Using the Kalman Filter for SLAM

AIMS 2015
Contents

• Trivial Kinematics
• Rapid sweep over localisation and mapping (components of SLAM)
• Basic EKF Feature Based SLAM
• Feature types and representations
• Implementation
• Convergence Properties
• Pose Based SLAM
• Sparse Inverse Pose Based Formulation
Vehicle Models - Prediction

Truth model

\[
x_v(k + 1) = f(x_v(k), u(k)); \quad u(k) = \begin{bmatrix} V(k) \\ \phi(k) \end{bmatrix}
\]

\[
\begin{bmatrix}
x_v(k + 1) \\ y_v(k + 1) \\ \theta_v(k + 1)
\end{bmatrix} = \begin{bmatrix}
x_v(k) + \delta TV(k) \cos(\theta_v(k)) \\ y_v(k) + \delta TV(k) \sin(\theta_v(k)) \\ \theta_v(k) + \frac{\delta TV(k) \tan(\phi(k))}{L}
\end{bmatrix}
\]
Noise is in control.

\[ u(k) = u_n(k) + v(k) \]

where \( u_n(k) \) is a nominal (intended) control signal and \( v(k) \) is a zero mean gaussian distributed noise vector:

\[ v(k) \sim \mathcal{N}(0, \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}) \]

\[ u(k) \sim \mathcal{N}(u_n(k), \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}) \]
Effect of control noise on uncertainty:

\[
\begin{align*}
x_v(k|k-1) &= f(\dot{x}(k-1|k-1), u(k), k) \\
P_v(k|k-1) &= \nabla F_x P_v(k-1|k-1) \nabla F_x^T + \nabla F_v Q \nabla F_v^T \\
Q &= \begin{bmatrix}
\sigma_v^2 & 0 \\
0 & \sigma_\phi^2
\end{bmatrix}
\end{align*}
\]

\[
\nabla F_x = \begin{bmatrix}
1 & 0 & -\delta T V \sin(\theta_v) \\
0 & 1 & \delta T V \cos(\theta_v) \\
0 & 0 & 1
\end{bmatrix}
\]

\[
\nabla F_u = \begin{bmatrix}
\delta T \cos(\theta_v) & 0 \\
\delta T \sin(\theta_v) & 0 \\
tan(\phi) & \delta T V \sec^2(\phi)
\end{bmatrix}
\]
Using Dead-Reckoned Data
Navigation Architecture

- **Hidden Control**
  - **Physics**
  - **Encoders Counts**
  - **Supplied Onboard System**
  - **Dead Reckoned output (drifts badly)**

- **Physics (to be estimated)**
  - **Required (corrected) navigation estimates**

- **DR Model**
  - **Integration**

- **Nav**
  - **Other Measurements**

- **Navigation (to be installed by us)**
Background T-Composition

\[ x_{i,k} = x_{i,j} \oplus x_{j,k} \]
\[ x_{j,i} = \ominus x_{i,j} \]

Compounding transformations
Just functions!

\[
\begin{align*}
\mathbf{x}_1 \oplus \mathbf{x}_2 &= \begin{bmatrix}
x_1 + x_2 \cos \theta_1 - y_2 \sin \theta_1 \\
y_1 + x_2 \sin \theta_1 + y_2 \cos \theta_1 
\end{bmatrix} \\
\ominus \mathbf{x}_1 &= \begin{bmatrix}
-x_1 \cos \theta_1 - y_1 \sin \theta_1 \\
x_1 \sin \theta_1 - y_1 \cos \theta_1 \\
-\theta_1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\mathbf{J}_1(\mathbf{x}_1, \mathbf{x}_2) &\triangleq \frac{\partial(\mathbf{x}_1 \oplus \mathbf{x}_2)}{\partial \mathbf{x}_1} \\
&= \begin{bmatrix}
1 & 0 & -x_2 \sin \theta_1 - y_2 \cos \theta_1 \\
0 & 1 & x_2 \cos \theta_1 - y_2 \sin \theta_1 \\
0 & 0 & 1
\end{bmatrix} \\
\mathbf{J}_2(\mathbf{x}_1, \mathbf{x}_2) &\triangleq \frac{\partial(\mathbf{x}_1 \oplus \mathbf{x}_2)}{\partial \mathbf{x}_2} \\
&= \begin{bmatrix}
\cos \theta_1 & -\sin \theta_1 & 0 \\
\sin \theta_1 & \cos \theta_1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\end{align*}
\]
Deduce an Incremental Move

