SEIF, EnKF, EKF SLAM, Fast SLAM, Graph SLAM

Pieter Abbeel
UC Berkeley EECS

Many slides adapted from Thrun, Burgard and Fox, Probabilistic Robotics

Information Filter

- From an analytical point of view == Kalman filter
- Difference: keep track of the inverse covariance rather than the covariance matrix [matter of some linear algebra manipulations to get into this form]
- Why interesting?
  - Inverse covariance matrix = 0 is easier to work with than covariance matrix = infinity (case of complete uncertainty)
  - Inverse covariance matrix is often sparser than the covariance matrix --- for the "insiders": inverse covariance matrix entry (i,j) = 0 if x_i is conditionally independent of x_j given some set $\{x_k, x_l, \ldots\}$
  - Downside: when extended to non-linear setting, need to solve a linear system to find the mean (around which one can then linearize)
  - See Probabilistic Robotics pp. 78-79 for more in-depth pros/cons and Probabilistic Robotics Chapter 12 for its relevance to SLAM (then often referred to as the "sparse extended information filter (SEIF)")

Ensemble Kalman filter (enKF)

- Represent the Gaussian distribution by samples
  - Empirically: even 40 samples can track the atmospheric state with high accuracy with enKF
  - <-> UKF: 2 * n sigma-points, n = 10^6 + then still forms covariance matrices for updates
- The intellectual innovation:
  - Transforming the Kalman filter updates into updates which can be computed based upon samples and which produce samples while never explicitly representing the covariance matrix

KF | enKF

Keep track of μ, Σ

Prediction:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Return μ_t, Σ_t

Keep track of ensemble [x_1, …, x_N]

Can update the ensemble by simply propagating through the dynamics model + adding sampled noise

?

enKF correction step

- KF:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

- Current ensemble $X = [x_1, \ldots, x_N]$
- Build observations matrix $Z = [z_t+v_1 \ \ldots \ z_t+v_N]$ where v_i are sampled according to the observation noise model
- Then the columns of

  $X + K_t(Z - C_t X)$

  form a set of random samples from the posterior

Note: when computing K_t, leave Σ_t in the format

$\Sigma_t = [x_1-\mu_t \ \ldots \ x_N-\mu_t]\ [x_1-\mu_t \ \ldots \ x_N-\mu_t]^T$

How about C?

- Indeed, would be expensive to build up C.
- However: careful inspection shows that C only appears as in:
  - $C X$
  - $C \Sigma C^T = C X X^T C^T$
- → can simply compute h(x) for all columns x of X and compute the empirical covariance matrices required
- [details left as exercise]

Are the columns of $X + K_t(Z - C_t X)$ really samples from $N(\mu_t, \Sigma_t)$?

one column: $y^{(i)} = x^{(i)} + K_t(z_t + v^{(i)} - C_t x^{(i)})$

where $x^{(i)} \sim N(\bar{\mu}_t, \bar{\Sigma}_t)$ $\quad v^{(i)} \sim N(0, Q_t)$

① $$E[x^{(i)}] = \bar{\mu}_t + K_t(z_t + 0 - C_t \cdot \bar{\mu}_t)$$
$$= \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$$
$$= \mu_t \quad ✓$$

② $$E[(y^{(i)} - Ey^{(i)})(y^{(i)} - Ey^{(i)})^T]$$
$$= E\Big[\big((x^{(i)} + K_t(z_t + v^{(i)} - C_t x^{(i)})) - (\bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t))\big) \big(\underline{\qquad\qquad\qquad}\big)^T\Big]$$
$$= E\Big[\big((I - K_t C_t)(x^{(i)} - \bar{\mu}_t) + K_t v^{(i)}\big)\big(\underline{\qquad\qquad}\big)^T\Big]$$

$v^{(i)}$ and $x^{(i)}$ independent

$$= E[(I - K_t C_t)(x^{(i)} - \bar{\mu}_t)(x^{(i)} - \bar{\mu}_t)^T (I - K_t C_t)^T]$$
$$+ E[K_t v^{(i)} v^{(i)T} K_t^T]$$
$$= (I - K_t C_t)\bar{\Sigma}_t (I - K_t C_t)^T + K_t Q_t K_t^T$$
$$= \bar{\Sigma}_t + K_t \underbrace{C_t \bar{\Sigma}_t C_t^T} K_t^T - K_t C_t \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T K_t^T + K_t Q_t K_t^T$$

