Consistency and SLAM

Simon J. Julier

Department of Computer Science
University College London
Malet Place
London WC1E 6BT
S.Julier@cs.ucl.ac.uk
+44 (0)20 7679 4132

August 30, 2006

Structure

- Definition of the Consistency Problem
- Modelling Errors and Consistency
  - ⋆ Effects
  - ⋆ Mitigating Strategies
- Linearisation Errors and Consistency
  - ⋆ Effects
  - ⋆ Mitigating Strategies
- Conclusions

Caveat

- This talk only looks at recursive estimators and, in particular, filters which use a mean and covariance representation
- It is not clear if these results generalise to other formulations of the problem
  - ⋆ Non-iterative schemes (e.g., GraphSLAM)
  - ⋆ Non-mean and covariance schemes (e.g., FastSLAM)

SLAM

- In state space form, the vehicle and beacons are put into the same state vector known as a *stochastic map*
- With a map of N beacons, the structure of the estimate is

$$\mathbf{x}_N(k) = \begin{bmatrix} \mathbf{x}_v(k) \\ \mathbf{p}_1(k) \\ \vdots \\ \mathbf{p}_N(k) \end{bmatrix}$$

- Throughout this talk, the suffix to denote the number of beacons will be used inconsistently (sorry...)

Structure of a SLAM Filter

- The state space for a SLAM Kalman filter stores all the vehicle and all the beacon states in a single state space:

$$\hat{\mathbf{x}}(k \mid k) = \left[\hat{\mathbf{x}}_v^T(k \mid k) \dots \hat{\mathbf{p}}_N^T(k \mid k)\right]^T$$

$$\mathbf{P}(k \mid k) = \begin{pmatrix} \mathbf{P}_{vv}(k \mid k) & \mathbf{P}_{v1}(k \mid k) & \dots & \mathbf{P}_{vN}(k \mid k) \\ \mathbf{P}_{1v}(k \mid k) & \mathbf{P}_{11}(k \mid k) & \dots & \mathbf{P}_{1N}(k \mid k) \\ \mathbf{P}_{2v}(k \mid k) & \mathbf{P}_{21}(k \mid k) & \dots & \mathbf{P}_{2N}(k \mid k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{Nv}(k \mid k) & \mathbf{P}_{N1}(k \mid k) & \dots & \mathbf{P}_{NN}(k \mid k) \end{pmatrix}$$

Definition of Consistency

- Consistency is the criteria which shows that the filter actually *works*

$$\mathbf{P}\left(i \mid j\right) - \mathrm{E}\left[\tilde{\mathbf{x}}\left(i \mid j\right)\tilde{\mathbf{x}}\left(i \mid j\right)^T\right] \geq \mathbf{0}$$

- The *zero mean* condition is often stipulated

$$\mathrm{E}\left[\tilde{\mathbf{x}}\left(i \mid j\right)\right] = \mathbf{0}$$

- This is overly restrictive; non-zero errors cause the consistent value of $\mathbf{P}\left(i \mid j\right)$ to increase accordingly

Consistency in SLAM

- Let the true state be $\mathfrak{x}(k)$ and the state estimate be $\hat{\mathbf{x}}(k \mid k)$
- The covariance propagated by the filter is

$$\mathbf{P}(k \mid k) = \begin{pmatrix} \mathbf{P}_{vv}(k \mid k) & \mathbf{P}_{vp}(k \mid k) \\ \mathbf{P}_{pv}(k \mid k) & \mathbf{P}_{pp}(k \mid k) \end{pmatrix}$$

- However, the true mean squared error is

$$\mathrm{E}\left[\tilde{\mathbf{x}}(i \mid j)\,\tilde{\mathbf{x}}(i \mid j)^T\right] = \begin{pmatrix} \mathfrak{P}_{vv}(k|k) & \mathfrak{P}_{vp}(k|k) \\ \mathfrak{P}_{pv}(k|k) & \mathfrak{P}_{pp}(k|k) \end{pmatrix}$$

- If we don't know the true mean squared error, can we assume a "conservative" vehicle-beacon cross correlation?

There is no “Conservative” Cross Correlation

- The condition for consistency is

$$\begin{pmatrix} \mathbf{P}_{vv}\left(k \mid k\right) - \mathfrak{P}_{vv}\left(k|k\right) & \mathbf{P}_{vp}\left(k \mid k\right) - \mathfrak{P}_{vp}\left(k|k\right) \\ \mathbf{P}_{pv}\left(k \mid k\right) - \mathfrak{P}_{pv}\left(k|k\right) & \mathbf{P}_{pp}\left(k \mid k\right) - \mathfrak{P}_{pp}\left(k|k\right) \end{pmatrix} \geq \mathbf{0}$$

- Suppose that the block diagonals are correct but the cross correlations are incorrect

- The error matrix is

$$\begin{pmatrix} \mathbf{0} & \mathbf{P}_{vp}\left(k \mid k\right) - \mathfrak{P}_{vp}\left(k|k\right) \\ \mathbf{P}_{pv}\left(k \mid k\right) - \mathfrak{P}_{pv}\left(k|k\right) & \mathbf{0} \end{pmatrix} \not\geq \mathbf{0}$$

Consistency Requires Inflation of the Block Diagonal Terms

- Let
$$\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{P}_{vp}(k \mid k) - \mathfrak{P}_{vp}(k|k)$$

- It can be shown (not here!) that the matrix
$$\begin{pmatrix} \mathbf{P}_{vv}(k \mid k) + \frac{1}{\omega}\mathbf{U}|\mathbf{S}|\mathbf{V}^T & \mathbf{P}_{vp}(k \mid k) \\ \mathbf{P}_{pv}(k \mid k) & \mathbf{P}_{pp}(k \mid k) + \frac{1}{1-\omega}\mathbf{V}|\mathbf{S}|\mathbf{U}^T \end{pmatrix}$$
is consistent for all $\omega \in [0, 1]$

- Consistency can be attained by *inflating* the values of both the block diagonals of the covariance matrix

- However, there is a trade-off between the vehicle and beacon states

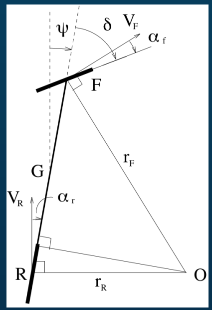
Consistency and SLAM: Summary

- Covariance consistency is an important measure for deterimining if a filter works
- The structure of SLAM means that if the cross correlations are known imprecisely, the covariances on the vehicle and beacon states must be inflated
- The uncertainty trades-off increases in vehicle covariance or beacon covariance
- Inflating beacon covariance terms undermines most analysis of the long term properties of SLAM

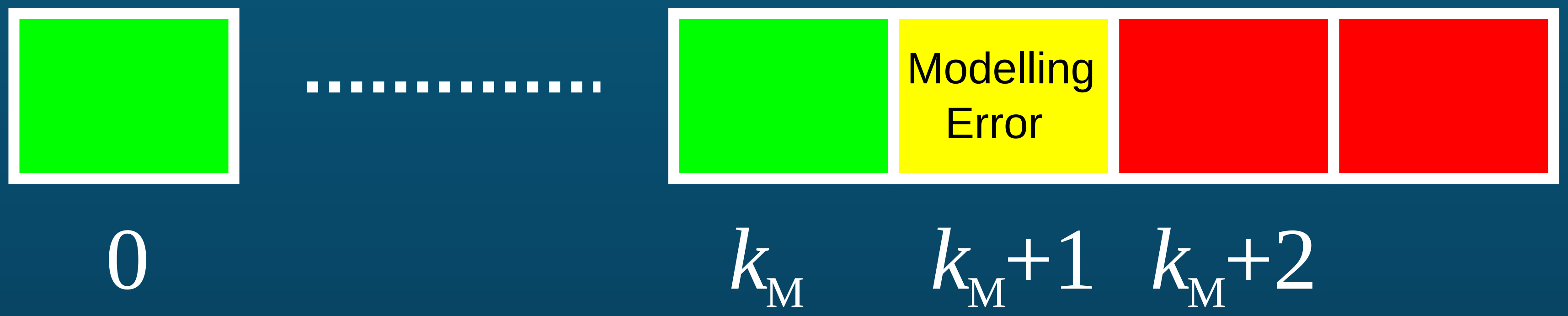
- Understanding the conditions which cause a filter to go inconsistent and preventing this from happening is *vital* for robust SLAM

Modelling Errors Are the Norm, Not the Exception

- *All* real systems have modelling errors:
  - ⋆ Physics of the underlying system too complicated to model in real-time
  - ⋆ Physics of the underlying system not known
- Normal solution is to choose *parsimonious models* and use “engineering judgement” to tune filters until they work
- This works fine in normal tracking examples — but what about SLAM?



Effects of *Process Model* Errors



- The filter and true model use the same process model up to time step k_M
- The true model uses a different process model from time step $k_M + 1$
- No finite value of $\mathbf{Q}_v\left(k_M + 1\right)$ exists to stop the filter becoming inconsistent at time step $k_M + 2$

Nominal and Real Process Models

- The true model is

$$\mathfrak{x}\left(k\right) = \mathfrak{F}\left(k\right)\mathfrak{x}\left(k-1\right) + \mathfrak{v}\left(k-1\right)$$

- The nominal model is

$$\mathbf{x}\left(k\right) = \mathbf{F}\left(k\right)\mathbf{x}\left(k-1\right) + \mathbf{v}\left(k-1\right)$$

Assumed Error Structure in the Prediction

- The filter assumes that the nominal model is always correct
- Therefore, the filter predicts using the equation

$$\hat{\mathbf{x}}\left(k_M+1 \mid k_M\right)=\mathbf{F}\left(k_M+1\right)\hat{\mathbf{x}}\left(k_M \mid k_M\right)$$

- The filter *assumes* the prediction error is

$$\tilde{\mathbf{x}}\left(k_M+1 \mid k_M\right)=\mathbf{F}\left(k_M+1\right)\tilde{\mathbf{x}}\left(k_M \mid k_M\right)-\mathbf{v}\left(k_M\right)$$