These can be in massive error

\[ u(k) = \ominus x_o(k - 1) \oplus x_o(k) \]

But the common error is subtracted out here
Use this “move” as a control

\[ x_v(k + 1) = f(x_v(k), u(k)) \]
\[ = x_v(k) \oplus (\oplus x_o(k - 1) \oplus x_o(k)) \]
\[ = x_v(k) \oplus u_o(k) \]

Substitution into Prediction equation (using J1 and J2 as Jacobians):

\[ P(k|k - 1) = \nabla F_x P_v(k - 1|k - 1) \nabla F_x^T + \nabla F_v Q \nabla F_v^T \]
\[ = J_1(x_v, u_o) P_v(k - 1|k - 1) J_1(x_v, u_o)^T + J_2(x_v, u_o) U_o J_2(x_v, u_o)^T \]

Diagonal covariance matrix (3x3) of error in \( u_o \)
Feature Based Mapping and Navigation

Look at the code!!
Problem Space

- Features and Landmarks
- Vehicle-Feature Relative Observation
- Mobile Vehicle
- Global Reference Frame

Data Association
- State estimation
- Feature modeling

$p_i$

$x_v$
Landmarks / Features

Things that standout to a sensor:
Corners, windows, walls, bright patches, texture...

Map

\[ M = \begin{bmatrix} x_{f,1} \\ x_{f,2} \\ \vdots \\ x_{f,n} \end{bmatrix} \]

Point Feature called “i”

\[ x_{f,i} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \]

• We shall initially proceed by considering only point features
• We’ll discuss other feature types later
Three Inference Problem

Input is measurements conditioned on map and vehicle

Data: \[ p(Z^k | M, x_v) \]

We want to use Bayes’ rule to “invert” this and get maps and vehicles given measurements.

Problem 1 – inferring location given a map (easy)
Problem 2 – inferring a map given a location (easy)
Problem 3 – SLAM – learning a map and locating simultaneously

We can use a KF for all the above
Simple Motion Model

Plant Model

\[
\dot{x}(k|k-1) = \dot{x}(k-1|k-1) \oplus (\otimes x_o(k-1) \oplus x_o(k)) \\
\dot{x}(k|k-1) = \dot{x}(k-1|k-1) \oplus u_o(k) \\
P(k|k-1) = \nabla F_x P(k-1|k-1) \nabla F_x^T + \nabla F_v Q \nabla F_v^T \\
= J_1(x_v, u_o) P_v(k-1|k-1) J_1(x_v, u_o)^T + J_2(x_v, u_o) U_o J_2(x_v, u_o)^T
\]

Remember:
u is control, J’s are a fancy way of writing jacobians (composition operator). U is strength of noise in plant model.
Observation Model

\[
\mathbf{z}(k) \triangleq \begin{bmatrix} r \\ \theta \end{bmatrix} \\
= \mathbf{h}(\mathbf{x}(k), \mathbf{w}(k)) \\
= \begin{bmatrix} \sqrt{(x_i - x_v(k))^2 + (y_i - y_v(k))^2} \\ \text{atan2} \left( \frac{y_i - y_v(k)}{x_i - x_v(k)} \right) - \theta_v \end{bmatrix}
\]

