters given the path and the measurements breaks down to m separate and independent estimates for each feature where only the measurements of that feature enter into the estimate.

These seperate Gaussian estimates can be done using light 'EKF's' for each feature. However, these EKF have no predict and are really just summing up Gaussian measurements using the EKF linearization for the update step.

FastSLAM, Notation Sync

We sync with the book notataion so

$\mathbf{y}_{1:t} = (\mathbf{x}_{1:t}, \mathbf{m})$

will replace

$(\mathbf{x}_{p,1:t}\mathbf{x}_g)$

That is we have a specific case or a robot path $\mathbf{x}_{1:t}$ and a map of features $\mathbf{m}$ being estimated.

When dealing with particles we will shorten this to $\mathbf{x}_t^i$ where all the poses of the path are implied.

For some odd reason the book switchs to use $g(\mathbf{x})$ for the measurement function instead of $h(\mathbf{x})$ as in earlier chapters. We stick with h.

For now I will just be assuming we know the data association so I will leave off the $c_{1:t}$.

FastSLAM, Iteration Start State

FastSLAM starts with the particles distributed according to the $p(\mathbf{x}_{1:t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$ distribution.

That is the weights are all $w^i = M^{-1}$

It restores this by resampling at the end of each iteration.

FastSLAM

for i=1 to M

1. Sample an extension to the path of particle i, $\mathbf{x}_t^i \sim p(x_t | x_{t-1}^i, u_t)$
2. update the gaussian of any observed features using EKF, (after data association) (next few slide).
3. compute the importance weight, w^i (next next slides).
4. Then resample using the w^i.

FastSLAM, Feature Update

It is essentially an EKF update but here only the $\mathbf{m}$ is included in the state, the robot pose is known (per particle).

$$
\begin{aligned}
p(\mathbf{m}|\mathbf{x}^i_{1:t}, \mathbf{z}_{1:t}) &\propto p(\mathbf{z}_t|\mathbf{x}^i_t, \mathbf{m}) p(\mathbf{m}|\mathbf{x}^i_{1:t-1}, \mathbf{z}_{1:t-1}) \\
&\propto G(\mathbf{z}_t, \mathbf{h}(\mathbf{x}^i_t, \mathbf{m}), R_t) G(\mathbf{m}, \mu^i_{t-1}, \Sigma^i_{t-1}) \\
S^i_t &= H^i_t \Sigma^i_{t-1} (H^i_t)^T + R_t. \\
K^i_t &= \Sigma^i_{t-1} (H^i_t)^T (S^i_t)^{-1} \\
\mu^i_t &= \mu^i_{t-1} + K^i_t (\mathbf{z}_t - \hat{\mathbf{z}}^i_t) \\
\Sigma^i_t &= \Sigma^i_{t-1} - K^i_t H^i_t \Sigma^i_{t-1} \\
H^i_t &= (\frac{\partial}{\partial m_1} \mathbf{h}(\mathbf{x}^i_t, \mathbf{m}) \mid_{\mathbf{m}=\mu^i_{t-1}}, \ldots)
\end{aligned}
$$

FastSLAM, Importance Weight

The importance weight $w^i = G(\mathbf{z}_t, \hat{\mathbf{z}}_t^i, S_t^i)$

$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}) + \delta$ where $\delta \sim N(0, R_t)$

$\hat{\mathbf{z}}_t^i$ is the expected measurement given the i^{th} particle's Gaussian map at $t-1$ and its pose at t.

$\hat{\mathbf{z}}_t^i = h(\mathbf{x}_t^i, \mu_{t-1}^i)$

$S_t^i = H_t^i \Sigma_{t-1}^i (H_t^i)^T + R_t.$

So H^i is the Jacobian of h wrt the features $\mathbf{m}$, evaluated at $(\mathbf{x}_t^i, \mu_{t-1}^i)$. That is treating the robot pose $\mathbf{x}_t$ as fixed and known to be $\mathbf{x}_t^i$.