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

$$= \bar{\Sigma}_t + \cancel{\bar{\Sigma}_t C_t^T K_t^T} - K_t C_t \bar{\Sigma}_t - \cancel{\bar{\Sigma}_t C_t^T K_t^T}$$
$$= \bar{\Sigma}_t - K_t C_t \bar{\Sigma}_t = \Sigma_t \quad \text{Q.E.D.} \quad \blacksquare$$

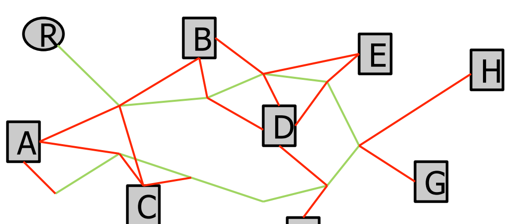
References for enKF

- Mandel, 2007 "A brief tutorial on the Ensemble Kalman Filter"
- Evensen, 2009, "The ensemble Kalman filter for combined state and parameter estimation"

KF Summary

- Kalman filter exact under linear Gaussian assumptions
- Extension to non-linear setting:
  - Extended Kalman filter
  - Unscented Kalman filter
- Extension to extremely large scale settings:
  - Ensemble Kalman filter
  - Sparse Information filter
- Main limitation: restricted to unimodal / Gaussian looking distributions
- Can alleviate by running multiple XKFs + keeping track of the likelihood; but this is still limited in terms of representational power unless we allow a very large number of them

EKF/UKF SLAM



- State: (n_R, e_R, θ_R, n_A, e_A, n_B, e_B, e_C, n_D, e_D, n_E, e_E, n_F, e_F, n_G, e_G, n_H, e_H)
  - Now map = location of landmarks (vs. gridmaps)
- Transition model:
  - Robot motion model; Landmarks stay in place

Simultaneous Localization and Mapping (SLAM)

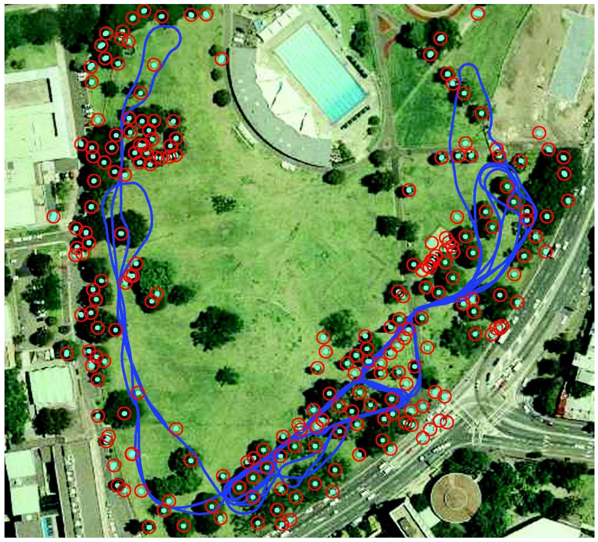
- In practice: robot is not aware of all landmarks from the beginning
- Moreover: no use in keeping track of landmarks the robot has not received any measurements about

→ Incrementally grow the state when new landmarks get encountered.

Simultaneous Localization and Mapping (SLAM)

- Landmark measurement model: robot measures [x_k; y_k], the position of landmark k expressed in coordinate frame attached to the robot:
  - $h(n_R, e_R, \theta_R, n_k, e_k) = [x_k; y_k] = R(\theta) ([n_k; e_k] - [n_R; e_R])$
- Often also some odometry measurements
  - E.g., wheel encoders
  - As they measure the control input being applied, they are often incorporated directly as control inputs (why?)