We differentiate w.r.t \( x_v \) to arrive at the observation model jacobian:

\[
\nabla \mathbf{H}_x \triangleq \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \\
= \begin{bmatrix} \frac{x_i - x_v(k)}{r} & \frac{y_i - y_v(k)}{r} \\ \frac{y_i - y_v(k)}{r^2} & -\frac{x_i - x_v(k)}{r^2} & 0 \end{bmatrix}
\]
Example

No features seen here

Example Code Provided
Location Covariance
Location Innovation
Problem II - Mapping

Key Point: State Vector GROWS!

\[ x(k|k)^* = y(x(k|k), z(k), x_v(k|k)) \]
\[ = \begin{bmatrix} x(k|k) \\ g(x(k), z(k), x_v(k|k)) \end{bmatrix} \]
\[ = \begin{bmatrix} x(k|k) \\ x_v + r \cos(\theta + \theta_v) \\ y_v + r \sin(\theta + \theta_v) \end{bmatrix} \]

"State augmentation"
where the coordinates of the new feature are given by the function \( g \):

\[ x_{new} = g(x(k), z(k), x_v(k|k)) \]
\[ = \begin{bmatrix} x_v + r \cos(\theta + \theta_v) \\ y_v + r \sin(\theta + \theta_v) \end{bmatrix} \]

The state vector is the map
How is P augmented?

Simple! Use the transformation of covariance rule.

\[ b = f(a) \]

\[ \mathcal{E}\{\tilde{b}\tilde{b}^T\} = \nabla Y \mathcal{E}\{\tilde{\alpha}\tilde{\alpha}^T\} \nabla Y^T \]

\[ x(k|k)^* = y(x(k|k), z(k), x_v(k|k)) \]

\[ = \begin{bmatrix} x(k|k) \\ g(x(k), z(k), x_v(k|k)) \end{bmatrix} \]

\[ P(k|k)^* = \nabla Y_{x,z} \begin{bmatrix} P(k|k) & 0 \\ 0 & R \end{bmatrix} \nabla Y_{x,z}^T \]

\[ \nabla Y_{x,z} = \begin{bmatrix} I_{n \times n} & 0_{n \times 2} \\ \nabla G_x & \nabla G_z \end{bmatrix} \]

G is the feature initialisation function
Leading to:

\[
\mathbf{x}(k|k) = \begin{bmatrix}
\mathbf{x}(k|k) \\
x_v + r \cos(\theta + \theta_v) \\
y_v + r \sin(\theta + \theta_v)
\end{bmatrix}
\]

Angle from Veh to feature

Vehicle orientation

\[
\mathbf{P}(k|k) = \begin{bmatrix}
\mathbf{P}(k|k) & 0 \\
0 & \nabla G
\end{bmatrix}
\]

\[
\nabla G_z R \nabla G_z^T
\]
So what are models \( h \) and \( f \)?

\( h \) is a function of the feature being observed:

\[
\nabla H_x = \begin{bmatrix}
\cdots & 0 & \cdots \\
\underbrace{\text{other features}} & \nabla H_{x_{f_i}} & \underbrace{\text{other features}}
\end{bmatrix}
\]

\( f \) is simply the identity transformation:

\[
\begin{align*}
x(k + 1|k)_{map} &= x(k|k)_{map} \\
\nabla F_x &= I_{n,n}
\end{align*}
\]
Turn the handle on the EKF:

All hail the Oracle! How do we know what feature we are observing?
Problem III SLAM

“Build a map and use it at the same time”

“This a cornerstone of autonomy”
Basic S.S.C SLAM

A union of Localisation and Mapping

\[ x \triangleq \begin{bmatrix} x_v \\ x_{f,1} \\ \vdots \\ x_{f,n} \end{bmatrix} \]

Vehicle parameterisation e.g. x,y,theta

Feature parameterisations e.g. x,y for points

State vector has vehicle AND map

Why naïve? Computation!
Prediction:

\[
\begin{bmatrix}
  x_v(k+1) \\
  x_{f,1}(k) \\
  \vdots \\
  x_{f,n}(k)
\end{bmatrix} = \begin{bmatrix}
  x_v(k) \oplus u(k) \\
  x_{f,1}(k) \\
  \vdots \\
  x_{f,n}(k)
\end{bmatrix}
\]