Predicted Covariance

- Taking outer products and expectations,

$$\mathbf{P}(k_M+1|k_M)=\begin{pmatrix} \mathbf{P}_{vv}(k_M+1|k_M) & \mathbf{P}_{v1}(k_M+1 \mid k_M) & \ldots & \mathbf{P}_{vN}(k_M+1 \mid k_M) \\ \mathbf{P}_{1v}(k_M+1 \mid k_M) & \mathbf{P}_{11}(k_M \mid k_M) & \vdots & \mathbf{P}_{1N}(k_M \mid k_M) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{Nv}(k_M+1 \mid k_M) & \mathbf{P}_{11}(k_M \mid k_M) & \vdots & \mathbf{P}_{NN}(k_M \mid k_M) \end{pmatrix}$$

where

$$\mathbf{P}_{vv}(k_M+1 \mid k_M) = \mathbf{F}_v(k_M+1)\,\mathbf{P}_{vv}(k_M+1 \mid k_M)\,\mathbf{F}_v^T(k_M+1) + \mathbf{Q}_v(k_M)$$

$$\mathbf{P}_{vi}(k_M+1 \mid k_M) = \mathbf{F}_v(k_M+1)\,\mathbf{P}_{vi}(k_M \mid k_M)$$

Actual Error Structure in the Prediction

- The real errors introduced at any given time step are

$$\begin{aligned}\tilde{\mathfrak{x}}(k_M+1|k_M) &= \hat{\mathbf{x}}(k_M+1 \mid k_M) - \mathfrak{x}(k_M+1)\\ &= \mathbf{F}(k_M+1)\,\hat{\mathbf{x}}(k_M \mid k_M) - \mathfrak{F}(k_M+1)\,\mathfrak{x}(k_M) - \mathfrak{v}(k_M)\\ &= \mathbf{F}(k_M+1)\,\tilde{\mathbf{x}}(k_M \mid k_M) + \tilde{\mathbf{F}}(k_M+1)\,\mathfrak{x}(k_M) - \mathfrak{v}(k_M)\end{aligned}$$

  where

$$\tilde{\mathbf{F}}(k_M+1) = \mathbf{F}(k_M+1) - \mathfrak{F}(k_M+1)$$

- Therefore, a component of the error is a function of the *value* of the true state itself, not an error term

- This value is time correlated, not generally zero mean and can have a large value

Actual Mean Squared Error

- Taking outer products and expectations,

$$\mathfrak{P}\left(k_M+1|k_M\right) = \begin{pmatrix} \mathfrak{P}_{vv}\left(k_M+1|k_M\right) & \mathfrak{P}_{v1}\left(k_M+1|k_M\right) & \ldots & \mathfrak{P}_{vN}\left(k_M+1|k_M\right) \\ \mathfrak{P}_{1v}\left(k_M+1|k_M\right) & \mathfrak{P}_{11}\left(k_M|k_M\right) & \vdots & \mathfrak{P}_{1N}\left(k_M|k_M\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{P}_{Nv}\left(k_M+1|k_M\right) & \mathfrak{P}_{11}\left(k_M|k_M\right) & \vdots & \mathfrak{P}_{NN}\left(k_M|k_M\right) \end{pmatrix}$$

- The expressions for $\mathfrak{P}_{vv}\left(k_M+1|k_M\right)$ and $\mathfrak{P}_{vi}\left(k_M+1|k_M\right)$ are a little more involved than in the no modelling error case...

Actual Mean Squared Error in the Vehicle Estimation

- The actual mean squared error in the vehicle estimate is

$$\begin{aligned}\mathfrak{P}_{vv}\left(k_M+1|k_M\right) = {} & \mathbf{F}_v\left(k_M+1\right)\mathbf{P}_{vv}\left(k_M \mid k_M\right)\mathbf{F}_v^T\left(k_M+1\right)\\ & + \mathbf{F}_v\left(k_M+1\right)\mathrm{E}\left[\tilde{\mathbf{x}}\left(k_M \mid k_M\right)\mathfrak{x}^T\left(k_M\right)\right]\tilde{\mathbf{F}}_v^T\left(k_M+1\right)\\ & + \tilde{\mathbf{F}}_v\left(k_M+1\right)\mathrm{E}\left[\mathfrak{x}\left(k_M\right)\tilde{\mathbf{x}}\left(k_M \mid k_M\right)^T\right]\mathbf{F}_v^T\left(k_M+1\right)\\ & + \tilde{\mathbf{F}}_v\left(k_M+1\right)\mathrm{E}\left[\mathfrak{x}\left(k_M\right)\mathfrak{x}^T\left(k_M\right)\right]\tilde{\mathbf{F}}_v^T\left(k_M+1\right)\\ & + \mathbf{Q}_{vv}\left(k_M\right)\end{aligned}$$

- In other words, this can be written as

$$\mathfrak{P}_{vv}\left(k_M+1|k_M\right) = \mathbf{P}_{vv}\left(k_M+1 \mid k_M\right) + \mathbf{\Delta P}_{vv}\left(k_M+1 \mid k_M\right)$$

Actual Vehicle-Beacon Cross Correlation

- Taking outer products,

$$\mathfrak{P}_{vi}(k_M+1|k_M) = \mathbf{F}_v(k_M+1)\,\mathbf{P}_{vi}(k_M+1 \mid k_M) + \tilde{\mathbf{F}}_v(k_M+1)\,\mathrm{E}\left[\mathfrak{x}(k_M+1)\,\tilde{\mathbf{p}}_i(k_M+1 \mid k_M)^T\right]$$

- This can be written as

$$\mathfrak{P}_{vi}(k_M+1|k_M) = \mathbf{P}_{vi}(k_M+1 \mid k_M) + \mathbf{\Delta P}_{vi}(k_M+1 \mid k_M)$$

- For $k \leq k_M + 1$, the nominal and truth models are the same and

$$\mathbf{\Delta P}_{vi}(k_M+1 \mid k_M) = \mathbf{0}$$

The Update

- The Kalman Filter updates the estimate according to

$$\hat{\mathbf{x}}\left(k_M+1 \mid k_M+1\right)=\mathbf{X}\left(k_M+1\right)\hat{\mathbf{x}}\left(k_M+1 \mid k_M\right)+\mathbf{W}\left(k_M+1\right)\mathbf{z}\left(k_M+1\right)$$

where

$$\mathbf{X}\left(k_M+1\right)=\mathbf{I}-\mathbf{W}\left(k_M+1\right)\mathbf{H}\left(k_M+1\right)$$

- The assumed error in the update is

$$\tilde{\mathbf{x}}\left(k_M+1 \mid k_M+1\right)=\mathbf{X}\left(k_M+1\right)\tilde{\mathbf{x}}\left(k_M+1 \mid k_M\right)+\mathbf{W}\left(k_M+1\right)\mathbf{w}\left(k_M+1\right)$$

The True Update

- Assuming that the observation model contains no errors, the true error expression is the same,

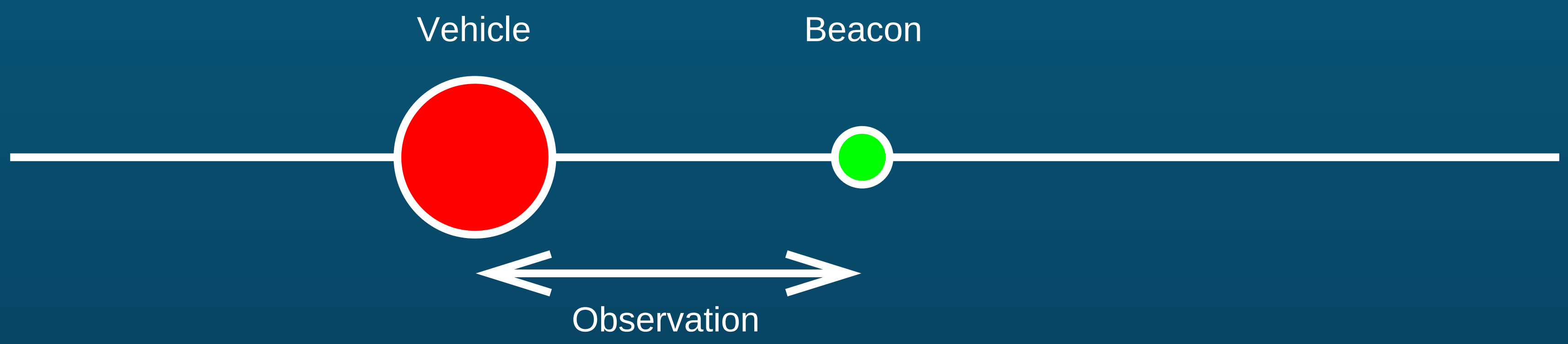
$$\tilde{\mathfrak{x}}\left(k_M+1|k_M+1\right)=\mathbf{X}\left(k_M+1\right)\tilde{\mathfrak{x}}\left(k_M+1|k_M\right)+\mathbf{W}\left(k_M+1\right)\mathfrak{w}\left(k_M+1\right)$$

- Therefore,

$$\tilde{\mathfrak{x}}\left(k_M+1|k_M+1\right)=\tilde{\mathbf{x}}\left(k_M+1 \mid k_M+1\right)+\mathbf{X}\left(k_M+1\right)\tilde{\mathbf{F}}\left(k_M+1\right)\mathfrak{x}\left(k_M\right)$$

- As a result, the updated estimate now includes the correlated error terms

Illustration



- Consider the following simple system:
  - ⋆ Vehicle is a 1D particle which moves on a line
  - ⋆ There is a single beacon
  - ⋆ The observation model measures displacement from vehicle to beacon
  - ⋆ Nominal filter and true system use same process model up to time k_1
  - ⋆ True system uses different process model to time k_2

Models

- The (continuous time) nominal model behaves according to two possible continuous time models,

$$\dot{\mathfrak{x}}\left(k\right) = \mathfrak{F}_c^{i_k}\left(k\right)\mathfrak{x}\left(k-1\right) + \mathfrak{v}\left(k-1\right)$$