Victoria Park Data Set



[courtesy by E. Nebot]

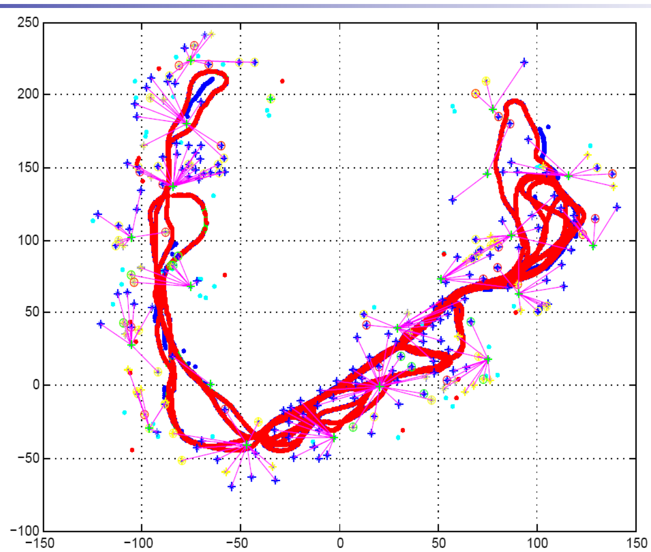
Victoria Park Data Set Vehicle

[courtesy by E. Nebot]

Data Acquisition

[courtesy by E. Nebot]

Estimated Trajectory



[courtesy by E. Nebot]

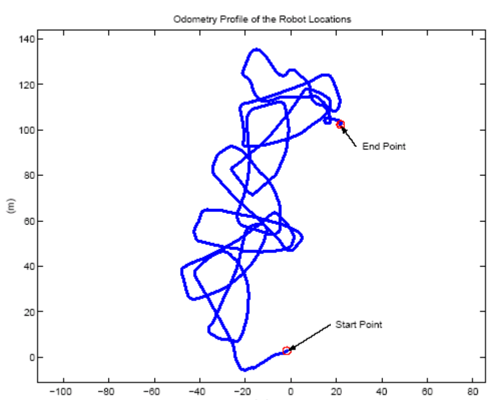
18

EKF SLAM Application

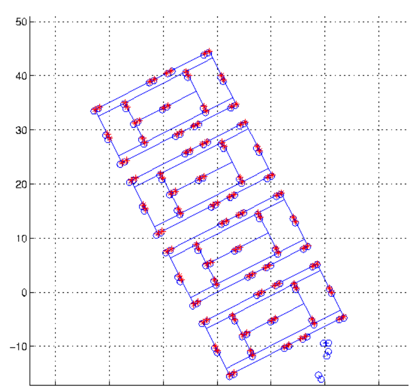
[courtesy by J. Leonard]

19

EKF SLAM Application



odometry

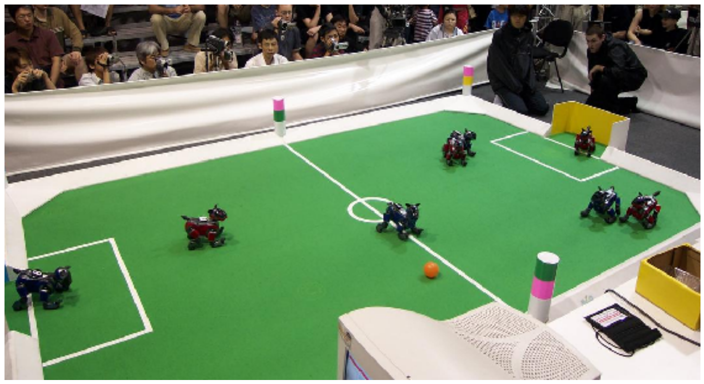


estimated trajectory

[courtesy by John Leonard]

20

Landmark-based Localization



21

EKF-SLAM: practical challenges

- Defining landmarks
  - Laser range finder: Distinct geometric features (e.g. use RANSAC to find lines, then use corners as features)
  - Camera: “interest point detectors”, textures, color, …
- Often need to track multiple hypotheses
  - Data association/Correspondence problem: when seeing features that constitute a landmark --- Which landmark is it?
  - Closing the loop problem: how to know you are closing a loop?
  - → Can split off multiple EKFs whenever there is ambiguity;
  - → Keep track of the likelihood score of each EKF and discard the ones with low likelihood score
- Computational complexity with large numbers of landmarks.