Vehicle Moves

Map remains unchanged

\[
\nabla F_x = \begin{bmatrix}
  \nabla F_{x_v} & 0 \\
  0 & I_{2n \times 2n}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  J_1(x_v, u) & 0 \\
  0 & I_{2n \times 2n}
\end{bmatrix}
\]

\[
\nabla F_u = \begin{bmatrix}
  J_2(x_v, u) & 0 \\
  0 & 0_{2n \times 2n}
\end{bmatrix}
\]

We have all the terms needed for prediction step

\[
x(k|k-1) = f(x(k|k), (\underbrace{u(k)}_{\text{control}} + \underbrace{v(k)}_{\text{noise}}))
\]

\[
P(k|k-1) = \nabla F_x P(k-1|k-1) \nabla F_x^T + \nabla F_v U \nabla F_v^T
\]
Feature Initialisation:

\[
x(k|k)^* = \begin{bmatrix} x(k|k) \\ x_v + r \cos(\theta + \theta_v) \\ y_v + r \sin(\theta + \theta_v) \end{bmatrix}
\]

These two lines are \( g() \)

\[
P(k|k)^* = \nabla Y_{x,z} \begin{bmatrix} P(k|k) & 0 \\ 0 & 0 \end{bmatrix} \nabla Y_{x,z}^T
\]

\[
\nabla Y_{x,z} = \begin{bmatrix} I_{n \times n} & 0_{n \times 2} \\ \nabla G_x & \nabla G_z \end{bmatrix}
\]

This a step only runs if a new feature is seen
Observation Model

\[ \nabla H_x = \begin{bmatrix} \nabla H_{x_v} & \ldots & 0 & \ldots & \nabla H_{x_{fi}} & \ldots & 0 & \ldots \end{bmatrix} \]

\[ x_{fi} = [x_i, y_i]^T \]

We have all the terms needed for update step
ASYNCHRONOUS DATA ACQUISITION

Data Collection
- raw sensor data

Data Storage

SYNCHRONOUS PROCESSING

START

State Projection

Perceptual Grouping

DATA ASSOCIATION

new data available

grouped observations

positive associations

unexplained observations

no new data

Map Update

Feature Manufature

Feature Management

Optional techniques
- Hough transform, RANSAC, least medians, maximum likely-hood

new and existing unexplained data is combined with a history of vehicle states to search for a stable initialisation of a new feature

stable initialisation

ambiguity

Batch Update

Delayed State Management

new feature

Addition and removal of past vehicle states

EXIT

Features can be fused with each other (equivalence assertion). Stagnant features can be removed (garbage collection). Compound features can be built.
Feature Types

• In theory we can use any geometric parameterisation
• In practice you need to be very careful about a choice of parameterisation.

If $\rho$ is large small errors in $\theta$ do crazy things!

Much smarter to use SPMap (Tardos) which represents geometry and uncertainty in frames centred on a feature. Also see Kai Arras’ s PhD as an excellent SPMap resource
The SPMap

Closed Loop of Transformations

\[ 0 = \mathbf{B}_{f,e}[\ominus x_{w,f} \oplus x_{w,r} \oplus x_{r,s} \oplus x_{s,e}] \]

This is the observation model for every feature type!
Symmetry Selection

Measurement to Feature Transformation

\[ 0 = B_{f,e}[\ominus x_{w,f} \oplus x_{w,r} \oplus x_{r,s} \oplus x_{s,e}] \]

\[ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{f,e} \]

Only the \( \Delta x \) and \( \Delta \theta \) components of \( x_{f,e} \) matter.
Calculating EKF Jacobians