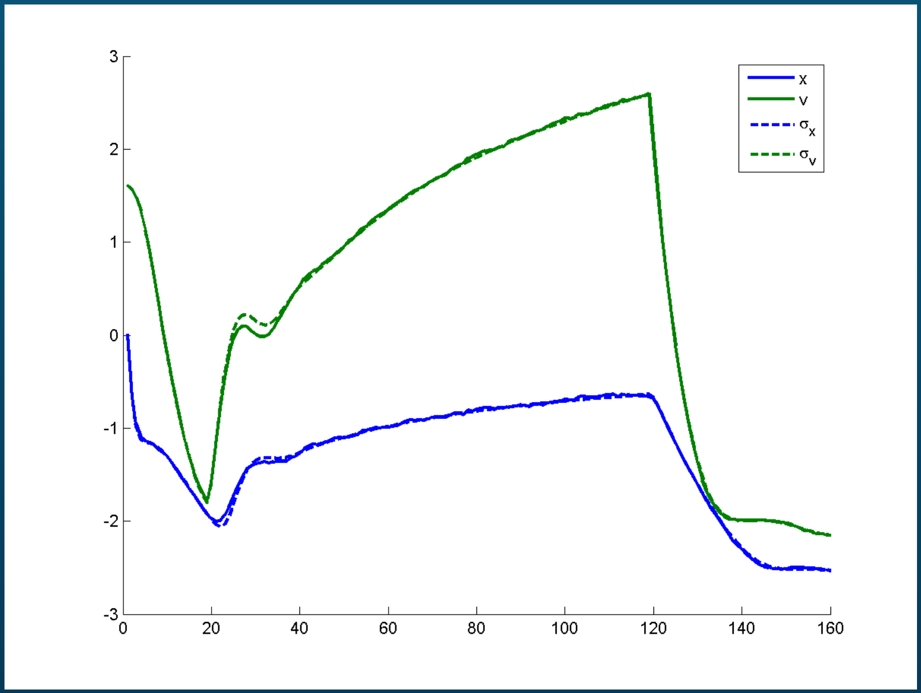
where

$$\mathfrak{F}_c^1\left(k\right) = \begin{pmatrix} 0 & 1 \\ -\omega & -\zeta \end{pmatrix}, \ \mathfrak{F}_c^2\left(k\right) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

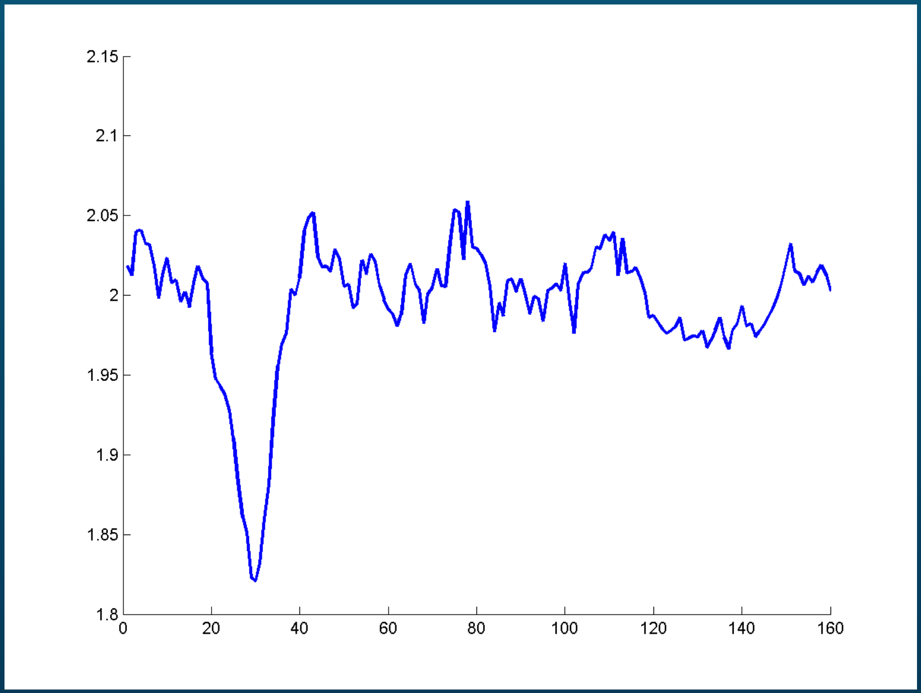
- The nominal filter assumes that the first mode acts for all time,
- The nominal process noise is chosen according to the iteration earlier to guarantee that $\mathbf{P}_{vv}\left(k \mid k-1\right)$ is consistent

![UCL logo]

Results (no SLAM)

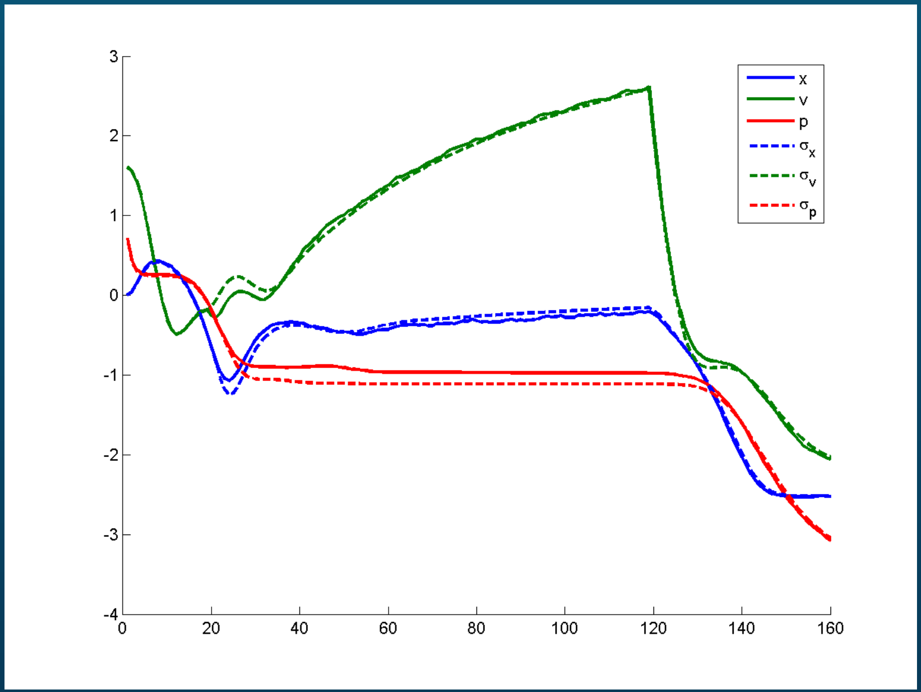


Actual mean squared error (solid) and estimated covariances (dashed) for x and v.

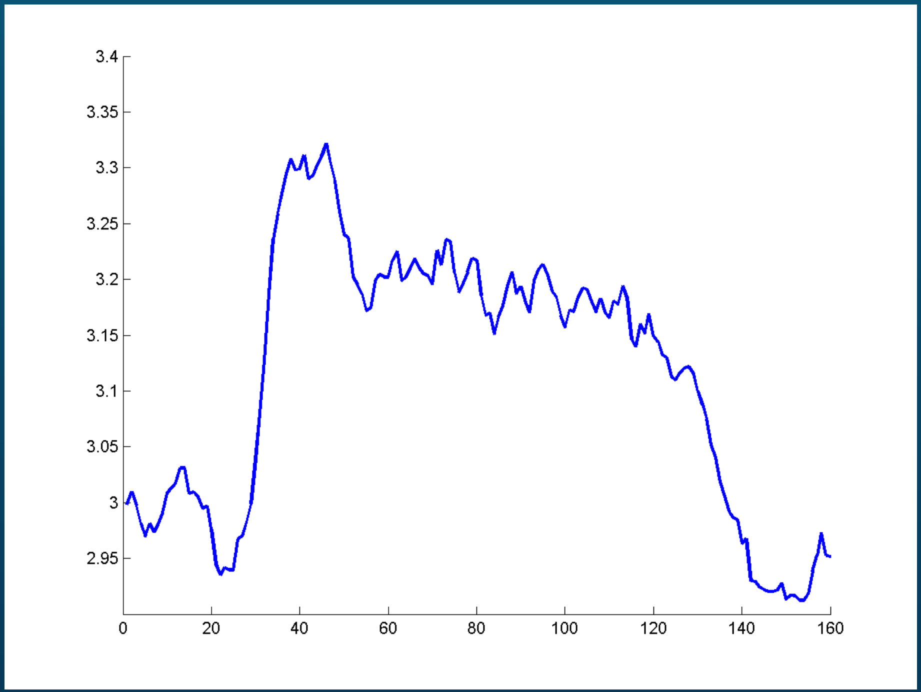


Normalised mean squared error.

Results (tuned process noise)



Actual mean squared error (solid) and estimated covariances (dashed) for x and v.



Normalised mean squared error.

Mitigating Strategies

- Mitigate errors in the process model
  - ⋆ Use more accurate models
  - ⋆ Refactor problem and use Inertial SLAM
- Tune the filter using stabilising noise
- Lock the map to prevent updates

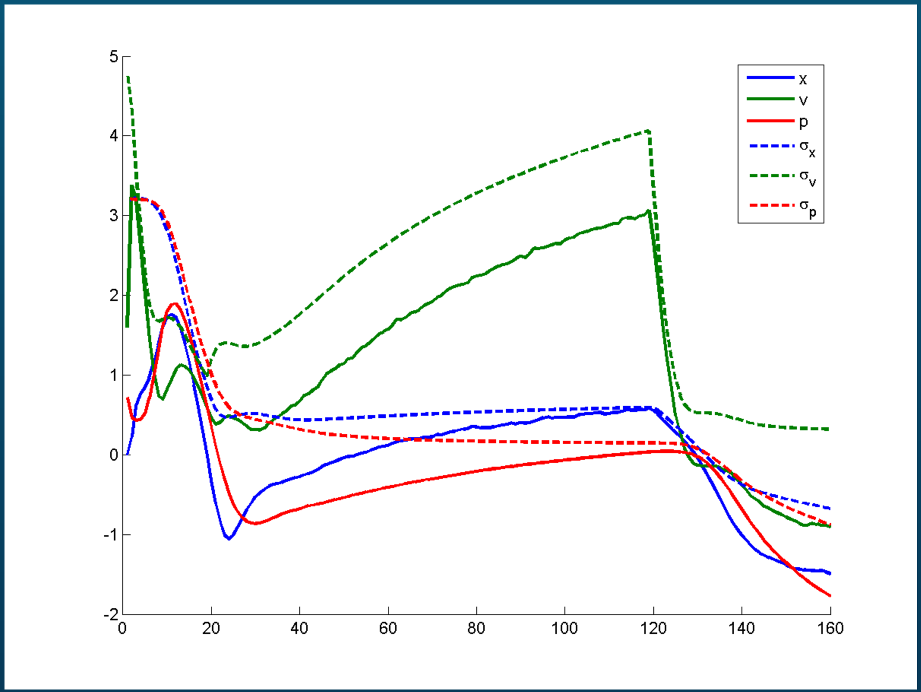
Stabilising Noise

- We earlier argued that consistency could be guaranteed if *both* $\mathbf{P}_{vv}\left(k \mid k\right)$ and $\mathbf{P}_{11}\left(k \mid k\right)$ are inflated