Σ^i and μ^i are the Gaussian parameters of the i^{th} 'map'.

FastSLAM, Importance Weight, Gory details

$w^i = G(\mathbf{z}_t, \hat{\mathbf{z}}_t^i, S_t^i) =$ target distribution / proposal distribution.

$$
\begin{aligned}
w^i &= \frac{p(\mathbf{x}_t^i | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_t^i | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} \\
&= \frac{p(\mathbf{x}_t^i | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{\int p(\mathbf{x}_t^i | \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) Bel(\mathbf{x}_{t-1}) dx_{t-1}} \\
&\propto \frac{\int p(\mathbf{z}_t | \mathbf{x}_t^i, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t^i | \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) Bel(x_{t-1}) dx_{t-1}}{\int p(\mathbf{x}_t^i | \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) Bel(x_{t-1}) dx_{t-1}} \\
&\propto p(\mathbf{z}_t | \mathbf{x}_t^i, \mathbf{z}_{1:t-1}) \\
&\propto \int p(\mathbf{z}_t | \mathbf{m}, \mathbf{x}_t^i, \mathbf{z}_{1:t-1}) p(\mathbf{m} | \mathbf{x}_t^i, \mathbf{z}_{1:t-1}) dm \\
&\propto \int p(\mathbf{z}_t | \mathbf{m}, \mathbf{x}_t^i, \mathbf{z}_{1:t-1}) p(\mathbf{m} | \mathbf{x}_{t-1}^i, \mathbf{z}_{1:t-1}) dm \\
&\propto \int G(\mathbf{z}_t, \mathbf{h}(\mathbf{x_t^i}, \mathbf{m}), R_t) G(\mathbf{m}, \mu_{t-1}^i, \Sigma_{t-1}^i) dm
\end{aligned}
$$

FastSLAM, Importance Weight, end of gore

$$w_i \propto \int G(\mathbf{z}_t, \mathbf{h}(\mathbf{x_t^i}, \mathbf{m}), R_t) G(\mathbf{m}, \mu_{t-1}^i, \Sigma_{t-1}^i) dm$$

Woodbury Matrix Identity:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Matrix Determinate Lemma

$$det(A + UWV^T) = det(W^{-1} + V^T A^{-1} U) det(W) det(A)$$

Gives us the $w^i = G(\mathbf{z}_t, \hat{\mathbf{z}}_t^i, S_t^i)$

$$S_t^i = (H_t^i)^T \Sigma_{t-1}^i H_t^i + R_t.$$

FastSLAM, 2.0

Again we can have the same problem as MCL with good sensors. The sensors might give us an accurate localization to a place with few or no particles.

The solution is again to change the proposal distribution.

$\mathbf{x}_t^i \sim p(\mathbf{x}_t | x_{1:t-1}^i, u_{1:t}, z_{1:t})$

So our sample step now is dependent on our measurement $\mathbf{z}_t$

This leads to a more complicated formula for the sampling distribution and the importance factors.

They again require linearizing the measurement function but this time we need Jacobians both in the feature and the pose parts.

The feature update works the same way because we are again given the path.

FastSLAM, Unknown Data Association

It is the fact that FastSLAM can keep track of different data association hypothesises that is the key advantage of it over EKF SLAM.

This however requires data association to be done on a per particle basis.

The association can be done my either maximizing or sampling the likelihood as a function of the association.

FastSLAM, Likelihood of a Data Association

$$
\begin{aligned}
& p(\mathbf{z}_t | c_t, \mathbf{x}_t^i, \mathbf{z}_{1:t-1}, u_{1:t}) \\
&\quad = \int p(\mathbf{z}_t | \mathbf{m}, c_t, \mathbf{x}_t^i, \mathbf{z}_{1:t-1}, u_{1:t}) p(\mathbf{m}_t | c_t, \mathbf{x}_t^i, \mathbf{z}_{1:t-1}, u_{1:t}) dm \\
&\quad = \frac{1}{\sqrt{2\pi S_t^i}} \exp -\frac{1}{2} \Delta \mathbf{z}_t^i (S_t^i)^{-1} \Delta \mathbf{z}_t^i \\
\Delta \mathbf{z}_t^i &= \mathbf{z}_t - \mathbf{h}(\mathbf{x}_t^i, \mu_t^i) \\
S_t^i &= R_t + H_t^i \Sigma_{t-1}^i (H_t^i)^T
\end{aligned}
$$

FastSLAM, Adding Features

A new feature is created when a measurement can not be explained well by the exisisting map for the given particle.