Differentiating with respect to feature location

\[
\frac{\partial \Psi(\cdot)}{\partial \mathbf{p}_f} = \mathbf{B}_{r,e} \times \frac{\partial \Psi(\cdot)}{\partial [\mathbf{B}^T_{f,f} \mathbf{p}_f]} \times \frac{\partial [\mathbf{B}^T_{f,f} \mathbf{p}_f]}{\partial \mathbf{p}_f} \times \frac{\partial [\mathbf{B}^T_{f,f} \mathbf{p}_f]}{\partial \mathbf{p}_f}
\]

\[
= \mathbf{B}_{r,e} \times J_1 \{ \mathbf{B}^T_{f,f} \mathbf{p}_f, \mathbf{x}_{r,e} \} \times \mathbf{J}_1 \{ \mathbf{B}^T_{f,f} \mathbf{p}_f \} \times \mathbf{B}^T_{r,f}
\]

\[
= -\mathbf{B}_{r,e} \times J_1 \{0, \mathbf{x}_{r,e} \} \times \mathbf{B}^T_{r,f}
\]

similarly with respect to robot center

\[
\frac{\partial \Psi(\cdot)}{\partial \mathbf{p}_r} = \mathbf{B}_{r,e} \times \frac{\partial \Psi(\cdot)}{\partial [\mathbf{J}_{e,r} \mathbf{B}^T_{r,r} \mathbf{p}_r]} \times \frac{\partial [\mathbf{J}_{e,r} \mathbf{B}^T_{r,r} \mathbf{p}_r]}{\partial \mathbf{p}_r} \times \frac{\partial [\mathbf{B}^T_{r,r} \mathbf{p}_r]}{\partial \mathbf{p}_r}
\]

\[
= \mathbf{B}_{r,e} \times J_2 \{ \mathbf{x}_{r,e}, 0 \} \times \mathbf{J}_{e,r} \times \mathbf{B}^T_{r,r}
\]

and finally with respect to observation vector

\[
\frac{\partial \Psi(\cdot)}{\partial \mathbf{p}_e} = \mathbf{B}_{r,e} \times \frac{\partial \Psi(\cdot)}{\partial \mathbf{B}^T_{e,e} \mathbf{p}_e} \times \frac{\partial [\mathbf{B}^T_{e,e} \mathbf{p}_e]}{\partial \mathbf{p}_e} \times \frac{\partial [\mathbf{B}^T_{e,e} \mathbf{p}_e]}{\partial \mathbf{p}_e}
\]

\[
= \mathbf{B}_{r,e} \times J_2 \{ \mathbf{x}_{r,e}, 0 \} \times \mathbf{B}^T_{e,e}
\]

Common term \(X_{fe}\): how does Observation appear in feature frame?
SLAM in action

SPMap Formulation
Human Driven Exploration
Navigating
Autonomous Homing
Using the Map

Homing…

Get Insurance….

Homing… Final Adjustment
Atlas – Extending the EKF Using Multiple Frame

- Mapping MIT’s “Infinite Corridor”
- 2.2 km driven path

In collaboration with M. Bosse (PhD Thesis)  Prof. J. Leonard and Prof. S. Teller (MIT)
The Convergence and Stability of SLAM

By analysing the behaviour of the LG-KF we can learn about the governing properties of the SLAM problem – which are actually completely intuitive....
We can show that:

• The determinant of any submatrix of the map covariance matrix decreases monotonically as observations are successively made.

• In the limit as the number of observations increases, the landmark estimates become fully correlated.