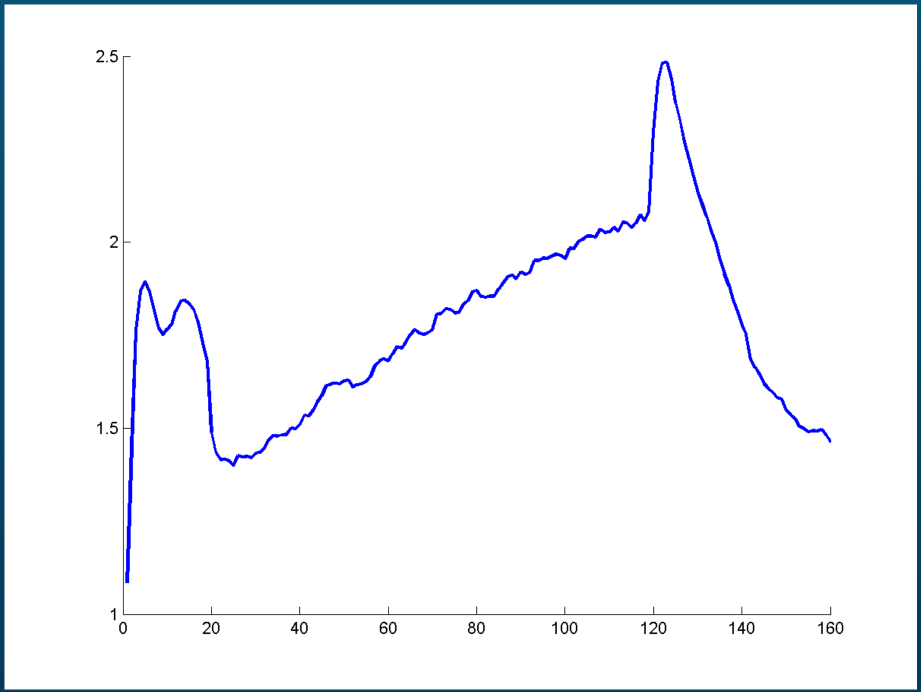
- One way to achieve this is to inflate both $\mathbf{Q}_{vv}\left(k\right)$ and $\mathbf{R}\left(k\right)$
  - ⋆ Inflating $\mathbf{Q}_{vv}\left(k\right)$ inflates $\mathbf{P}_{vv}\left(k \mid k\right)$ and also inflates $\mathbf{P}_{11}\left(k \mid k\right)$ because the covariance is "too large" when the beacon is first initialised
  - ⋆ Inflating $\mathbf{R}\left(k\right)$ reduces the amount of information actually available in the observation and causes the steady-state beacon covariane to increase

- Empirical tests suggest inflating $\mathbf{Q}_{vv}\left(k\right)$ by 24

Results (Stabilising Noise)



Actual mean squared error (solid) and estimated covariances (dashed) for x and v.



Normalised mean squared error.

Adding Process Noise to Beacons

- The diagonal terms could be increased by injecting process noise into the beacons,

$$\mathbf{P}_{11}\left(k \mid k-1\right) = \mathbf{P}_{11}\left(k-1 \mid k-1\right) + \mathbf{Q}_{11}\left(k\right)$$

- This looks like it could cause the map to wander off but if

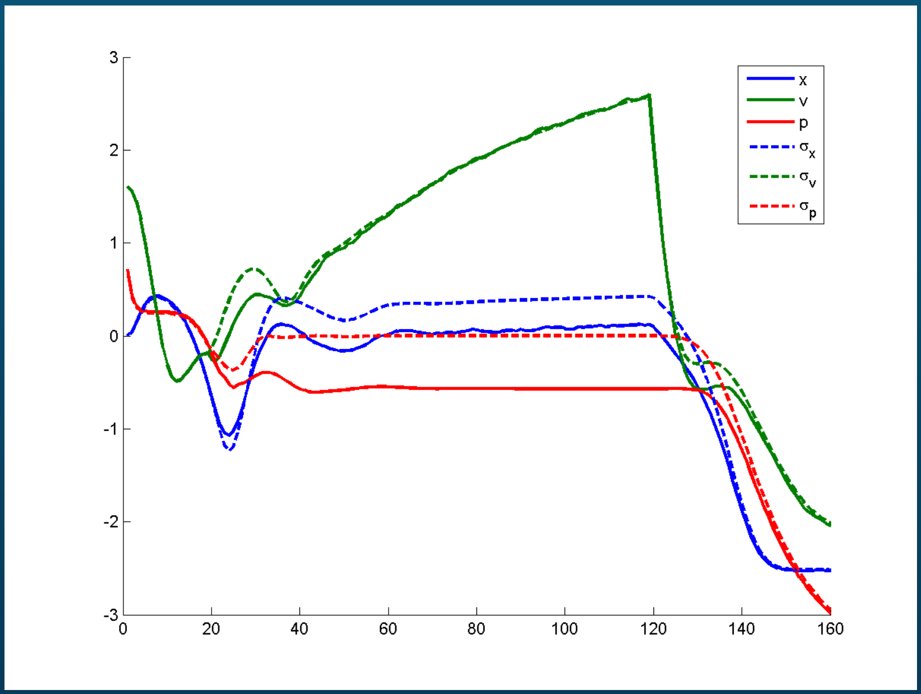
$$\mathbf{P}_{11}\left(k-1 \mid k-2\right) \geq \mathbf{P}_{11}\left(k \mid k-1\right)$$

the determinant in the submap for the beacon is nonincreasing

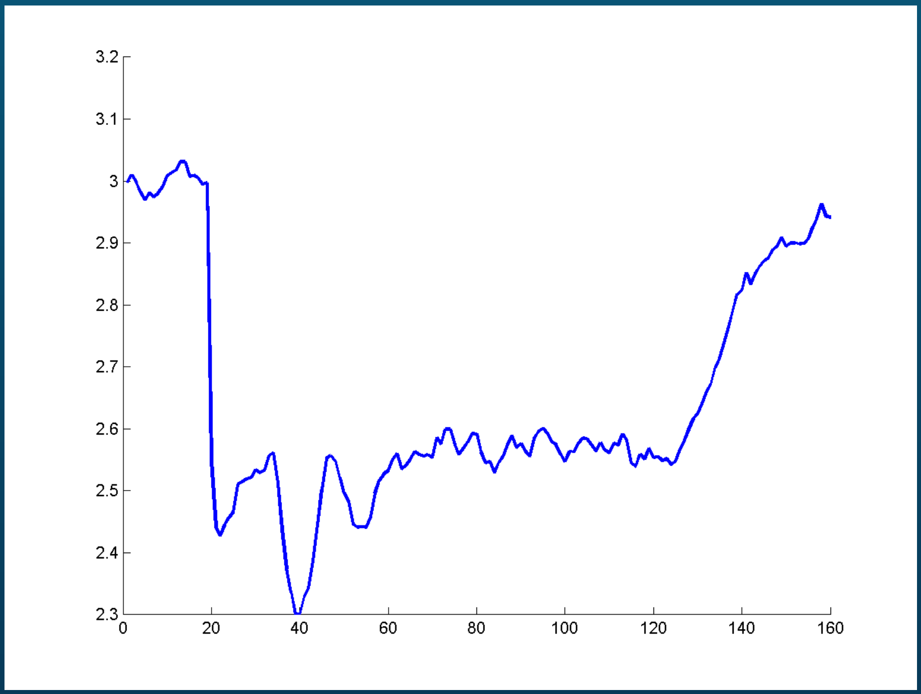
- The largest value of $\mathbf{Q}_{11}\left(k\right)$ is such that

$$\mathbf{P}_{11}\left(k-1 \mid k-2\right) = \mathbf{P}_{11}\left(k \mid k-1\right)$$

Results (Inflating Beacon Covariance)



Actual mean squared error (solid) and estimated covariances (dashed) for x and v.



Normalised mean squared error.

Locking the Map

- The previous methods all attempt to mitigate the effects of modelling errors by expanding the covariance significantly
- However, the fundamental problem is that the true state $\mathfrak{x}(k)$ enters the map in the form of a modelling error term
- Therefore, if map updates cease as soon as modelling errors occur, the correlated error terms cannot enter the map and the cross correlations terms remain correct
- This "map locking" can be achieved using a Schmidt-Kalman Filter

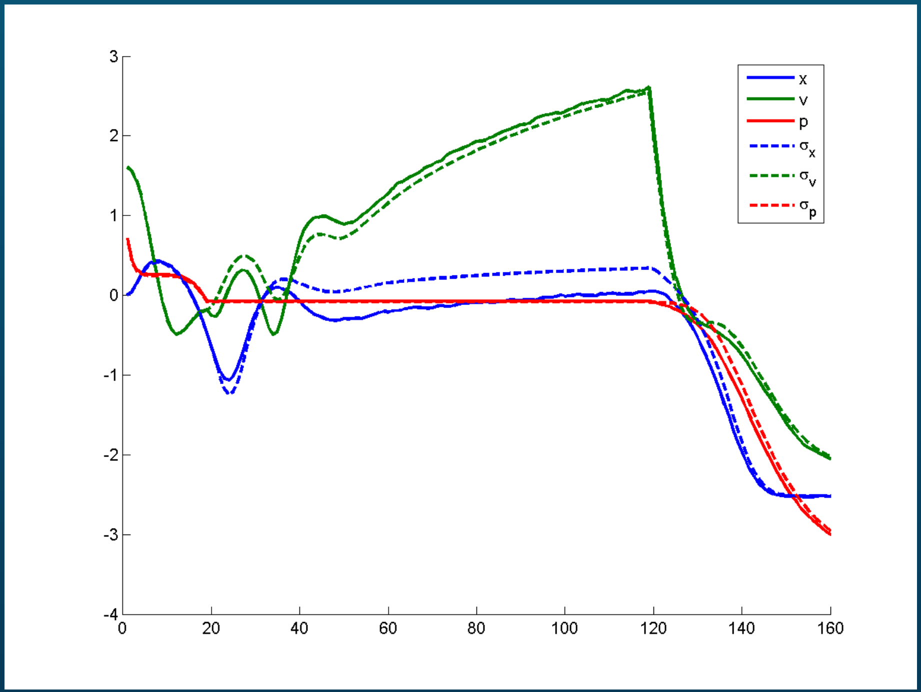
Schmidt-Kalman Filter

- The Schmidt-Kalman Filter treats a subset of the state spaces as "parameters" whose values are not updated
- $\mathbf{M}(k)$ is indicator matrix with 1s for states to update; 0s otherwise
- The minimum mean squared error estimate is