KF over very large state spaces

- High-dimensional ocean and atmospheric circulation models (10^6 dimensional state space)
- SLAM with 10^6 landmarks

→ Becomes computationally very challenging to work with the $10^6 \times 10^6$ covariance matrix (terabytes!)
  - → In SLAM community: information filter which keeps tracks of the inverse covariance matrix, which can often be well approximated by a sparse matrix
  - → In civil engineering community: ensemble Kalman filter, with applications often being in tracking systems described by partial differential equations

Fast SLAM

- Rao-Blackwellized particle filter
  - Robot state = x, y, θ (just like gMapping)
  - Map = Landmark based (vs. map = gridmap for gMapping)

- Key observation (why Rao Blackwellization is so useful):
  - Location of landmark i is independent of location of landmark j given the entire robot pose sequence
  - → Instead of joint Gaussian over poses of all landmarks, can just keep track of Gaussian for each landmark separately
    - → Linear scaling with number of landmarks (rather than quadratic)

SLAM thus far

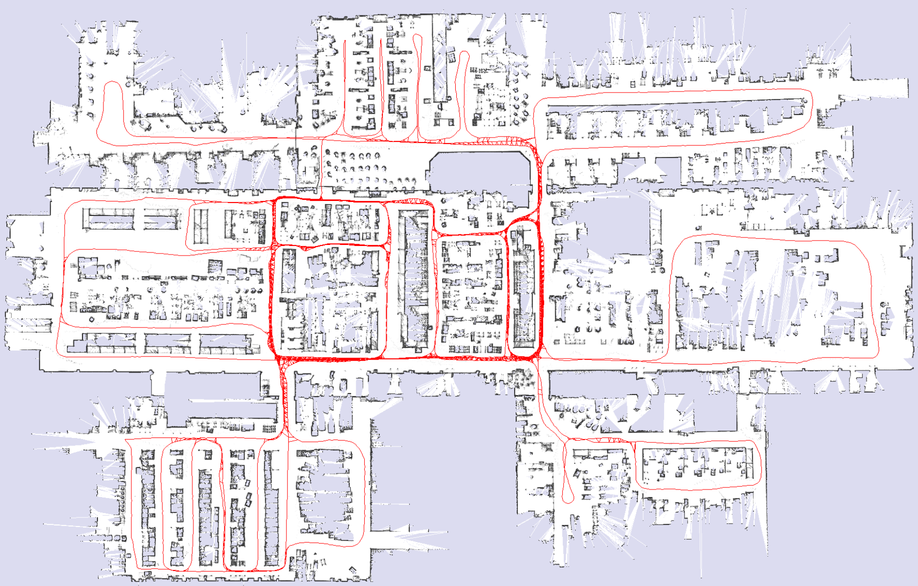
- Landmark based vs. occupancy grid
- Probability distribution representation:
  - EKF vs. particle filter vs. Rao-Blackwellized particle filter
- EKF, SEIF, FastSLAM are all "online"
- Currently popular 4th alternative: GraphSLAM

Graph-based Formulation

- Use a **graph** to represent the problem
- **Every node** in the graph **corresponds to a pose** of the robot during mapping
- **Every edge** between two nodes **corresponds to the spatial constraints** between them
- **Goal**:
  Find a configuration of the nodes that **minimize the error** introduced by the constraints

$$\begin{aligned} J_{\text{GraphSLAM}} &= x_0^\top \Omega_0 x_0 + \sum_t (x_t - f(u_t, x_{t-1}))^\top R_t^{-1} (x_t - f(u_t, x_{t-1})) \\ &+ \sum_t \sum_i (z_t^i - h(x_t, m, c_t^i))^\top Q_t^{-1} (z_t^i - h(x_t, m, c_t^i)) \end{aligned}$$

The KUKA Production Site



The KUKA Production Site

scans	59668
total acquisition time	4,699.71 seconds
traveled distance	2,587.71 meters
total rotations	262.07 radians
size	180 x 110 meters
processing time	< 30 minutes