• In the limit, the covariance associated with any single landmark location estimate is determined only by the initial covariance in the vehicle location estimate.
\[
\det \mathbf{P}(k + 1|k + 1) = \det \left[ \mathbf{P}(k + 1|k) - \mathbf{W}_i(k + 1) \mathbf{S}_i(k + 1) \mathbf{W}^T(k + 1) \right]
\leq \det \mathbf{P}(k + 1|k)
\]  
(1)

Any principal submatrix of a \textit{psd} matrix is also \textit{psd} Thus,

\[
\det \mathbf{P}_{mm}(k + 1|k + 1) \leq \det \mathbf{P}_{mm}(k + 1|k)
\]  
(2)

From the prediction equations (features don’t change)

\[
\mathbf{P}(k + 1|k) = 
\begin{bmatrix}
\mathbf{F}_v \mathbf{P}_{vv}(k|k) \mathbf{F}_v^T + \mathbf{Q}_{vv} & \mathbf{F}_v \mathbf{P}_{vm}(k|k) \\
\mathbf{P}_{vm}^T(k|k) \mathbf{F}_v^T & \mathbf{P}_{mm}(k|k)
\end{bmatrix}
\] .

\[
\det \mathbf{P}_{mm}(k + 1|k + 1) \leq \det \mathbf{P}_{mm}(k|k).
\]  
(3)

\[
\sigma_{ii}^2(k + 1|k + 1) \leq \sigma_{ii}^2(k|k).
\]

where \( \sigma_{ii}^2 \) is a diagnoal element of \( \mathbf{P}_{mm} \).
As the number of observations taken tends to infinity a lower limit on the map covariance limit will be reached such that

\[
\lim_{k \rightarrow \infty} [P_{mm}(k + 1|k + 1)] = P_{mm}(k|k)
\]  

(1)

Writing \( P(k + 1|k) \) as \( P^\oplus \) and \( P(k + 1|k + 1) \) as \( P^\ominus \) for notational clarity the update for the map can be written as

\[
P_{mm}(k + 1|k + 1) = P_{mm}(k + 1|k) - M_2 S_i^{-1} M_2^T
\]

\[
= P_{mm}(k|k) - M_2 S_i^{-1} M_2^T
\]  

(2)

where

\[
M_2 = -P^\ominus_{vm} H_v^T + P^\oplus_{mm} H_{pi}^T
\]  

(3)
In the limit then $M_2 S_i^{-1} M_2^T = 0$ which implies that $M_2 = 0$

$$P_{mm}(k|k)H_{pi}^T = P_{vm}(k|k)H_v^T \quad \forall i$$  

(1)

this holds for all $i$ and therefore the block columns of $P_{mm}$ are linearly dependent and therefore

$$\lim_{k \to \infty} [\det P_{mm}(k|k)] = 0$$  

(2)

Consider now the relative position between any two landmarks $p_i$ and $p_j$ of the same type.

$$d = \hat{p}_i - \hat{p}_j$$

$$= G_{ij} x$$

The covariance $P_d$ of $d$ is given by

$$P_d = G_{ij} P G_{ij}^T$$

$$= P_{ii} + P_{jj} - P_{ij} - P_{ij}^T$$

With similar landmarks types, $H_{pi} = H_{pj}$ and so Equation (2) implies that the block columns of $P_{mm}$ are identical. Furthermore, because $P_{mm}$ is symmetric it follows that

$$P_{ii} = P_{jj} = P_{ij} = P_{ij}^T \quad \forall i, j$$  

(3)

Therefore, in the limit, $P_d = 0$ and the relationship between the landmarks is known with complete certainty.
To be Tattooed on Inside of Eyelids

• The entire structure of the SLAM problem critically depends on maintaining complete knowledge of the cross correlation between landmark estimates. Minimizing or ignoring cross correlations is precisely contrary to the structure of the problem.

• As the vehicle progresses through the environment the errors in the estimates of any pair of landmarks become more and more correlated, and indeed never become less correlated.

• In the limit, the errors in the estimates of any pair of landmarks becomes fully correlated. This means that given the exact location of any one landmark, the location of any other landmark in the map can also be determined with absolute certainty.

• As the vehicle moves through the environment taking observations of individual landmarks, the error in the estimates of the relative location between different landmarks reduces monotonically to the point where the map of relative locations is known with absolute precision.