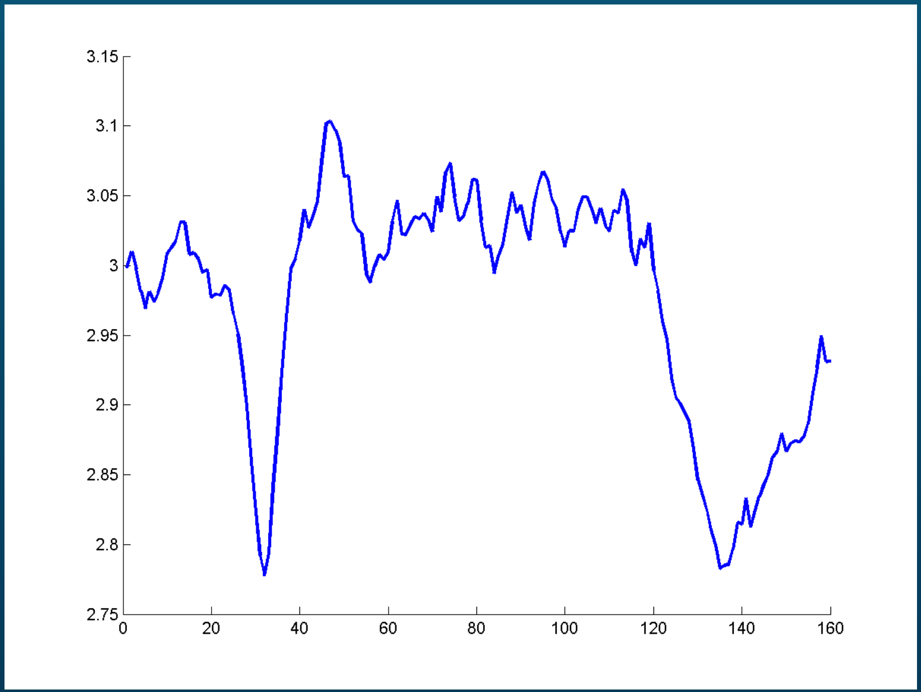
$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{M}(k)\,\mathbf{W}(k)\,\boldsymbol{\nu}(k)$$

$$\begin{aligned}\mathbf{P}(k \mid k) = {} & \Big(\mathbf{I} - \mathbf{M}(k)\,\mathbf{W}(k)\,\mathbf{H}(k)\Big)\mathbf{P}(k \mid k-1)\Big(\mathbf{I} - \mathbf{M}(k)\,\mathbf{W}(k)\,\mathbf{H}(k)\Big)^T \\ & + \mathbf{M}(k)\,\mathbf{W}(k)\,\mathbf{R}(k)\,\mathbf{M}^T(k)\,\mathbf{W}^T(k)\end{aligned}$$

Results (Locked Map)



Actual mean squared error (solid) and estimated covariances (dashed) for x and v.



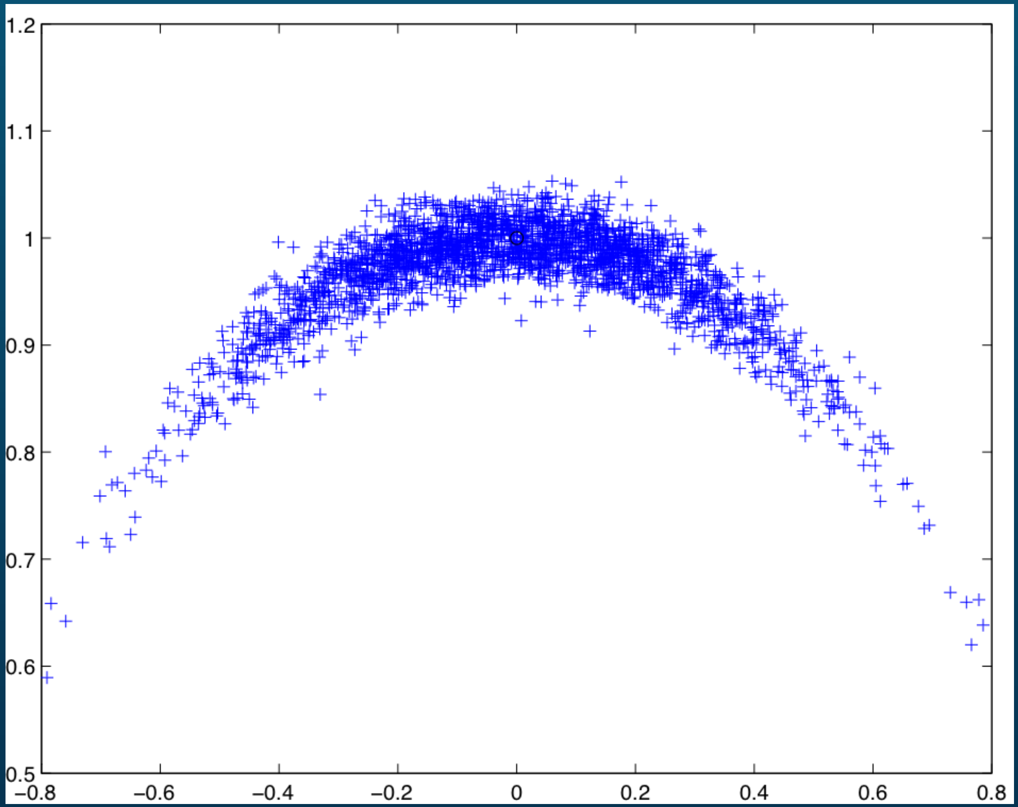
Normalised mean squared error.

Summary of Modelling Errors and SLAM

- No true system is modelled correctly and any real filter uses nominal models
- The modelling errors introduce correlated process and observation noises
- These cause SLAM to become inconsistent
- Lots of strategies seem to exist for the linear case
  - ⋆ Stabilising noise
  - ⋆ Adding noies to the beacons
  - ⋆ Map locking
- But these assume that you can identify the modelling errors as they happen

Non-Linear is not a Special Case of Linear

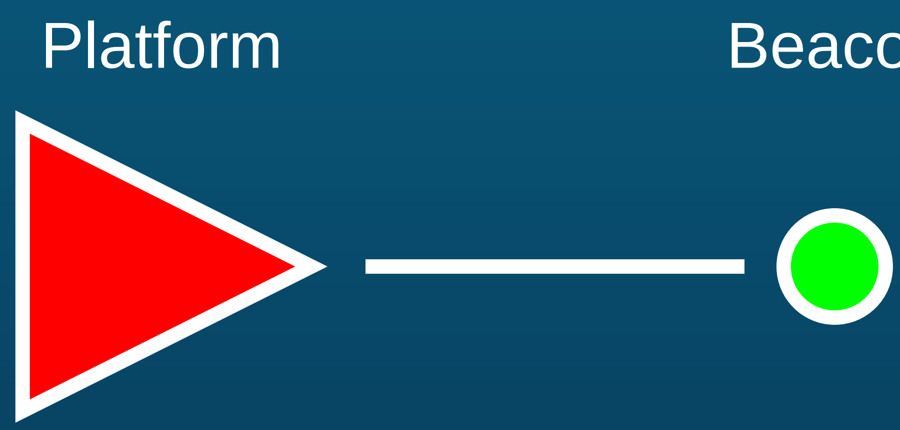
- *All* real systems are nonlinear
- The full probabilistic description *cannot* be maintained and propagated
- Normal solution is to choose approximation (EKF, IEKF, unscented, ...), and tune it until it works
- This can be made to work in tracking examples — but what about SLAM?



Approach

- Unfortunately characterising these effects in general is very hard to do
  - ⋆ No closed form solution of how the system *really* behaves
  - ⋆ Numerical methods take a very long time to run and raise questions about their own validity
  - ⋆ Lots of algebra; very little intuitive interpretation
- Therefore, we demonstrate effects by considering a tractable "point solution"
  - ⋆ Stationary vehicle with no process noise
- We extend this analysis by looking at the behaviour of SLAM algorithms and see if they accord with our intuition

Effects of Linearisation Errors on a Stationary Vehicle



- Consider a stationary vehicle (no process noise or control inputs) with state $\mathbf{x}_v(k) = (x_v, y_v, \theta)$
- It has a range-bearing sensor (r, ϕ) to perform SLAM with a single beacon
- The covariance of the vehicle orientation, $P_{\theta\theta}$, decreases if the beacon position *changes* from the initialised position

Stationary Vehicle Update Equations

- If $\mathbf{P}_{vv}(k \mid k)$ is to remain constant, the Kalman weight associated with the update must be of the form

$$\mathbf{W}(k) = \begin{pmatrix} \mathbf{W}_v(k) \\ \mathbf{W}_p(k) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{W}_p(k) \end{pmatrix} \forall k = 1, 2, \ldots$$

- This condition is equivalent to

$$\mathbf{P}(k \mid k-1)\, \boldsymbol{\nabla}^T \mathbf{h} = \begin{pmatrix} \mathbf{0} \\ \mathbf{W}_p \mathbf{S}(k) \end{pmatrix}$$

- This property is true only if a condition between the cross correlations and the observation model holds

System Equations

- The process model is

$$\mathbf{x}(2) = \mathbf{x}(1)$$

- The beacon is initialised by

$$\mathbf{p}_1(1) = \mathbf{g}_1[\mathbf{x}_v(1), \mathbf{z}_1(1)] = \begin{bmatrix} x_v + z_1^r(1)\cos\psi_1(1) \\ y_v + z_1^r(1)\sin\psi_1(1) \end{bmatrix}$$

where $\psi_1(1) = \theta_v + z_1^\phi(1)$

- The observation model is

$$\mathbf{z}_i(k) = \mathbf{h}_i[\mathbf{x}_v(k), \mathbf{p}_i(k)] = \begin{bmatrix} \sqrt{(x_v - x_i)^2 + (y_v - y_i)^2} \\ \arctan\left(\frac{y_v - y_i}{x_v - x_i}\right) \end{bmatrix}$$

Initialising the Beacon at Time Step 1

- The correlation structure between the vehicle and the beacon is created when a new beacon is initialised,

$$\hat{\mathbf{x}}_1(1 \mid 1) = \begin{bmatrix} \hat{\mathbf{x}}_v(1 \mid 0) \\ \mathbf{g}_1\left[\hat{\mathbf{x}}_v(1 \mid 0), \mathbf{z}_1(1)\right] \end{bmatrix}$$

$$\mathbf{P}_1(1 \mid 1) = \begin{pmatrix} \mathbf{P}_{vv}(1 \mid 0) & \mathbf{P}_{v1}(1 \mid 1) \\ \mathbf{P}_{1v}(1 \mid 1) & \mathbf{P}_{11}(1 \mid 1) \end{pmatrix}$$

where

$$\mathbf{P}_{v1}(1 \mid 1) = \mathbf{P}_{vv}(1 \mid 0)\, \boldsymbol{\nabla}_x^T \mathbf{g}_1$$

$$\mathbf{P}_{11}(1 \mid 1) = \boldsymbol{\nabla}_x \mathbf{g}_1 \mathbf{P}_{vv}(1 \mid 0)\, \boldsymbol{\nabla}_x^T \mathbf{g}_1 + \boldsymbol{\nabla}_w \mathbf{g}_1 \mathbf{R}(1)\, \boldsymbol{\nabla}_w^T \mathbf{g}_1$$