So different particles can have different numbers of features.

So if the maximum likelihood data association has a likelihood below some threshold p_0 one creates a new feature instead of using that association.

Another way of thinking is that there is a likelihood of a new feature of p_0.

If sampling associations then one includes this in the sampled distribution.

FastSLAM, Adding Features

Often one does not add a new feature immediately but rather waits until it is observed several times.

This is known as provisional feature list.

One then must maintain a separate EKF for this feature in order to evaluate the data association likelihood.

This then looks like any other feature except it does not get to influence the importance factor until it is taken off the list.

Aside from counting the number of observations of features on the list one might count the number of non-observations or negative information.

These then can be combined into criterior for discarding or adding the feature to the map.

FastSLAM, Efficiency

The complexity is at first sight $O(MN)$ where M is the number of particles and N is the number of features.

However most of the features do not change from one iteration to the next.

It is only the observed features that change and that influence the importance factors.

One must do the data association over all features but one can organize the features so that only the nearby ones are checked.

By using clever data sturctures one avoids excessive recopying and long searches through the feature list.

All this leads to an $O(Mln(N))$ complexity which is a fairly important advantage compared to EKF SLAM, $O(N^2)$.

FastSLAM, Loop Closure

Loop Closure occurs when the robot returns to a previously visited region by a different path than back tracking.

This presents a challenge as the errors in the map and the pose may make the data association to fail.

It also can be impossible to find a particle with an estimate that is consistent with true data association.

With particles filters, if there is no particle near the right answer there never will be.

The problem is simply that the particle filter is simply pretending to represent the posteori (not everyone agrees with this). The longer the path the larger the space of possible paths and it simply get impossible.

FastSLAM vs EKF SLAM

There are 2 versions of the SLAM problem, the on-line and the Full SLAM problem.

They differ in that the full SLAM problem includes the full robot path, that is it estimates the posteriori distribution of the pose of the robot along the entire path given all the measurements.

FastSLAM seems to do this for an infinite number of particles M, but for finite M this claim weakens as the length of the path grows.

The on-line SLAM problem is an incremental approach where one only estimates the most rescent robot pose along with the map distribution given all the measurements.

EKF SLAM does this explictly by marginalizing out all the previous time's robot poses.

FastSLAM vs EKF SLAM

The rather unexpected quality of EKF SLAM is seen when closing a loop.

When the robot returns to a previously seen feature the robot's pose will move to a position consistent with the latest measurement.

Also the covariance of the pose will shrink to about that of the re-observed feature plus sensor noise.

Also the features moving back along the robot path will move to consistent locations and have their covariances shrink.

This is in sharp constrast to FastSLAM which can not close a loop if it lacks the right particle.

FastSLAM vs EKF SLAM

How can EKFSLAM do this loop closing.

The key is the off diagonal elements of the Covariance Matrix.

These say how much, for example, feature 1's x component should change if we change the robot's current x component.

Also how much the covariance of feature 1 depends on the uncertainty in the robot's x.

All this is only as good as the linearization. It assumes everthing moves along lines.

If the true correlations move on arcs this will not be quite correct.

FastSLAM vs EKF SLAM

EKF SLAM must make one single decision on data association.

FastSLAM can maintain many hypothesises.

This makes EKF SLAM 'brittle' that is not robust.

Also EKF SLAM has an $O(N^2)$ complexity in the number of features.

This forces us to limit the number of features we add to the map. That is we only take the 'best'.

Having fewer features makes the data association brittleness even worse.