• As the map converges in the above manner, the error in the absolute location of every landmark (and thus the whole map) reaches a lower bound determined only by the error that existed when the first observation was made. (We didn’t prove this here. However it is an excellent test for consistency in new SLAM algorithms)

This is all under the assumption that we observe all features equally often…

for other cases see Kim Sun-Joon PhD MIT 2004
Pose Based SLAM

Scan matching between past poses produces observation with which to update state-vector

- State vector contains only past vehicle poses.
- Each pose has a scan rigidly attached to it

\[ x_v(k + 1|k) = x_v(k|k) \oplus u(k + 1) \]

\[ x(k + 1|k) = \begin{bmatrix} x(k|k) \\ x_v(k|k) \oplus u(k + 1) \\ x_v(0|0) \\
\vdots \\ x_v(k|k) \\ x_v(k + 1|k) \end{bmatrix} \]

Control

Past poses

Growth with time
Pose Based Prediction Step

\[ \mathbf{x}(k+1|k) = \begin{bmatrix} x_v[0] \\ \vdots \\ x_v[k] \\ x_v[k+1] \end{bmatrix} = \begin{bmatrix} \mathbf{x}(k|k) \\ x_v[k] \oplus \mathbf{u}(k+1) \end{bmatrix} \]

Past poses

Writing this as a transformation allows us to calculate the augmented state covariance

\[ \mathbf{x}(k+1|k) = f(\mathbf{x}(k|k), \mathbf{u}(k+1)) \]

Note \( f() \) changes the size of the state vector!

\[ \nabla \mathbf{F}_x = \begin{bmatrix} \mathbf{0}_{3 \times n-3} & \mathbf{I}_{n \times n} \\ \mathbf{J}_1(x_v[k], \mathbf{u}(k+1)) \end{bmatrix} \]

\[ \nabla \mathbf{F}_u = \begin{bmatrix} \mathbf{0}_{3 \times n-3} & \mathbf{J}_2(x_v[k], \mathbf{u}(k+1)) \end{bmatrix} \]

and so

\[ \mathbf{P}(k+1|k) = \nabla \mathbf{F}_x \mathbf{P}(k|k) \nabla \mathbf{F}_x^T + \nabla \mathbf{F}_u \mathbf{U} \nabla \mathbf{F}_u^T \]

Plant model \( f() \) and Jacobian are all we need to plug into EKF prediction equations
Pose Based Update Step

\[ z_{i,j}(k) = \Theta x_v[i] \oplus x_v[j] + w(k) \]
\[ = h(x(k)) + w(k) \]

Observation model (h) and Jacobian are all we need to plug into EKF update equations

\[ \nabla H_x = \begin{bmatrix} 0 \ldots J_1(\Theta x_v[i], x_v[j]) J_m(x_v[i]) \ldots 0 \ldots J_2(\Theta x_v[i], x_v[j]) \ldots 0 \ldots \end{bmatrix} \]

RUN LIVE DEMO HERE
Demonstration 3D Pose Based SLAM
Issues with Single Frame SLAM

- It is uni-modal. It cannot cope with ambiguous situations
- It is inconsistent - the linearisations lead to errors which underestimate the covariance of the underlying pdf
- It is fragile - if the estimated is in error the linearisation is very poor – disaster.

- But the biggest problem is (was) scaling

Much progress has been made on scaling – see upcoming talks. In particular
- FastSLAM Montemerlo et. al
- Atlas(Bosse et al)
- Postponement (Davison, Nebot)
- CTS (Leonard Newman)
- Exactly Sparse Delayed State Filter (Eustice et. Al (see third practical) )
And Finally: turn it all on its Head – the information form

\[ p(x) = \mathcal{N}(x; \hat{x}, P) \]
\[ \alpha \exp\left\{ -\frac{1}{2}(x - \hat{x})^T P^{-1}(x - \hat{x}) \right\} \]
\[ \alpha \exp\left\{ -\frac{1}{2}x^T P^{-1}x + \hat{x}^T P^{-1}x \right\} \]
\[ = \exp\left\{ -\frac{1}{2}x^T \Lambda x + \eta^T x \right\} \]

where

\[ \Lambda = P^{-1} \]
\[ \eta = \Lambda \hat{x} \]

We can propagate information matrix an information vector instead of state and covariance.