Updating at the kth Time Step

- At time step $k > 1$, the Kalman Weight is

$$\begin{aligned}
\mathbf{W}(k)\,\mathbf{S}(k) &= \mathbf{P}(k \mid k-1)\,\boldsymbol{\nabla}_x^T \mathbf{h}_1 \\
&= \begin{pmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vv}\boldsymbol{\nabla}_x^T \mathbf{g}_1 \\ \boldsymbol{\nabla}_x \mathbf{g}_1 \mathbf{P}_{vv} & \boldsymbol{\nabla}_x \mathbf{g}_1 \mathbf{P}_{vv} \boldsymbol{\nabla}_x^T \mathbf{g}_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\nabla}_x \mathbf{h}_1 \\ \boldsymbol{\nabla}_p \mathbf{h}_1 \end{pmatrix}^T \\
&= \begin{pmatrix} \mathbf{P}_{vv}\{\boldsymbol{\nabla}_x^T \mathbf{h}_1 + \boldsymbol{\nabla}_x^T \mathbf{g}_1 \boldsymbol{\nabla}_p^T \mathbf{h}_1\} \\ \cdots \end{pmatrix}
\end{aligned}$$

- Taking transposes, the vehicle state does not update if

$$\boldsymbol{\nabla}_x \mathbf{h}_1 + \boldsymbol{\nabla}_p \mathbf{h}_1 \boldsymbol{\nabla}_x \mathbf{g}_1 = \mathbf{0}$$

Expanding the Jacobians

- Expanding all the terms (isn't this fun?)

$$\nabla_x \mathbf{g}_1 = \begin{pmatrix} 1 & 0 & -z_1^r(1)\sin\psi_1(1) \\ 0 & 1 & z_1^r(1)\cos\psi_1(1) \end{pmatrix}$$

$$\nabla_x \mathbf{h}_1 = \begin{pmatrix} -(\hat{x}_1 - \hat{x}_v)/\hat{r}_1 & -(\hat{y}_1 - \hat{y}_v)/\hat{r}_1 & 0 \\ (\hat{y}_1 - \hat{y}_v)/\hat{r}_1^2 & -(\hat{x}_1 - \hat{x}_v)/\hat{r}_1^2 & -1 \end{pmatrix}$$

$$\nabla_p \mathbf{h}_1 = \begin{pmatrix} (\hat{x}_1 - \hat{x}_v)/\hat{r}_1 & (\hat{y}_1 - \hat{y}_v)/\hat{r}_1 \\ -(\hat{y}_1 - \hat{y}_v)/\hat{r}_1^2 & (\hat{x}_1 - \hat{x}_v)/\hat{r}_1^2 \end{pmatrix}$$

$$\hat{r}_1 = \sqrt{(\hat{x}_1 - \hat{x}_v)^2 + (\hat{y}_1 - \hat{y}_v)^2}$$

Behaviour of the No-Update Condition

- Noting that,

$$(\hat{x}_1 - \hat{x}_v)/\hat{r}_1 = \cos\hat{\psi}_1(k|k-1), \;\; (\hat{y}_1 - \hat{y}_v)/\hat{r}_1 = \sin\hat{\psi}_1(k|k-1)$$

- The update condition is

$$\nabla_x \mathbf{h}_1 + \nabla_p \mathbf{h}_1 \nabla_x \mathbf{g}_1 =$$

$$\begin{pmatrix} 0 & 0 & z_1^r(1)\Big(\sin\psi_1(1)\cos\hat{\psi}_1(k|k-1) - \cos\psi_1(1)\sin\hat{\psi}_1(k|k-1)\Big) \\ 0 & 0 & \frac{z_1^r(1)}{\hat{r}_1(k|k-1)}\Big(\cos\psi_1(1)\cos\hat{\psi}_1(k|k-1) + \sin\psi_1(1)\sin\hat{\psi}_1(k|k-1)\Big) - 1 \end{pmatrix}$$

Conditions for Invariance

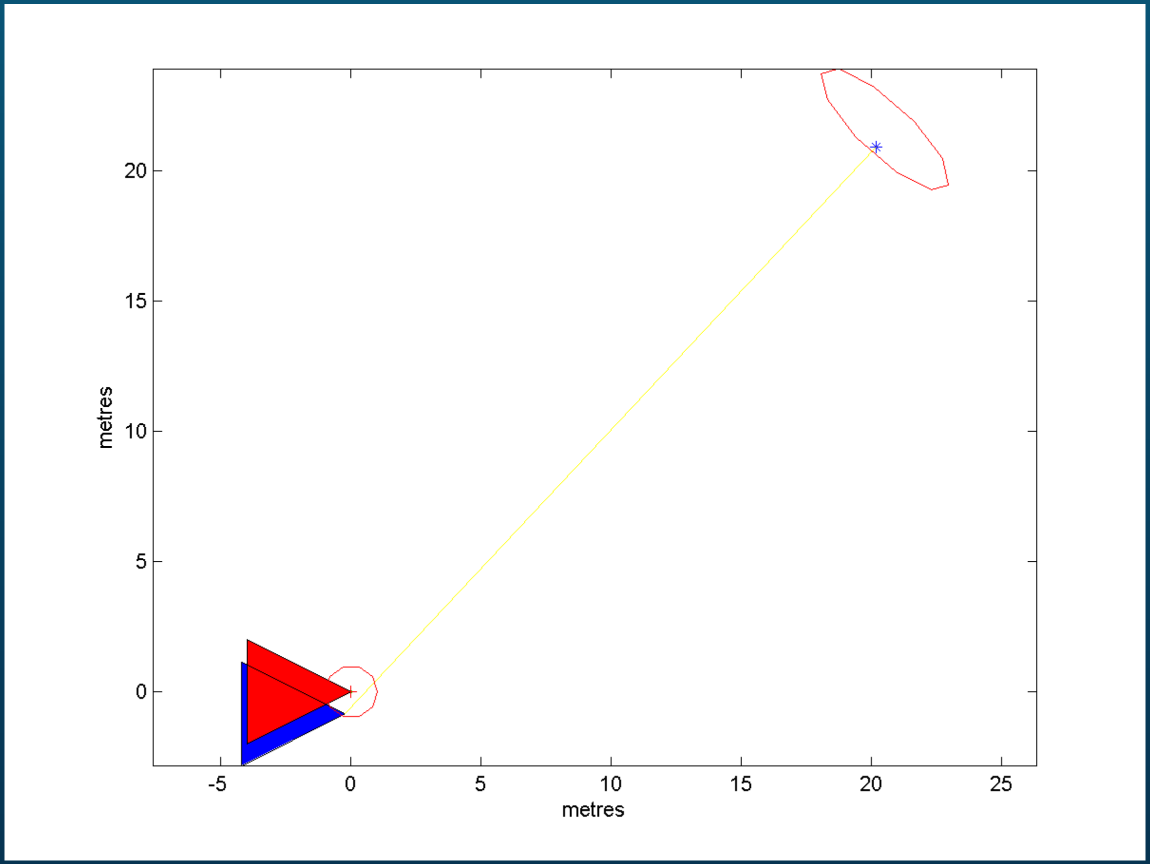
- The update condition is maintained if

$$\psi_1(1) = \hat{\psi}_1(k|k-1)$$
$$z_1^r(1) = \hat{r}_1(k|k-1)$$

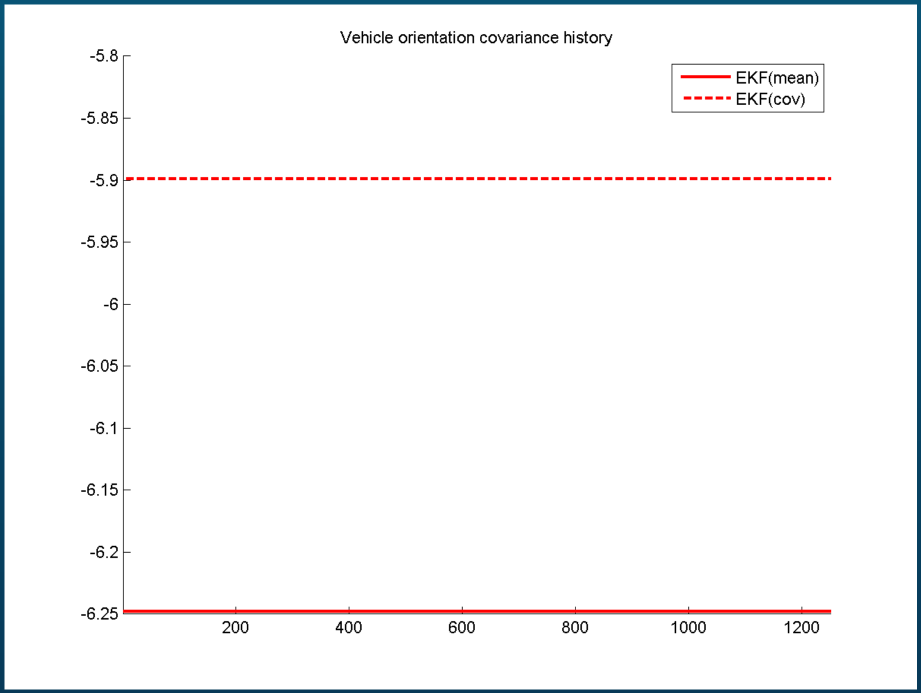
- Both $\hat{\psi}_1(k|k-1)$ and $\hat{r}_1(k|k-1)$ are computed from the pairs $\left[\hat{x}_v(k|k-1),\ \hat{y}_v(k|k-1)\right]$ and $\left[\hat{x}_1(k|k-1),\ \hat{y}_1(k|k-1)\right]$

- Since the vehicle state does not change in the prediction step, the update condition fails if the beacon moves from its initialised position

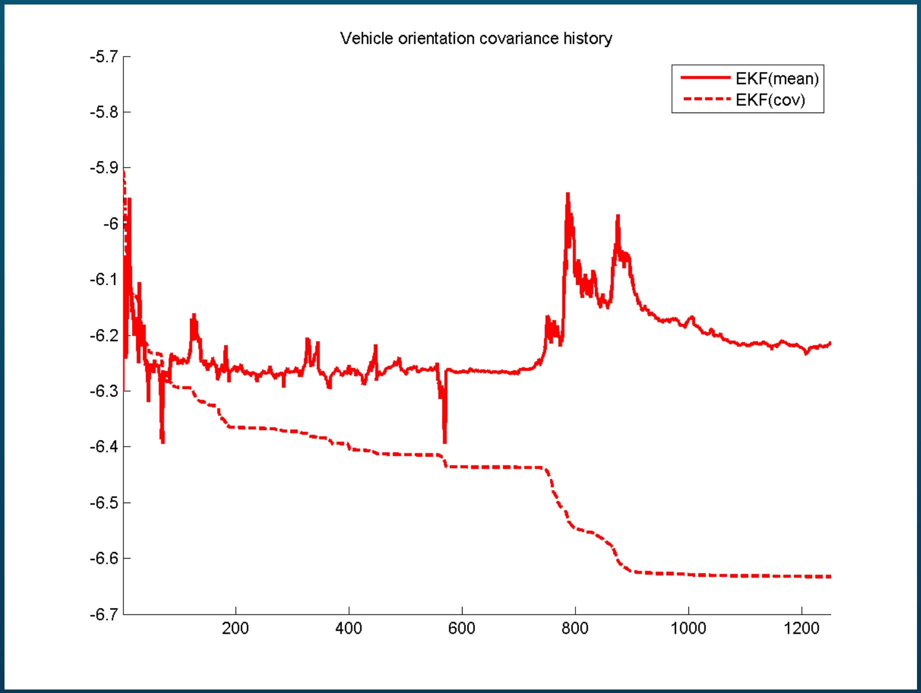
Stationary Example



Results (EKF)



Actual mean squared error (solid) and estimated covariances (dashed) for θ with no observation noise samples.

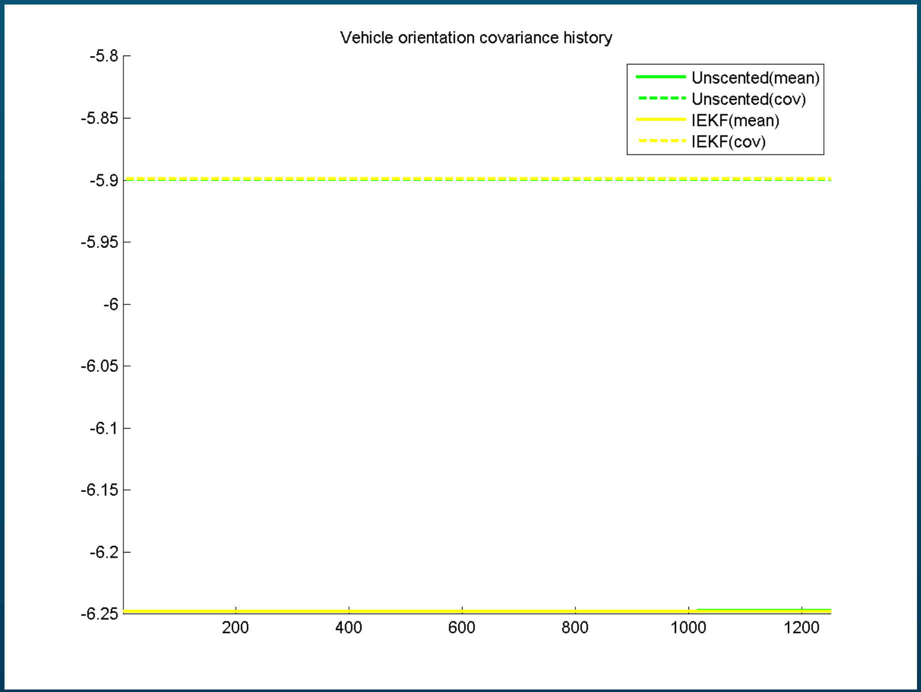


Actual mean squared error (solid) and estimated covariances (dashed) for θ with observation noise samples.

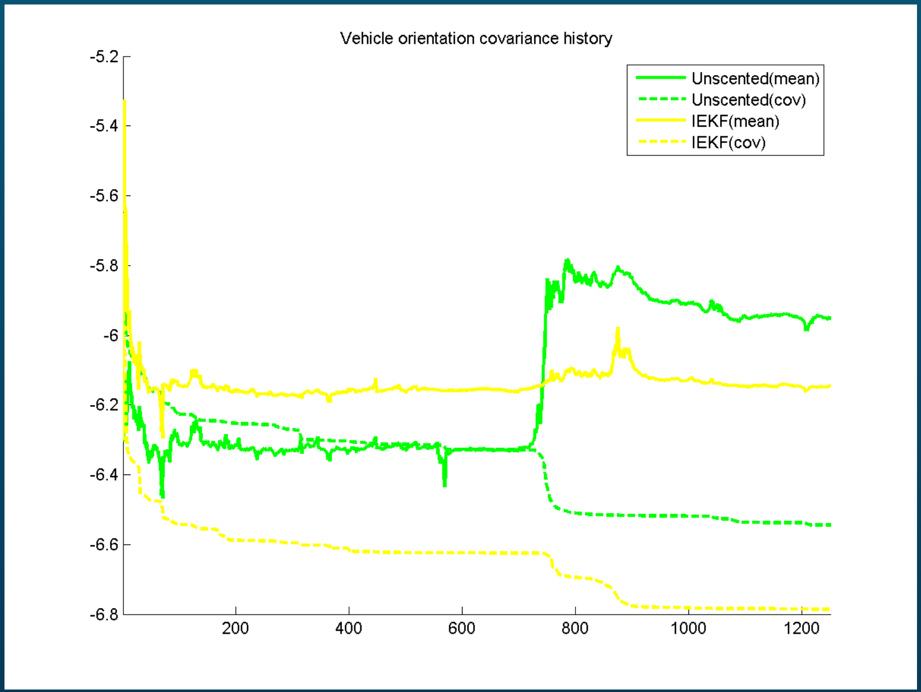
Mitigating Effects of Stationary Vehicles

- The problem is that linearisation fails to fully capture the dependence relationship between the vehicle-beacon estimate
  - ⋆ The problem is structural — stabilising noise will not prevent it
- One possible solution is to use a higher order Kalman Filter representation
- Another possible solution is to eliminate the initial vehicle pose uncertainty
  - ⋆ Build the map relative to the frame of the vehicle
  - ⋆ The initial vehicle uncertainty becomes the uncertainty of the frame itself and is placed outside of the Kalman Filter
  - ⋆ This eliminates the initial condition problem

Results (IEKF and UKF)

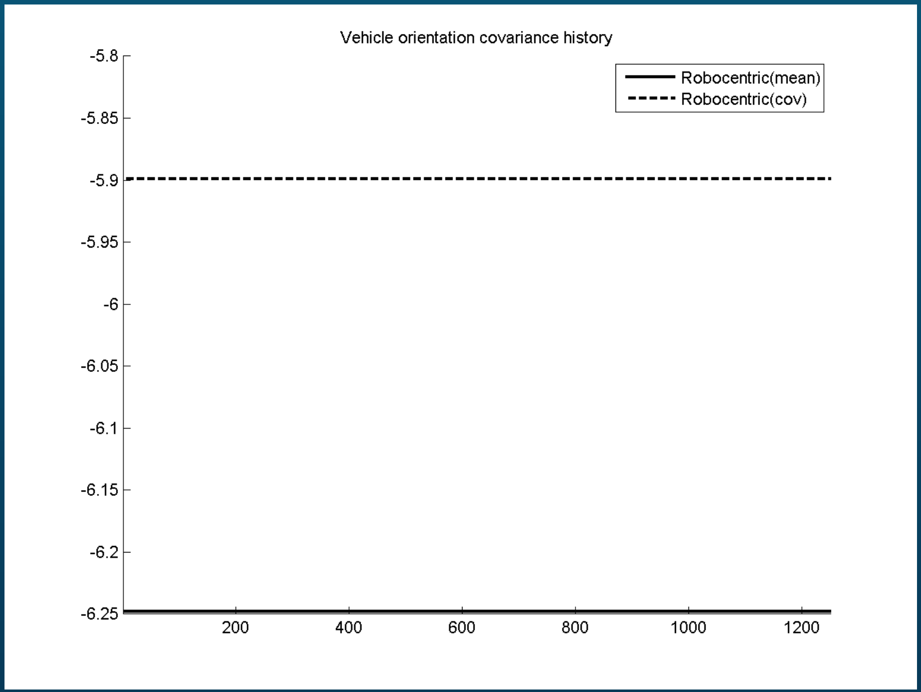


Actual mean squared error (solid) and estimated covariances (dashed) for θ with no observation noise samples.

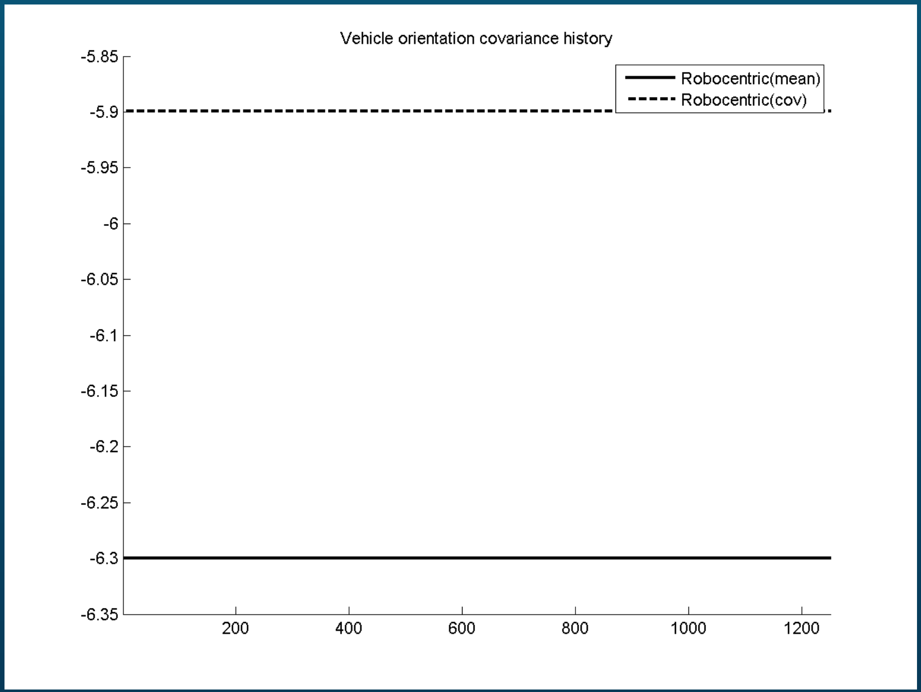


Actual mean squared error (solid) and estimated covariances (dashed) for θ with observation noise samples.