Information Matrix is inverse of covariance (makes sense!)
Look Again at Augmentation

\[ \hat{x} = \begin{bmatrix} x_m \\ x_v[0 : k - 1] \\ \vdots \\ x_v[k] \\ f(x_v[k], u) \end{bmatrix}, \quad P = \begin{bmatrix} P_{m,m} & P_{m,v} & P_{m,v} \nabla F_x P_{m,v} \\ P_{v,m}^T & P_{v,v} & \nabla F_x P_{v,v} \\ \nabla F_x P_{m,v} & \nabla F_x P_{v,v} & \nabla F_x P_{v,v} + \nabla F_u U \nabla F_u \end{bmatrix} \]

\[ \Lambda = P^{-1}, \quad \eta = \Lambda \hat{x} \]

\[ \eta = \begin{bmatrix} \eta_{m} \\ \eta_{v} - \nabla F_x^T Q^{-1} (f(x_v[k], u) - \nabla F_x x_v[k]) \\ Q^{-1} (f(x_v[k], u) - \nabla F_x x_v[k]) \end{bmatrix} \]

\[ \Lambda = \begin{bmatrix} \Lambda_{m,m} & \Lambda_{m,v} & 0 \\ \Lambda_{v,m} & \Lambda_{v,v} + \nabla F_x^T Q^{-1} \nabla F_x & -\nabla F_x^T Q^{-1} \\ 0 & -Q^{-1} \nabla F_x & Q^{-1} \end{bmatrix} \]

Matrix inversion lemma

Note the zeros in the information matrix that don’t appear in the EKF version.....
The information matrix remains band-diagonal when augmenting! → constant time. (compare with EKF which is linear with state size)

• Why? Because of the markov property of the plant model – we only need the previous state to predict a new pose.
• There is much that can be said about the structure of the information matrix and the underlying pd.f but time is short....
And there’s more...

Information update equations are:

\[ \eta^+ = \eta^- + \nabla H_x^T R^{-1} (z - h(x) + \nabla H_x x) \]
\[ \Lambda^+ = \Lambda^- + \nabla H_x^T R^{-1} \nabla H_x \]

where

\[ \nabla H_x = \begin{bmatrix} 0 & \cdots & \nabla H_{xi} & \cdots & 0 & \cdots & \nabla H_{xj} & \cdots & 0 & \cdots \end{bmatrix} \]

so Information update is additive and very very sparse:

\[ \Lambda^- + \nabla H_x^T R^{-1} \nabla H_x = \Lambda^+ \]

• It appears then that the update is constant time…..
But There’s No Free Lunch

The augmentation and update equations need knowledge of $x$ which is also needed to evaluate $\nabla H_x$ and $\nabla F_x$. Naively we could solve as

$$x = \Lambda^{-1} \eta$$

but that would be stupid:

- Cubic cost!
- We don’t want all of $\Lambda^{-1} = P$ at run time

Better plan is to solve for $x$ using Cholesky decomposition as $\Lambda$ is P.S.D.

- note that because $\nabla H_x$ and $\nabla F_x$ are so sparse we only ever need part of the state vector.
- this allows us to effectively side step the cost of solving for all $x$ if we keep a broadly correct $x$ around (problem when loop closing though)

More details in Eustice et al’ s ICRA 2005 paper
Relates strongly to contemporary graphical-SLAM methods (see upcoming talks)
What remains under the carpet?

- Data Association (what am I looking at)
- Loop Closing (was I really that lost?)
- Consistency and Robustness
- Scaling
- Relationship to large scale optimisation

These are meaty topics and will be the focus of other summer school lectures…

Many thanks for your time