Results (Robocentric)

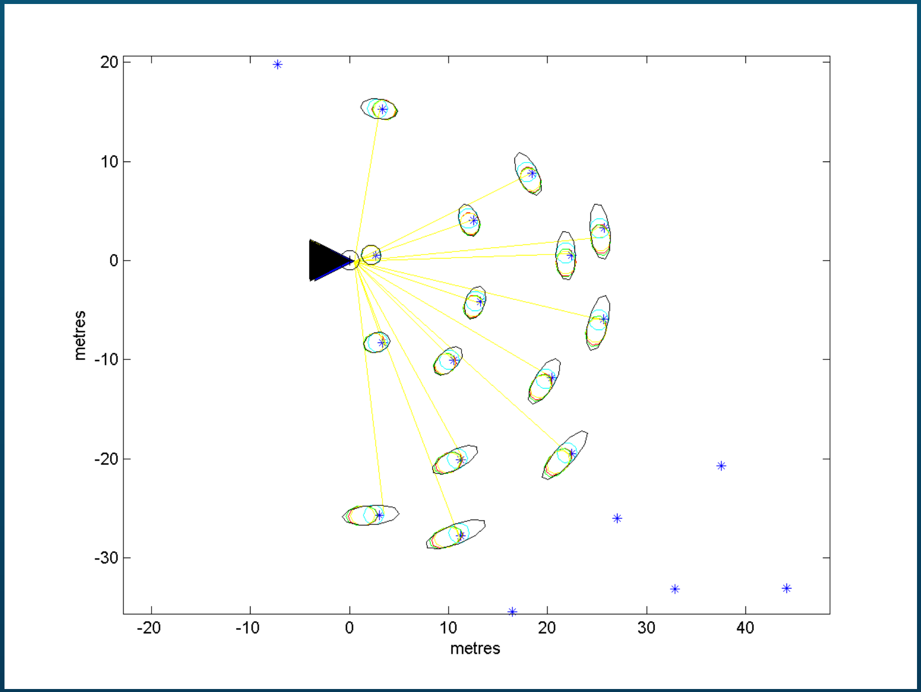


Actual mean squared error (solid) and estimated covariances (dashed) for θ with no observation noise samples.

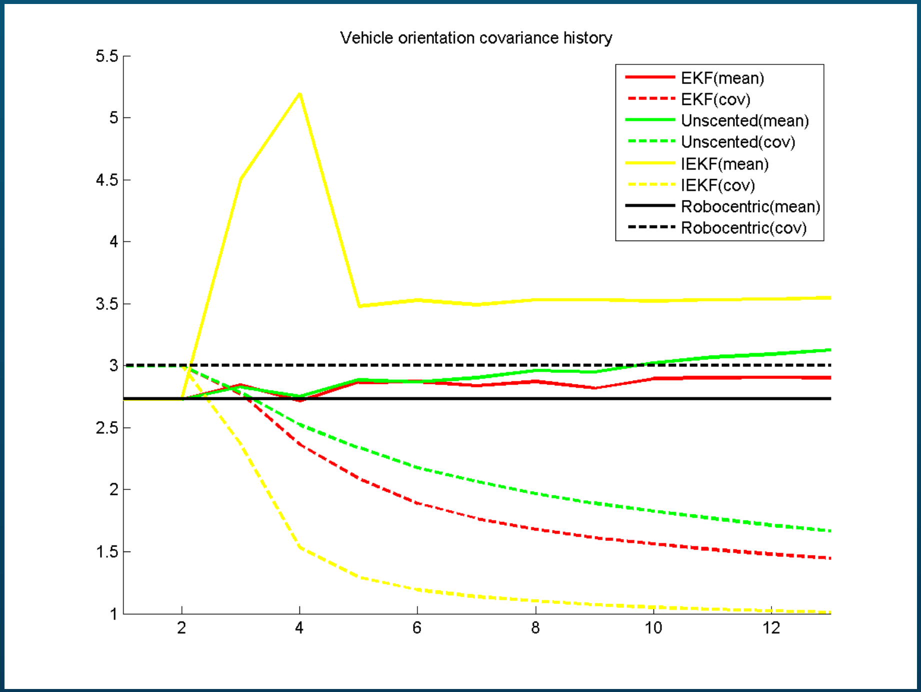


Actual mean squared error (solid) and estimated covariances (dashed) for θ with observation noise samples.

More Beacons Do *Not* Help...



Scenario when vehicle updates with a large number of beacons.

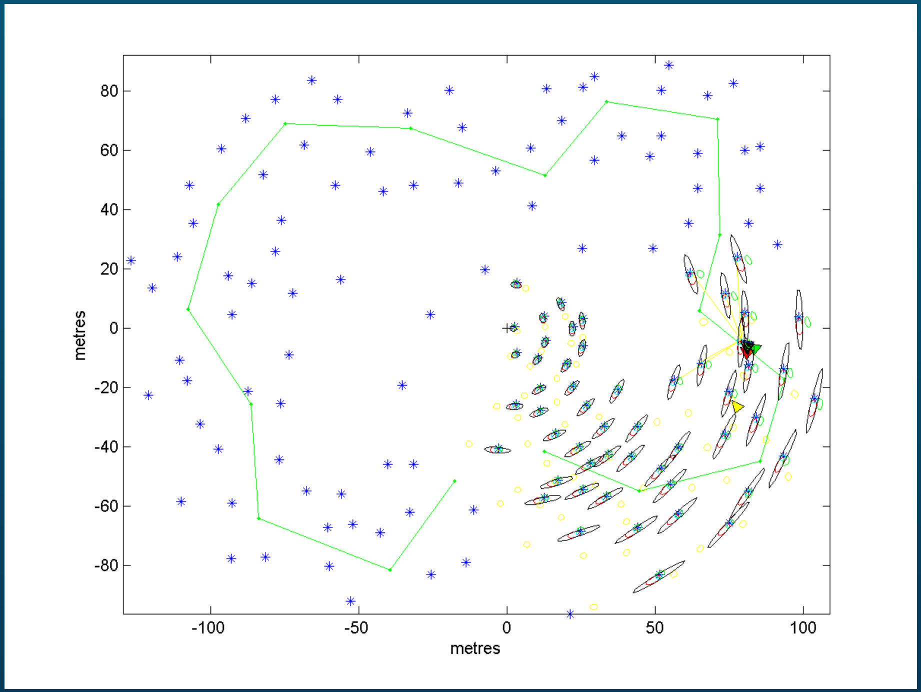


Actual mean squared error (solid) and estimated covariances (dashed) for θ with observation noise samples.

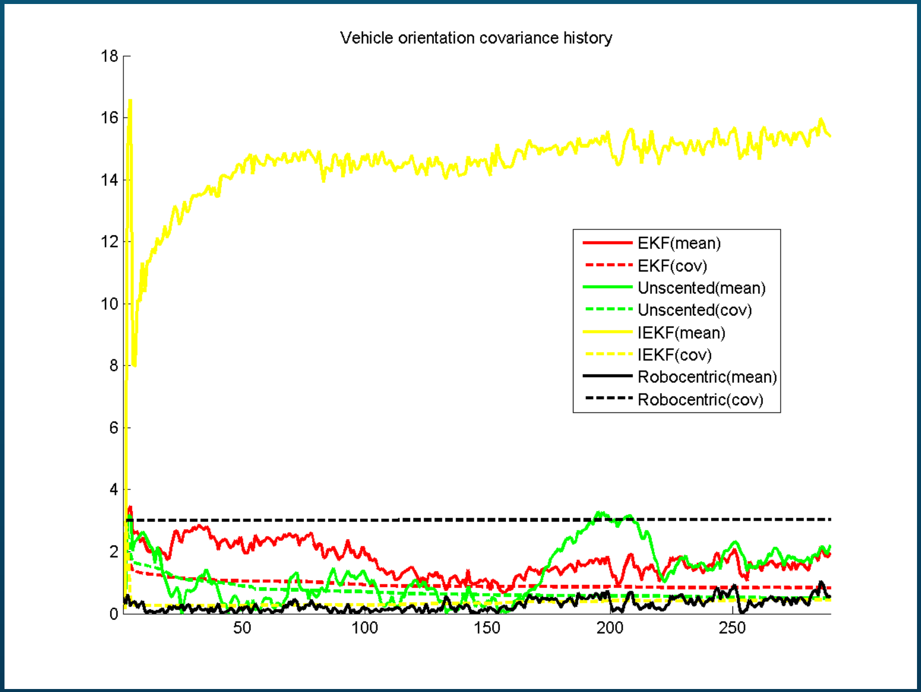
Moving Vehicle with Process Noise

- A real SLAM problem has a moving vehicle with process noise injected in it
- Qualitatively this is different
  - ⋆ The update condition does not hold for this system
  - ⋆ Process noise will help to offset the collapse in orientation covariance
- Unfortunately we cannot solve this analytically — we have to look at the vehicle behaviour

Moving Scenario



Scenario when vehicle updates with a large number of beacons.



Actual mean squared error (solid) and estimated covariances (dashed) for θ with observation noise samples.

Summary and Conclusions: Consistency

- Covariance consistency is a useful notion in a wide range of mean and covariance applications
  - ⋆ Only usable metric to determine if a filter actually works
- As an estimation problem, SLAM has a very unusual structure
  - ⋆ It is not observable
  - ⋆ Large chunks of the state space (map) have no dynamics and process noise
- Therfore, the usual "stabilising noise" cannot be used to make SLAM maps consistent

Summary and Conclusions: Modelling Errors

- Modelling errors cause SLAM to become inconsistent
  - ⋆ Cross correlations incorrect
  - ⋆ We can only make the map consistent by adding noise to the beacons
- Mitigating strategies include
  - ⋆ Better process models
  - ⋆ Formulations which do not require process models
  - ⋆ Lock the map to prevent correlated noise terms from entering the map

Summary and Conclusions: Linearisation Errors

- Linearisation errors cause SLAM to become inconsistent
  - ⋆ Cross correlations fail to capture dependence relationships
  - ⋆ It is possible that stabilising noise will work (haven't tried)
- Mitigating strategy
  - ⋆ Robocentric mapping is most successful solution tested so far